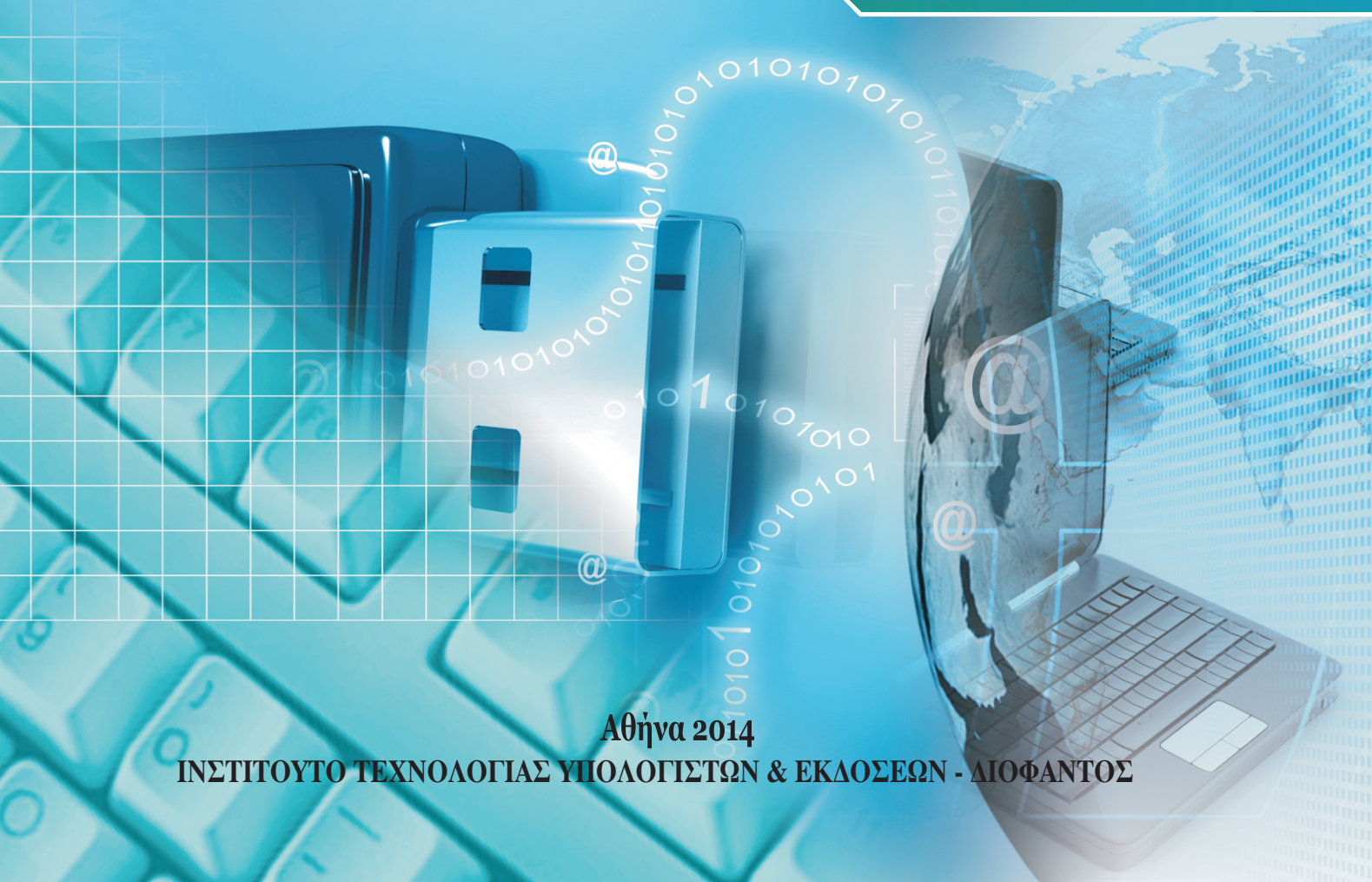
Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

Β΄ ΛΥΚΕΙΟΥ



Αθήνα 2014
ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΕΚΔΟΣΕΩΝ - ΔΙΟΦΑΝΤΟΣ

Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

Β΄ ΛΥΚΕΙΟΥ

Αθήνα 2014
ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΕΚΔΟΣΕΩΝ - ΔΙΟΦΑΝΤΟΣ

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ
Πρόεδρος: Σωτήριος Γκλαβάς

ΓΡΑΦΕΙΟ ΕΡΕΥΝΑΣ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΕΦΑΡΜΟΓΩΝ Β΄
Προϊστάμενος: Παύλος Φ. Μάραντος

ΣΥΓΓΡΑΦΕΙΣ:
Δρ. Σπυρίδων Δουκάκης, Πληροφορικός, Μαθηματικός, PIERCE-Αμερικανικό Κολλέγιο Ελλάδος
Χρήστος Δουληγέρης, Καθηγητής Τμήματος Πληροφορικής Πανεπιστημίου Πειραιώς
Δρ. Θεόδωρος Καρβουνίδης, Εκπαιδευτικός ΠΕ19
Χρήστος Κοίλιας, Καθηγητής Τμήματος Μηχανικών Πληροφορικής Τ.Ε. ΤΕΙ Αθήνας
Δρ. Αθανάσιος Πέρδος, Πληροφορικός, Φυσικός, Ελληνογαλλική Σχολή Καλαμαρί

ΣΥΝΤΟΝΙΣΤΗΣ: Χρήστος Κοίλιας

ΣΥΛΛΟΓΗ - ΕΠΕΞΕΡΓΑΣΙΑ ΥΛΙΚΟΥ:
Δημήτριος Κοτσιφάκος, Εκπαιδευτικός, ΠΕ 1708

ΚΡΙΤΕΣ-ΑΞΙΟΛΟΓΗΤΕΣ:
Παναγιώτης Βαρζάκας, Μέλος ΔΕΠ (συντονιστής)
Σοφία Τζελέπη, Σχολική Σύμβουλος, ΠΕ19
Πέτρος Ματζάκος, Εκπαιδευτικός, ΠΕ19

ΦΙΛΟΛΟΓΙΚΗ ΕΠΙΜΕΛΕΙΑ: Μαρία Κοίλια

ΕΞΩΦΥΛΛΟ: Γιώργος Σκούφος

ΣΕΛΙΔΟΠΟΙΗΣΗ: Γιώργος Σκούφος

ΑΝΑΔΟΧΟΣ: ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ

NewTech Pub.

Στουρνάρη 49Α, 106 82, Αθήνα
Τηλ. 210-38.45.594 - Fax: 210-38.08.009
E-mail: contact@newtech-publications.gr
URL: www.newtech-pub.com

«ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΓΙΑ ΤΑ ΝΕΑ ΜΑΘΗΜΑΤΑ ΤΟΥ ΓΕΝΙΚΟΥ ΛΥΚΕΙΟΥ» της Πράξης «ΝΕΟ ΣΧΟΛΕΙΟ (ΣΧΟΛΕΙΟ 21ου αιώνα) - ΝΕΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ» ΜΕ ΚΩΔ. ΟΠΣ 295450, των Αξόνων Προτεραιότητας 1, 2 και 3 - ΟΡΙΖΟΝΤΙΑ ΠΡΑΞΗ του ΕΠΙΧΕΙΡΗΣΙΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ «ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ», που συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση - Ευρωπαϊκό Κοινωνικό Ταμείο και από Εθνικούς Πόρους (ΕΣΠΑ 2007 - 2013).

ΙΕΠ
Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

Πρόλογος **7**

ΕΝΟΤΗΤΑ 1. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Κεφάλαιο 1.1. Επιστήμη των Υπολογιστών **9**
1.1.1. Εισαγωγή 9
1.1.2. Θεωρητική Επιστήμη των Υπολογιστών 9
1.1.3. Εφαρμοσμένη Επιστήμη των Υπολογιστών 10

ΕΝΟΤΗΤΑ 2. ΘΕΜΑΤΑ ΘΕΩΡΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κεφάλαιο 2.1. Πρόβλημα **13**
2.1.1. Η έννοια του προβλήματος 13
2.1.2. Κατηγορίες προβλημάτων 14
2.1.3. Υπολογιστικά προβλήματα 15
2.1.4. Διαδικασίες επίλυσης (υπολογιστικού) προβλήματος 16

Κεφάλαιο 2.2. Αλγόριθμοι **19**
2.2.1. Ορισμός αλγορίθμου 19
2.2.2. Χαρακτηριστικά αλγορίθμου 22
2.2.3. Ανάλυση Αλγορίθμων, Θεωρία Υπολογισμού, Πολυπλοκότητα Αλγορίθμων, Υπολογισιμότητα Αλγορίθμων 23
2.2.4. Βασικοί τύποι αλγορίθμων 25
2.2.5. Αναπαράσταση αλγορίθμου 27
2.2.6. Δεδομένα και αναπαράστασή τους 29
2.2.7. Εντολές και δομές αλγορίθμου 31
2.2.7.1. Εκχώρηση, Είσοδος και Έξοδος τιμών *32*
2.2.7.2. Δομή ακολουθίας *33*
2.2.7.3. Δομή επιλογής *34*
2.2.7.4. Δομή επανάληψης *38*
2.2.7.5. Κλήση αλγόριθμου από αλγόριθμο *41*
2.2.7.6. Αναδρομή *43*
2.2.8. Βασικές αλγοριθμικές λειτουργίες σε δομές δεδομένων 43
2.2.9. Εκσφαλμάτωση σε λογικά λάθη 48
2.2.10. Τεκμηρίωση 50

Κεφάλαιο 2.3. Προγραμματισμός **55**
2.3.1. Αναφορά σε γλώσσες προγραμματισμού και «Προγραμματιστικά Υποδείγματα» 55
2.3.1.1. Πρόγραμμα και Γλώσσες Προγραμματισμού *55*
2.3.1.2. Προγραμματιστικά Υποδείγματα *58*
2.3.1.3. Δομημένος Προγραμματισμός *59*
2.3.2. Σχεδίαση και συγγραφή κώδικα 62
2.3.3. Κύκλος ζωής εφαρμογής λογισμικού 70

ΕΝΟΤΗΤΑ 3. ΘΕΜΑΤΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κεφάλαιο 3.1. Λειτουργικά Συστήματα 77
3.1.1. Λογισμικό και Υπολογιστικό Σύστημα 77
3.1.2. Το Λειτουργικό Σύστημα και οι Αρμοδιότητές του 77
3.1.3. Η Δομή και η Ιεραρχία ενός Λειτουργικού Συστήματος 78
3.1.4. Βασικές Εργασίες του Λ.Σ. 78
3.1.4.1. Διαχείριση της ΚΜΕ 78
3.1.4.2. Διαχείριση της Μνήμης 79
3.1.4.3. Διαχείριση του Συστήματος Αρχείων 79
3.1.4.4. Διαχείριση Λειτουργιών Εισόδου/Εξόδου 80
3.1.5. Γνωστά Λειτουργικά Συστήματα 80

Κεφάλαιο 3.2. Πληροφοριακά Συστήματα 83
3.2.1. Τι είναι τα Πληροφοριακά Συστήματα 83
3.2.2. Αρχιτεκτονικές Αποθήκευσης 84
3.2.3. Βάσεις Δεδομένων 85
3.2.4. Γλώσσες Ερωτοαποκρίσεων (SQL, XML) 86

Κεφάλαιο 3.3. Δίκτυα 87
3.3.1. Τι είναι ένα Δίκτυο Υπολογιστών 87
3.3.2. Στοιχεία δικτύων 88
3.3.3. Κατηγορίες δικτύων 88
3.3.3.1. Είδη δικτύων ανάλογα με την τεχνολογία μετάδοσης 88
3.3.3.2. Είδη δικτύων ανάλογα με την τεχνολογία προώθησης της πληροφορίας 88
3.3.3.3. Είδη δικτύων βάσει περιοχής που καλύπτουν 89
3.3.4. Τοπολογίες Δικτύων 89
3.3.5. Σύγχρονες υπηρεσίες δικτύων 90

Κεφάλαιο 3.4. Τεχνητή Νοημοσύνη 93
3.4.1. Τι είναι η Τεχνητή Νοημοσύνη 93
3.4.2. Εξέλιξη της Τεχνητής Νοημοσύνης 94
3.4.3. Τομείς εφαρμογών της Τεχνητής Νοημοσύνης 95
3.4.4. Γλώσσες προγραμματισμού που χρησιμοποιούνται στην Τ.Ν. 96

Βιβλιογραφία 97

Γλωσσάριο 99

Πρόλογος

Το παρόν σύγγραμμα έρχεται να υπηρετήσει τη διδασκαλία του μαθήματος «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ», προσεγγίζοντας θέματα τόσο της Θεωρητικής όσο και της Εφαρμοσμένης Επιστήμης των Η/Υ. Το πρώτο μέρος καλύπτει θέματα της Θεωρητικής Επιστήμης των Υπολογιστών –από το Πρόβλημα στον Αλγόριθμο και από εκεί στον Προγραμματισμό και τις Εφαρμογές του– μέσω του οποίου επιδιώκεται η ανάπτυξη της αναλυτικής και συνθετικής σκέψης των μαθητών και των μαθητριών. Στο δεύτερο μέρος, το βιβλίο πραγματεύεται ζητήματα των βασικών τομέων της Εφαρμοσμένης Επιστήμης των Υπολογιστών, ώστε να βοηθήσει τους μαθητές και τις μαθήτριες να είναι σε θέση να κατονομάζουν και να εξηγούν επιστημονικές περιοχές της Πληροφορικής.

Αναλυτικότερα, στην πρώτη ενότητα, στο **Κεφάλαιο 1.1** επιχειρείται η διάκριση της Επιστήμης των Η/Υ σε Θεωρητική και Εφαρμοσμένη. Η παραπάνω διάκριση διαχωρίζει το σύγγραμμα σε δύο ακόμα ενότητες. Στη δεύτερη ενότητα και στο **Κεφάλαιο 2.1** επαναπροσεγγίζεται η έννοια του προβλήματος και οι φάσεις επίλυσής του και αναδεικνύονται τα υπολογιστικά προβλήματα. Στη συνέχεια, στο **Κεφάλαιο 2.2** περιγράφεται η έννοια του αλγορίθμου και των χαρακτηριστικών του, προσδιορίζονται οι διάφορες μορφές αναπαράστασής του και επεξηγούνται οι βασικές έννοιες στην Ανάλυση Αλγορίθμων. Ακολούθως, αναφέρονται οι βασικοί τύποι και οι δομές δεδομένων καθώς και οι βασικές εντολές και δομές που χρησιμοποιούνται σε έναν αλγόριθμο. Τέλος, στο κεφάλαιο αυτό, προσδιορίζεται ο τρόπος εντοπισμού και διόρθωσης λογικών λαθών σε έναν αλγόριθμο. Όλα τα παραπάνω βήματα, αποτελούν τα στάδια προετοιμασίας των μαθητών και μαθητριών, ώστε στο **Κεφάλαιο 2.3** να προχωρήσουν στη δημιουργία ενός γνωσιακού και νοητικού σχήματος που να περιλαμβάνει τα είδη και τις τεχνικές προγραμματισμού και συγχρόνως να είναι σε θέση να συνδυάζουν τις αλγοριθμικές δομές, τα δεδομένα και τις δομές δεδομένων για να δημιουργήσουν προγράμματα. Τέλος, στην ενότητα αυτή αναδεικνύεται ότι οι σημερινές εφαρμογές είναι αρκετά πολύπλοκες και η δημιουργία τους ακολουθεί συγκεκριμένα μοντέλα ανάπτυξης εφαρμογών λογισμικού που εξελίσσονται σε φάσεις.

Η τρίτη ενότητα ασχολείται με θέματα της Εφαρμοσμένης Επιστήμης των Υπολογιστών, η οποία εστιάζει τόσο στην επίλυση προβλημάτων, όσο και στη βελτίωση υπαρχουσών λύσεων στον πραγματικό κόσμο. Σε όλους τους τομείς της καθημερινότητας υπάρχει ένας τεράστιος όγκος διαθέσιμης πληροφορίας και γνώσεων, η διαχείριση του οποίου προϋποθέτει τις κατάλληλες υπολογιστικές υποδομές για την αποθήκευση της πληροφορίας καθώς και το σχεδιασμό, την ανάπτυξη και τη συντήρηση λογισμικού μέσω κατάλληλων γλωσσών προγραμματισμού και συστημάτων λογισμικού, τα οποία συνεργάζονται με την υπολογιστική υποδομή. Τέλος, προϋποθέτει την υποδομή μέσω των οποίων οι πληροφορίες διακινούνται με ασφάλεια. Από τους βασικούς τομείς της Εφαρμοσμένης Επιστήμης των Υπολογιστών που έχουν σκοπό τη διερεύνηση και την κάλυψη των προαναφερθεισών αναγκών, έχουν επιλεγεί τέσσερις. Στο **Κεφάλαιο 3.1** περιγράφονται τα Λειτουργικά Συστήματα, δηλαδή το λογισμικό που συντονίζει το υλικό και επικοινωνεί με το χρήστη. Στη συνέχεια, στο **Κεφάλαιο 3.2** επεξηγούνται τα Πληροφοριακά Συστήματα, μέσω των οποίων επιτελείται συλλογή, ανάκτηση, επεξεργασία και αποθήκευση πληροφοριών. Ακολούθως, στο **Κεφάλαιο 3.3** επιχειρείται επισκόπηση των Δικτύων Υπολογιστών για τη λήψη και την προώθηση πληροφοριών και, τέλος, στο **Κεφάλαιο 3.4** περιγράφεται η Τεχνητή Νοημοσύνη, η οποία ερευνά τρόπους ανάπτυξης υπολογιστικών μοντέλων ανθρώπινης γνώσης.

Καταβλήθηκε προσπάθεια, ώστε το υλικό του συγγράμματος να στηρίζεται στις γνώσεις και τις εμπειρίες που έχουν ήδη αποκομίσει οι μαθητές και οι μαθήτριες από την προηγούμενη σχολική τους εκπαίδευση. Με αυτόν τον τρόπο η μελέτη τους θα είναι μία ευχάριστη και δημιουργική διαδικασία οικοδόμησης της γνώσης, που πραγματοποιείται πάνω σε τεχνικές «γνωστικής σκαλωσιάς». Το βιβλίο συνοδεύεται από Παράρτημα, στο οποίο υπάρχει ένα εκτεταμένο γλωσσάρι-λεξικό όρων της Επιστήμης των Υπολογιστών.

Για την υποβοήθηση της αναγνωσιμότητας εκτός από σχήματα, πίνακες και διάφορα πλαίσια, έχουν χρησιμοποιηθεί και αρκετά εικονίδια, τα οποία χαρακτηρίζουν το μέρος του κειμένου που συνοδεύουν. Αυτά είναι:


Προερωτήσεις


Ορισμός


Ιστορικό σημείωμα


Συμβουλή


Προσοχή


Χρήσιμη πληροφορία


Σημείωση


Ανακεφαλαίωση


Λέξεις-κλειδιά

Θα θέλαμε να ευχαριστήσουμε όλους όσους μας στήριξαν στην προσπάθεια συγγραφής αυτού του βιβλίου και ιδιαιτέρως, την κυρία Αγγελική Γερούση για τη βοήθειά της στην ανάγνωση μέρους του υλικού και τις εύστοχες παρατηρήσεις της.

Παραμένουμε στη διάθεση των εκπαιδευτικών και των μαθητών για οποιεσδήποτε παρατηρήσεις ή σχόλια, ώστε να ενισχυθεί περαιτέρω το παρόν σύγγραμμα με απώτερο στόχο να υποστηρίζει τη διδασκαλία του μαθήματος.

Αθήνα, Ιούλιος 2014
Οι συγγραφείς



ΕΝΟΤΗΤΑ 1η

Βασικές Έννοιες

ΚΕΦΑΛΑΙΟ

1.1. Επιστήμη των Υπολογιστών



Επιστήμη των Υπολογιστών

Στόχοι του κεφαλαίου είναι οι μαθητές

- ✓ να περιγράφουν τους βασικούς τομείς της Επιστήμης των Υπολογιστών και
- ✓ να μπορούν να αναφερθούν στα πεδία τόσο της Θεωρητικής όσο και σε αυτά της Εφαρμοσμένης Επιστήμης των Υπολογιστών.

1.1 Η Επιστήμη των Υπολογιστών

Η Επιστήμη των Υπολογιστών μελετά τα θεωρητικά θεμέλια και τη φύση των πληροφοριών, των αλγορίθμων και των υπολογισμών, καθώς και τις τεχνολογικές εφαρμογές τους σε αυτοματοποιημένα υπολογιστικά συστήματα, από τις σκοπιές σχεδίασης, ανάπτυξης, υλοποίησης, διερεύνησης και ανάλυσης. Η Επιστήμη των Υπολογιστών διακρίνεται σε δύο μεγάλες ενότητες: τη Θεωρητική και την Εφαρμοσμένη.

1.2 Θεωρητική Επιστήμη των Υπολογιστών

Η Θεωρητική Επιστήμη των Υπολογιστών (Theoretical Computer Science) ερευνά κυρίως το σχεδιασμό των αλγορίθμων και των υπολογιστικών μεθόδων που χρησιμοποιούνται για την άντληση, την επεξεργασία, την ανάλυση και την αποθήκευση πληροφοριών. Βασικές έννοιες της Θεωρητικής Επιστήμης των Υπολογιστών, είναι η ***Ανάλυση Αλγορίθμων***, η ***Θεωρία Υπολογισιμότητας*** και η ***Θεωρία Πολυπλοκότητας***. Υπάρχει μία διαρκής αλληλεπίδραση μεταξύ της Θεωρητικής και της Εφαρμοσμένης Επιστήμης των Υπολογιστών. Για παράδειγμα, η ***Θεωρία Γλωσσών Προγραμματισμού***, η οποία μελετά προσεγγίσεις για την περιγραφή των υπολογισμών, οδηγεί στην ανάπτυξη γλωσσών προγραμματισμού και το σχεδιασμό λογισμικού και εφαρμογών.

1.3 Εφαρμοσμένη Επιστήμη των Υπολογιστών

Η Εφαρμοσμένη Επιστήμη των Υπολογιστών (Applied Computer Science) μελετά τρόπους εφαρμογής της Θεωρίας των Υπολογιστών για την επίλυση προβλημάτων στον πραγματικό κόσμο. Βασικά επιστημονικά πεδία που εντάσσονται στην Εφαρμοσμένη Επιστήμη των Υπολογιστών είναι:



Προερωτήσεις

- Τι ερευνά η Επιστήμη των Υπολογιστών; Ποιες επιστημονικές περιοχές προσπαθεί να εξελίξει;
- Πώς συνδέεται το Θεωρητικό της μέρος με το Εφαρμοσμένο;
- Πώς συνδέεται η εφαρμογή της με την επίλυση προβλημάτων του πραγματικού κόσμου;



Η Επιστήμη των Υπολογιστών ως διακριτή επιστήμη προέκυψε κατά τη δεκαετία του 1940 χάρη στην εύρεση των μαθηματικών ιδιοτήτων του υπολογισμού και την κατασκευή ηλεκτρονικών υπολογιστικών μηχανών.

Η Ανάλυση Αλγορίθμων ασχολείται με τον σχεδιασμό και την ανάλυση της πολυπλοκότητας των αλγορίθμων.

Η Θεωρία Υπολογισιμότητας ερευνά αν και πόσο αποδοτικά κάποια προβλήματα μπορούν να επιλυθούν με συγκεκριμένα υπολογιστικά μοντέλα.

Η Θεωρία Πολυπλοκότητας μελετά τους πόρους που απαιτούνται για την επίλυση ενός προβλήματος βάσει ενός συγκεκριμένου αλγορίθμου.



Ξεκινήστε την αναζήτησή σας από την κατηγοριοποίηση του οργανισμού ACM.

- Ο σχεδιασμός υλικού για την κατασκευή των υπολογιστών, όπως ο σκληρός δίσκος, η κεντρική μονάδα επεξεργασίας κτλ.
- Ο σχεδιασμός, η ανάπτυξη και η συντήρηση λογισμικού, όπως των λειτουργικών συστημάτων τα οποία συνεργάζονται με το υλικό, καθώς και των ποικίλων προγραμμάτων που αναπτύσσονται με τη βοήθεια των γλωσσών προγραμματισμού.
- Ο σχεδιασμός πληροφοριακών συστημάτων για τη συλλογή, ανάκτηση, επεξεργασία και αποθήκευση πληροφοριών.
- Η τεχνητή νοημοσύνη, η οποία ερευνά τρόπους ανάπτυξης υπολογιστικών μοντέλων ανθρώπινης γνώσης.
- Ο σχεδιασμός δικτύων υπολογιστών για την παραγωγή, τη λήψη και την προώθηση πληροφοριών.
- Ο σχεδιασμός βάσεων δεδομένων και συστημάτων διαχείρισης βάσεων δεδομένων για την υποστήριξη πληροφοριακών συστημάτων.
- Η ασφάλεια των υπολογιστών, δηλαδή το σύνολο των μεθόδων που χρησιμοποιούνται για την προστασία πληροφοριών ή υπηρεσιών από φθορά, αλλοίωση ή μη εξουσιοδοτημένη χρήση.

Στην συνέχεια του βιβλίου θα μελετηθούν θέματα τόσο της Θεωρητικής (αλγόριθμοι και προγραμματισμός) όσο και της Εφαρμοσμένης Επιστήμης Υπολογιστών (όπως τα Λειτουργικά Συστήματα, τα Πληροφοριακά Συστήματα, τα Δίκτυα Υπολογιστών και η Τεχνητή Νοημοσύνη).

Ανακεφαλαίωση

Η Επιστήμη Υπολογιστών πραγματεύεται δύο μεγάλες θεματικές ενότητες - τη Θεωρητική και την Εφαρμοσμένη - οι οποίες περιλαμβάνουν πολλούς επί μέρους κλάδους με έμφαση τόσο στην διαχείριση πληροφοριών όσο και στην επίλυση προβλημάτων στον πραγματικό κόσμο.



Χρήσιμοι Υπερσύνδεσμοι

http://www.acm.org
Association for Computing Machinery

http://www.computer.org/portal/web/guest/home
IEEE-Computer Society

http://aisnet.org
AIS: Association for Information Systems

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Πώς αντιλαμβάνεστε την Επιστήμη των Υπολογιστών;
2. Να αναφέρετε τουλάχιστον τρία επιστημονικά πεδία που εντάσσονται στην Εφαρμοσμένη Επιστήμη των Υπολογιστών, αναζητώντας σχετικές πληροφορίες στο διαδίκτυο.
3. Να αναφέρετε τρεις τομείς της Θεωρητικής Πληροφορικής οι οποίοι έχουν άμεση εφαρμογή σε προβλήματα που αντιμετωπίζει η Εφαρμοσμένη Πληροφορική.
4. Να αναζητήσετε στο Διαδίκτυο, εργαζόμενοι σε ομάδες, όρους που σχετίζονται με την Επιστήμη των Υπολογιστών, τους τομείς της, τα πεδία εφαρμογής καθεμιάς και να συσχετίσετε τις έννοιες μεταξύ τους. Με βάση την αναζήτηση αυτή, να γίνει απαρίθμηση των πλέον γνωστών τομέων και ο διαχωρισμός τους σε θεωρητικούς και εφαρμοσμένους.



Λέξεις κλειδιά
Θεωρητική Επιστήμη των Υπολογιστών, Εφαρμοσμένη Επιστήμη των Υπολογιστών.



ΕΝΟΤΗΤΑ 2η

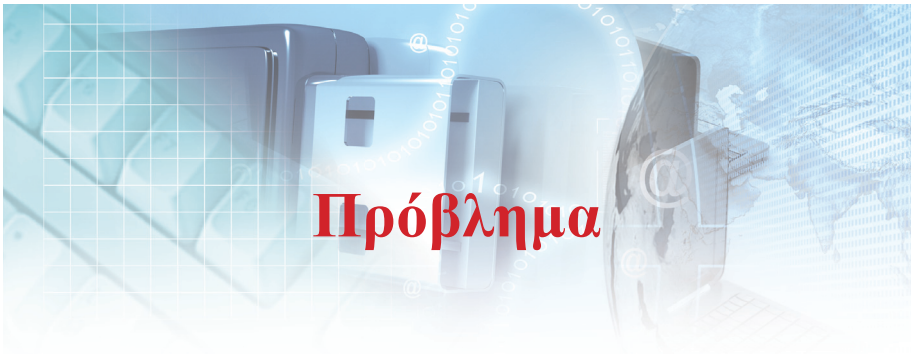
Θέματα Θεωρητικής Επιστήμης των Υπολογιστών

ΚΕΦΑΛΑΙΑ

2.1. Πρόβλημα

2.2. Αλγόριθμοι

2.3. Προγραμματισμός



Πρόβλημα

Στόχοι του κεφαλαίου αυτού είναι να μπορούν οι μαθητές:

- ✓ να περιγράφουν την έννοια του προβλήματος
- ✓ να κατατάσσουν ένα πρόβλημα στην κατηγορία που ανήκει
- ✓ να διακρίνουν την ύπαρξη υπολογιστικών και μη προβλημάτων
- ✓ να αναφέρουν τις φάσεις επίλυσης ενός υπολογιστικού προβλήματος.

2.1.1 Η έννοια του προβλήματος

Οι άνθρωποι, από την πρώτη στιγμή της ύπαρξής τους, ήρθαν αντιμέτωποι με ποικίλα προβλήματα, τόσο στις καθημερινές τους δραστηριότητες όσο και σε διάφορους επιστημονικούς τομείς. Κάνοντας μια αναδρομή στην ιστορία είναι δυνατό να εντοπιστούν πληθώρα προβλημάτων.

- Ο Όμηρος στην Οδύσσεια περιγράφει τα προβλήματα που αντιμετώπιζε ο Οδυσσέας για να φτάσει στην Ιθάκη.
- Το πρόβλημα που κλήθηκε να επιλύσει ο Αρχιμήδης με τη βασιλική κορώνα που οδήγησε στη γνωστή φράση του «Εύρηκα-Εύρηκα».
- Το πρόβλημα μέτρησης του χρόνου, το οποίο αντιμετωπίστηκε με τη χρήση της κλεψύδρας και του εκκρεμούς.
- Τα προβλήματα επιδημιών στην ανθρωπότητα και η αντιμετώπισή τους με εμβόλια.
- Το πρόβλημα του «ιού του 2000» και η αντιμετώπισή του, ώστε τα υπολογιστικά συστήματα να λειτουργήσουν σωστά την 1/1/2000.

Όπως φαίνεται, η ύπαρξη προβλημάτων είναι ένα διαχρονικό φαινόμενο. Στην εποχή μας, οι άνθρωποι έρχονται αντιμέτωποι με προβλήματα στον προσωπικό, στον επαγγελματικό και στον κοινωνικό χώρο γενικότερα. Όλοι οι άνθρωποι δεν αντιμετωπίζουν τα ίδια προβλήματα και επιπλέον δίνουν διαφορετική σημασία και βαρύτητα σε αυτά. Ωστόσο υπάρχουν προβλήματα που αναγνωρίζονται από τους περισσότερους ως πολύ σημαντικά.

Τα προβλήματα εκτός από δυσάρεστες ή πιεστικές καταστάσεις που απαιτούν λύση (περιβαλλοντικά προβλήματα, κοινωνικά προβλήματα, προσωπικά προβλήματα κ.ά.) μπορούν να είναι είτε ενδιαφέρουσες προκλήσεις (π.χ. η επίλυση ενός γρίφου ή η νίκη σε ένα παιχνίδι σκάκι), είτε ευκαιρίες για να προκύψει κάτι ωφέλιμο για την κοινωνία μέσω της επίλυσής τους (π.χ. νέα ασφαλέστερα υλικά για την κατασκευή αυτοκινήτων, τρισδιάστατες εκτυπώσεις κ.ά.).



Προερωτήσεις

• Συζητήστε με τους συμμαθητές σας και καταγράψτε δύο προβλήματα.
• Ποια είναι τα βασικότερα προβλήματα της ανθρωπότητας;
• Ρωτήστε το συμμαθητή σας ποιο θεωρεί το σημαντικότερο πρόβλημα της ανθρωπότητας και ποιο θεωρεί το σημαντικότερο πρόβλημα που χρήζει αντιμετώπισης στο σχολείο.



Τα προβλήματα δεν είναι απαραίτητα μαθηματικές καταστάσεις που απαιτούν αντιμετώπιση.



Εικόνα 2.1. Ο κύβος του Ρούμπικ (Rubik)



«Η ζωή είναι επίλυση προβλημάτων», (Καρλ Πόππερ, Karl Popper, 1994)

Όλα τα προβλήματα δεν μπορούν να αντιμετωπιστούν με έναν ενιαίο και μοναδικό τρόπο. Επιπλέον, κάθε ξεχωριστό πρόβλημα μπορεί να αντιμετωπίζεται και να επιλύεται με ποικίλους τρόπους, ενώ συγχρόνως μπορεί να έχει πολλές λύσεις (το πρόβλημα οργάνωσης μιας εκπαιδευτικής επίσκεψης).

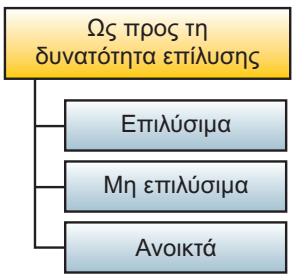
Με τον όρο Πρόβλημα προσδιορίζεται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής.

Η διατύπωση ενός προβλήματος και η αντιμετώπισή του, αποτελούν ζητήματα που απαιτούν ικανότητες ορθολογικής, αναλυτικής και συνθετικής σκέψης, αλλά και σωστό χειρισμό της φυσικής γλώσσας. Επιπλέον, οι δεξιότητες που αποκτούνται από την ενασχόληση με τα προβλήματα (διατύπωση και αντιμετώπισή τους), αποτελούν χρήσιμα εφόδια για κάθε ανθρώπινη δραστηριότητα.

2.1.2 Κατηγορίες Προβλημάτων

Τα προβλήματα ανάλογα με τη δυνατότητα επίλυσης διακρίνονται σε τρεις κατηγορίες.

Επιλύσιμα είναι εκείνα τα προβλήματα για τα οποία η λύση έχει βρεθεί και έχει διατυπωθεί.



Εικόνα 2.2. Κατηγορίες προβλημάτων

Παραδείγματα επιλύσιμων προβλημάτων είναι η αποψίλωση μιας έκτασης γης (ο καθαρισμός δηλαδή ενός χωραφιού από κάθε είδους βλάστηση), η επίλυση της δευτεροβάθμιας εξίσωσης κ.ά..

Μη επιλύσιμα χαρακτηρίζονται εκείνα τα προβλήματα για τα οποία έχει αποδειχτεί, ότι δεν επιδέχονται λύση.

Το πρόβλημα του τετραγωνισμού του κύκλου με κανόνα και διαβήτη που είχε διατυπωθεί από τους αρχαίους ελληνιστικούς χρόνους είναι ένα τέτοιο πρόβλημα. Παρότι το πρόβλημα επιδέχεται προσεγγιστική λύση, δεν μπορεί να λυθεί με τη χρήση κανόνα και διαβήτη.

Ανοικτά ονομάζονται τα προβλήματα για τα οποία η λύση τους δεν έχει ακόμα βρεθεί, ενώ ταυτόχρονα δεν έχει αποδειχτεί, ότι δεν επιδέχονται λύση.



Ένα άλλο μη επιλύσιμο πρόβλημα είναι η εύρεση ακέραιων λύσεων οποιασδήποτε διοφαντικής εξίσωσης (ακέραια πολυωνυμική εξίσωση) όπως της $6x + 15y = 4$. Το 1970 αποδείχτηκε ότι το πρόβλημα είναι μη επιλύσιμο.

Ένα τέτοιο παράδειγμα είναι το πρόβλημα της ενοποίησης των τεσσάρων πεδίων δυνάμεων (βαρυτικού, ηλεκτρομαγνητικού, ασθενούς πυρηνικού και ισχυρού πυρηνικού), το οποίο προς το παρόν, δεν έχει επιλυθεί. Επίσης, η εικασία του Γκόλντμπαχ (Goldbach, κάθε άρτιος μπορεί να γραφτεί ως άθροισμα δύο πρώτων αριθμών) αποτελεί ένα ανοικτό πρόβλημα αφού δεν έχει ακόμα αποδειχθεί.

2.1.3 Υπολογιστικά Προβλήματα

Στις αρχές του 20ου αιώνα, ο Ντέβιντ Χίλμπερτ (David Hilbert) παρουσίασε έναν κατάλογο προβλημάτων, εκ των οποίων το τελευταίο έθετε το ερώτημα «αν υπάρχει αλγόριθμος που μπορεί να αποφασίσει την αλήθεια οποιασδήποτε λογικής πρότασης που αφορούσε τους φυσικούς αριθμούς». Με το ερώτημα αυτό ουσιαστικά ρωτούσε «αν μπορεί να αυτοματοποιηθεί η διαδικασία επίλυσης όλων των μαθηματικών προβλημάτων». Το 1931, το θεώρημα της μη πληρότητας του Κουρτ Γκέντελ (Kurt Gödel) έδειξε ότι, σε οποιαδήποτε γλώσσα που έχει την ισχύ να περιγράψει τις ιδιότητες των φυσικών αριθμών, υπάρχουν αληθείς προτάσεις των οποίων η αλήθεια δεν μπορεί να βεβαιωθεί με κανέναν αλγόριθμο. Με τον τρόπο αυτό, απέδειξε ότι υπάρχουν μερικές συναρτήσεις οι οποίες δεν μπορούν να αναπαρασταθούν από έναν αλγόριθμο, και άρα δεν μπορούν να υπολογιστούν. Στη συνέχεια, ο Άλαν Τιούρινγκ (Alan Turing) όρισε τη μηχανή Turing η οποία είναι ικανή να υπολογίσει οποιαδήποτε υπολογίσιμη συνάρτηση και έδειξε επίσης ότι υπήρχαν μερικές συναρτήσεις τις οποίες καμία μηχανή Turing δεν μπορεί να υπολογίσει.



Εικόνα 2.3. David Hilbert και Kurt Gödel



Εικόνα 2.4. Alan Turing και Pierre de Fermat

Από τα παραπάνω φάνηκε ότι τα προβλήματα με βάση τη δυνατότητα επίλυσής τους μέσω του υπολογιστή, μπορούν να διακριθούν σε υπολογιστικά και μη υπολογιστικά.

Οποιοδήποτε πρόβλημα μπορεί να λυθεί και μέσω του υπολογιστή, χαρακτηρίζεται υπολογιστικό πρόβλημα.

Για να λυθεί ένα πρόβλημα με τη βοήθεια του υπολογιστή, χρειάζεται να διατυπωθεί το αντίστοιχο υπολογιστικό πρόβλημα και στη συνέχεια να υλοποιηθεί η επίλυσή του μέσω του υπολογιστή.

Παραδείγματα υπολογιστικών προβλημάτων είναι:

- Η επίλυση της δευτεροβάθμιας εξίσωσης.
- Η ταξινόμηση των μαθητών σε αλφαβητική σειρά.
- Η αναζήτηση και ο υπολογισμός της χιλιομετρικά συντομότερης διαδρομής που θα κάνει ένας ταχυδρόμος για να επισκεφθεί δέκα χωριά και να επιστρέψει στο χωριό από όπου ξεκίνησε περνώντας μόνο μία φορά από κάθε χωριό, με βάση έναν δεδομένο χάρτη των χωριών και των δρόμων που συνδέουν τα χωριά.
- Η εύρεση λέξης που να ξεκινά από ένα γράμμα και να τελειώνει σε ένα άλλο γράμμα.

Από την άλλη, τα μη υπολογιστικά προβλήματα δεν μπορούν να λυθούν από έναν υπολογιστή ή από άλλα μηχανικά μέσα. Για παράδειγμα, καμία μηχανή δεν μπορεί γενικά να αποφανθεί αν ένα δεδομένο πρόγραμμα θα επιστρέψει απάντηση για μια δεδομένη είσοδο, ή αν θα εκτελείται για πάντα.



Το θεώρημα του Πιέρ ντε Φερμά (Pierre de Fermat): «Τρεις θετικοί αριθμοί a, b, c δεν μπορούν να ικανοποιήσουν την εξίσωση $a^n+b^n=c^n$ για κάθε ακέραιο αριθμό n>2» διατυπώθηκε από τον ίδιο το 1637 ως σημείωση στο βιβλίο του «Αριθμητικά του Διόφαντου». Η επιτυχής απόδειξη του θεωρήματος δημοσιεύθηκε το 1995.



Για να διατυπωθεί ένα πρόβλημα μπορεί να χρησιμοποιηθεί οποιοδήποτε μέσο με συνηθέστερα τον προφορικό ή το γραπτό λόγο. Ο λόγος όμως όταν χρησιμοποιείται για την επικοινωνία, χρειάζεται να χαρακτηρίζεται από σαφήνεια.



Η άστοχη χρήση ορολογίας και η λανθασμένη σύνταξη, μπορούν να προκαλέσουν παρερμηνείες και παραπλανήσεις. Ωστόσο, παρερμηνείες μπορούν να υπάρξουν ακόμα και σε περιπτώσεις όπου όλοι οι λεξικολογικοί και συντακτικοί κανόνες τηρούνται.



Στη διαγραμματική αναπαράσταση, το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο. Κάθε ένα από τα απλούστερα προβλήματα, αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο.
Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα, σχηματίζονται ένα επίπεδο χαμηλότερα.

2.1.4 Διαδικασίες επίλυσης (υπολογιστικού) προβλήματος

Η προσπάθεια αντιμετώπισης και επίλυσης ενός προβλήματος προϋποθέτει αρχικά την πλήρη κατανόηση του προβλήματος. Η ***κατανόηση*** ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων, της σωστής διατύπωσης εκ μέρους του δημιουργού του και της αντίστοιχα σωστής ερμηνείας από τη μεριά εκείνου που καλείται να το αντιμετωπίσει.

Η κατανόηση του προβλήματος είναι βασική προϋπόθεση για να ξεκινήσει η διαδικασία ανάλυσης του προβλήματος σε άλλα απλούστερα και ο διαχωρισμός των κύριων στοιχείων του προβλήματος σε σχέση με τα δευτερεύοντα στοιχεία (αφαίρεση). Η ***ανάλυση-αφαίρεση*** αποτελεί το δεύτερο βήμα στην διαδικασία επίλυσης ενός προβλήματος. Στόχος της ανάλυσης, είναι η διάσπαση του προβλήματος σε άλλα απλούστερα προβλήματα για να είναι εύκολη η αντιμετώπισή τους.

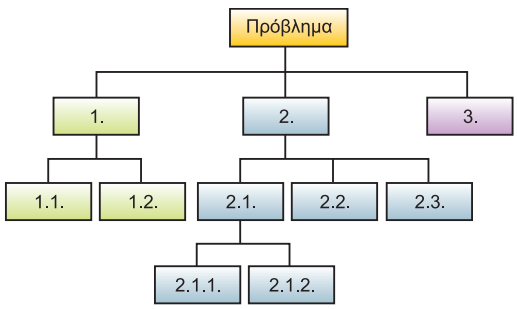
Η ανάλυση ενός προβλήματος μπορεί να πραγματοποιηθεί είτε φραστικά είτε διαγραμματικά, όπως φαίνεται στο παράδειγμα 2.1.

Παράδειγμα 2.1. Ανάλυση προβλήματος: Εξυπηρέτηση πολιτών από τις υπηρεσίες του δημοσίου.

Φραστική ανάλυση

1. Προσδιορισμός αναγκών
  1.1. Ταχύτερη εξυπηρέτηση πολιτών
  1.2. Περιορισμός μετακινήσεων
2. Δράση
  2.1. Ανάπτυξη ηλεκτρονικών υπηρεσιών εξυπηρέτησης
    2.1.1. Ποιες υπηρεσίες θα είναι διαθέσιμες;
    2.1.2. Με ποια διαδικασία θα γίνονται διαθέσιμες;
  2.2. Ενημέρωση πολιτών
  2.3. Ενημέρωση υπαλλήλων για να συνδράμουν το έργο
3. Εφαρμογή του σχεδίου.

Διαγραμματική ανάλυση



Εικόνα 2.5. Διαγραμματική αναπαράσταση ανάλυσης προβλήματος

Για τη σωστή επίλυση ενός προβλήματος είναι σημαντικός ο επακριβής προσδιορισμός των **δεδομένων** που παρέχει το πρόβλημα και η λεπτομερειακή καταγραφή των **ζητούμενων** που αναμένονται σαν αποτελέ-

σματα της επίλυσης του προβλήματος. Για να βρει κάποιος τα ζητούμενα χρειάζεται να επεξεργαστεί τα δεδομένα.

Επεξεργασία δεδομένων είναι η συστηματική εκτέλεση πράξεων σε δεδομένα.

Αφού ολοκληρωθεί η ανάλυση του προβλήματος ακολουθεί το στάδιο της σύνθεσης. Κατά τη **σύνθεση** επιχειρείται η κατασκευή μιας νέας δομής, με την οργάνωση των επιμέρους στοιχείων του προβλήματος. Επιπλέον, η **κατηγοριοποίηση** του προβλήματος είναι ένα εξίσου σημαντικό στάδιο, μέσω του οποίου το πρόβλημα κατατάσσεται σε κάποια κατηγορία, σε μία οικογένεια παρόμοιων προβλημάτων και έτσι διευκολύνεται η επίλυση, αφού παρέχεται η ευκαιρία να προσδιοριστεί το ζητούμενο ανάμεσα σε παρόμοια «αντικείμενα». Τέλος, με τη **γενίκευση**, μπορούν να μεταφερθούν τα αποτελέσματα σε άλλες παρεμφερείς καταστάσεις ή προβλήματα.

Παράδειγμα 2.2. Να διερευνηθεί η εξίσωση $\alpha x + \beta = 0$ ως προς x, για τις διάφορες τιμές του α και β.

Απάντηση

Υπάρχουν 2 περιπτώσεις: Αν ο συντελεστής της μεταβλητής x είναι διάφορος του μηδενός ($\alpha \neq 0$) ή αν ο συντελεστής της μεταβλητής x είναι ίσος με μηδέν ($\alpha = 0$).

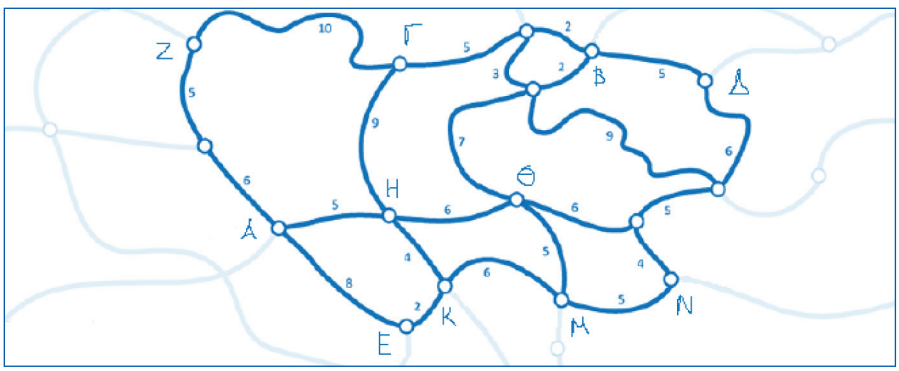
Περίπτωση 1: Αν $\alpha \neq 0$, τότε η εξίσωση έχει μοναδική λύση την $x = -\frac{\beta}{\alpha}$.

Περίπτωση 2: Αν $\alpha = 0$ τότε υπάρχουν δύο υποπεριπτώσεις: Αν ο σταθερός όρος β είναι διάφορος του μηδενός ($\beta \neq 0$) ή αν είναι ίσος με μηδέν ($\beta = 0$).

Περίπτωση 2.1. Αν $\beta \neq 0$, η εξίσωση είναι αδύνατη.

Περίπτωση 2.2. Αν $\beta = 0$, η εξίσωση είναι αόριστη.

Παράδειγμα 2.3. Δίνεται ο ακόλουθος χάρτης διαδρομών (εικόνα 2.6) που συνδέει ορισμένες πόλεις. Ο χάρτης δείχνει το χρόνο που απαιτείται για τη μετακίνηση από πόλη σε πόλη.



Εικόνα 2.6. Χάρτης διαδρομών



Δεδομένο είναι μια παράσταση γεγονότων, εννοιών ή εντολών σε τυποποιημένη μορφή που είναι κατάλληλη για επικοινωνία, ερμηνεία ή επεξεργασία από τον άνθρωπο ή από αυτόματα μέσα.

Με τον όρο **ζητούμενο** δηλώνεται οτιδήποτε προκύπτει ή τίθεται ως αντικείμενο έρευνας ή αναζήτησης.

Με τον όρο **πληροφορία** αναφέρεται οποιοδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων.

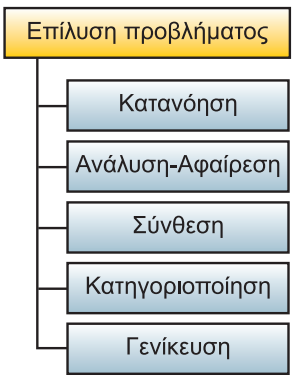
Κατανόηση: Δίνονται οι σταθεροί όροι α, β της εξίσωσης και ζητείται η τιμή της μεταβλητής x για τις διάφορες τιμές των α και β.

Ανάλυση: Το πρόβλημα διασπάται αρχικά σε δύο υποπροβλήματα. Στο πρώτο ο συντελεστής α της μεταβλητής x είναι διάφορος του μηδενός. Στο δεύτερο ο συντελεστής είναι μηδέν. Το δεύτερο υποπρόβλημα διασπάται σε δύο υποπροβλήματα: Ο σταθερός όρος β είναι ίσος με μηδέν ή είναι διάφορος του μηδενός.

Σύνθεση: Η εξίσωση είτε έχει μοναδική λύση, είτε είναι αδύνατη, είτε είναι αόριστη.

Κατηγοριοποίηση και γενίκευση: όλες οι πρωτοβάθμιες εξισώσεις αντιμετωπίζονται με αυτή την προσέγγιση.

Το παράδειγμα 2.3 εντάσσεται σε μία γενικότερη κατηγορία προβλημάτων εύρεσης συντομότερης διαδρομής.



Εικόνα 2.7. Στάδια επίλυσης προβλήματος.





Το 1976 το θεώρημα των 4 χρωμάτων αποδείχθηκε. Για την απόδειξη χρησιμοποιήθηκε ένας υπολογιστής για 1200 ώρες.



Λέξεις κλειδιά
Πρόβλημα, Επιλύσιμα, Ανοικτά, Υπολογιστικά, Επίλυση προβλήματος, Κατανόηση, Ανάλυση, Αφαίρεση, Σύνθεση, Κατηγοριοποίηση, Γενίκευση

1. Ποια διαδρομή είναι η συντομότερη από την πόλη Α στην πόλη Β;
2. Σε ποια πόλη θα συναντηθούν τρεις φίλοι ώστε κανένας να μην κινηθεί περισσότερο από δεκαπέντε λεπτά αν βρίσκονται στις πόλεις Γ, Δ και Ε αντίστοιχα και τα τρένα τους ξεκινούν όλα στις 19:00;

Απάντηση

1. Όπως φαίνεται στο χάρτη, υπάρχουν πολλές διαδρομές για να μεταβεί κάποιος από την πόλη Α στην πόλη Β. Από αυτές τις διαδρομές χρειάζεται να υπολογιστεί η συντομότερη με βάση το χρόνο που απαιτείται για τη μετακίνηση από πόλη σε πόλη. Για το σκοπό αυτό, μετριούνται οι αποστάσεις μεταξύ των πόλεων, από όπου προκύπτει ότι η συντομότερη με βάση το χρόνο διαδρομή είναι 20 λεπτά.
2. Με παρόμοιο τρόπο, μετριούνται οι σχετικές αποστάσεις. Η συνάντηση θα γίνει στην πόλη Θ, αφού ο άνθρωπος από την πόλη Γ θα κάνει 15 λεπτά, ο άνθρωπος από την πόλη Δ θα κάνει 14 λεπτά και ο άνθρωπος από την πόλη Ε θα κάνει 12 λεπτά.

Ανακεφαλαίωση

Στο κεφάλαιο αυτό παρουσιάστηκε η διαχρονικότητα του προβλήματος και επιχειρήθηκε να γίνει σαφής η ανεξαρτησία της λύσης του από τον υπολογιστή. Κατηγοριοποιήθηκαν τα προβλήματα ως προς τη δυνατότητα επίλυσης και επιπλέον ως προς τη δυνατότητα επίλυσης με τον υπολογιστή. Επισημάνθηκαν βασικά στοιχεία στη διαδικασία επίλυσης ενός προβλήματος (κατανόηση, καθορισμός δεδομένων και ζητουμένων, ανάλυση-αφαίρεση, σύνθεση, κατηγοριοποίηση και γενίκευση).

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Να αναφέρετε τις κατηγορίες προβλημάτων ως προς τη δυνατότητα επίλυσης και να δώσετε παραδείγματα προβλημάτων από κάθε κατηγορία.
2. Εργαστείτε σε ομάδες. Κάθε ομάδα θα θέτει ένα πρόβλημα στην άλλη ομάδα, η οποία καλείται να ανακαλύψει την κατηγορία στην οποία ανήκει το πρόβλημα.
3. Να διερευνήσετε την εξίσωση $\alpha x^2 + \beta x + \gamma = 0$ ως προς x για τις διάφορες τιμές των α, β και γ.
4. Μπορεί κάθε χάρτης να χρωματιστεί με τέσσερα χρώματα το πολύ, ώστε οι γειτονικές χώρες να είναι χρωματισμένες διαφορετικά;
5. Να χαρακτηρίσετε με Σωστό ή Λάθος τις παρακάτω προτάσεις:
Α. Κάθε επιλύσιμο πρόβλημα είναι υπολογιστικό.
Β. Η κατανόηση προηγείται της επίλυσης.
Γ. Όλα τα προβλήματα μπορούν να λυθούν με τη βοήθεια του υπολογιστή.
Δ. Η ανάλυση του προβλήματος βοηθάει στην επίλυσή του.
Ε. Υπάρχουν μη υπολογιστικά μαθηματικά προβλήματα.



Αλγόριθμοι

Στόχοι του κεφαλαίου αυτού είναι να μπορούν οι μαθητές:

- ✓ να περιγράφουν την έννοια του αλγορίθμου και να διακρίνουν την ύπαρξη συγκεκριμένων χαρακτηριστικών που χρειάζεται να έχει ένας αλγόριθμος
- ✓ να αναγνωρίζουν βασικές έννοιες στην Ανάλυση Αλγορίθμων
- ✓ να αναγνωρίζουν τις διάφορες μορφές αναπαράστασης αλγορίθμου
- ✓ να αναφέρουν τους βασικούς τύπους και δομές δεδομένων
- ✓ να διακρίνουν τις βασικές εντολές και δομές που χρησιμοποιούνται σε έναν αλγόριθμο
- ✓ να προσδιορίζουν τον τρόπο λειτουργίας των δομών δεδομένων
- ✓ να εκπονούν απλούς αλγορίθμους
- ✓ να εντοπίζουν και να διορθώνουν τα λογικά λάθη ενός αλγορίθμου
- ✓ να εξηγούν την ανάγκη δημιουργίας της κατάλληλης τεκμηρίωσης.

2.2.1 Ορισμός αλγορίθμου

Η λέξη αλγόριθμος (algorithm) προέρχεται από μια μελέτη του Πέρση μαθηματικού Μοχάμεντ Ιμπν Μουσά Αλ Χουαρίζμι (Muḥammad ibn Mūsā al-Khwārizmī), που έζησε περί το 825 μ.Χ. (Εικόνα 2.8). Παρόλα αυτά, η ύπαρξη και η ηλικία μερικών αλγορίθμων αριθμεί χιλιάδες χρόνια. Σήμερα, το πεδίο της μελέτης των αλγορίθμων (το οποίο καλείται θεωρία αλγορίθμων) είναι ένα ιδιαίτερα ευρύ πεδίο έρευνας. Γενικά,

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

Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής. Για παράδειγμα, το δέσιμο της γραβάτας αποτελεί ένα πρόβλημα, για την επίλυση του οποίου χρειάζεται να εκτελεστεί μια πεπερασμένη σειρά ενεργειών (Εικόνα 2.9.). Η αλληλουχία των ενεργειών οδηγεί στο επιθυμητό αποτέλεσμα. Η αλληλουχία δεν είναι απαραίτητα μοναδική για την επίτευξη αυτού του, αφού, υπάρχουν και άλλοι τρόποι για το δέσιμο της γραβάτας.



Προερωτήσεις

- Ξέρεις ότι ήδη έχεις χρησιμοποιήσει πολλούς αλγορίθμους;
- Μπορείς να περιγράψεις πώς βρίσκεις το Μέγιστο Κοινό Διαιρέτη δύο αριθμών;
- Τι κάνεις για να εντοπίσεις μία λέξη σε ένα λεξικό;



Εικόνα 2.8. Ο Αλ Χουαρίζμι

Η μελέτη του Αλ Χουαρίζμι μεταφράστηκε στα λατινικά και άρχιζε με τη φράση «Algoritmi dixit...» (ο αλγόριθμος λέει ....).



Εικόνα 2.9. Αλγόριθμος δεσίματος γραβάτας

Ο όρος αλγόριθμος επέζησε επί χίλια χρόνια ως σπάνιος όρος, που σήμαινε κάτι σαν «συστηματική διαδικασία αριθμητικών χειρισμών». Τη σημερινή του αξία απόκτησε στις αρχές του 20ού αιώνα με την ανάπτυξη της θεωρίας αλγορίθμων και φυσικά με τους υπολογιστές.



«Ο ευκλείδειος αλγόριθμος είναι ο παππούς όλων των αλγορίθμων, αφού είναι ο παλαιότερος μη τετριμμένος αλγόριθμος που χρησιμοποιείται ακόμα και σήμερα.» (Ντόναλντ Κνουθ, Donald Knuth, 1981)



Στον ευκλείδειο αλγόριθμο, δεν έχει σημασία αν θα είναι η τιμή του y μικρότερη από την τιμή του x. Έτσι, στην αρχική εκτέλεση του αλγορίθμου έχει επιλεγεί ο πρώτος αριθμός να είναι μεγαλύτερος του δευτέρου. Στη συνέχεια θα εκτελεστεί ο αλγόριθμος με τους ίδιους αριθμούς, όπου ο πρώτος αριθμός θα είναι μικρότερος του δευτέρου.

Απαραίτητο είναι οι τιμές των δύο μεταβλητών να είναι ακέραιες και μία εκ των δύο μεταβλητών να είναι διάφορη του μηδενός. Στην περίπτωση αρνητικών τιμών ο ΜΚΔ προκύπτει από τη μελέτη των απολύτων τιμών των δύο αριθμών.

Η εκτέλεση του ευκλείδειου αλγορίθμου, μπορεί να πραγματοποιηθεί από κάποιον που δεν χρειάζεται απαραίτητα να έχει κατασκευάσει τον αλγόριθμο. Επιπλέον, ο αλγόριθμος εφόσον κωδικοποιηθεί σε γλώσσα προγραμματισμού, μπορεί να εκτελεστεί από τον υπολογιστή.

Ιστορικά, ένας από τους πρώτους αλγορίθμους, είναι ο αλγόριθμος για την εύρεση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ) δύο ακεραίων αριθμών x και y.

Παράδειγμα 2.4. Να βρεθεί ο Μέγιστος Κοινός Διαιρέτης (ΜΚΔ) δύο θετικών ακεραίων αριθμών x και y.

Απάντηση

Ο αλγόριθμος περιγράφεται σε ομιλούμενη γλώσσα ως εξής: Θέσε στο z τον διαιρέτη. Αν $z = 0$, τότε ΜΚΔ είναι ο x. Αν $z \neq 0$ τότε διαίρεσε το x με το y, και έστω z το υπόλοιπο και επανάλαβε τη διαίρεση με τους ακέραιους y και z αντί για x και y μέχρι το z να γίνει 0.

Για παράδειγμα προκειμένου να βρεθεί ο ΜΚΔ δύο μη μηδενικών αριθμών, π.χ. των 78 και 27, σύμφωνα με τον αλγόριθμο μπορείτε να κάνετε τις ακόλουθες ενέργειες:

Βρείτε το υπόλοιπο της διαίρεσης του 78 με το 27. Το υπόλοιπο είναι 24. Ελέγξτε αν είναι 0. Στην περίπτωση αυτή δεν είναι 0. Αφού δεν είναι 0, βρείτε το υπόλοιπο της διαίρεσης του 27 με το 24. Το υπόλοιπο είναι 3. Ελέγξτε αν είναι 0. Στην περίπτωση αυτή δεν είναι 0. Αφού δεν είναι 0, βρείτε το υπόλοιπο της διαίρεσης του 24 με το 3. Το υπόλοιπο είναι 0. Αφού το υπόλοιπο είναι 0, βρέθηκε ο ΜΚΔ. Ο ΜΚΔ είναι 3.

Ο αλγόριθμος αυτός μπορεί να εκφραστεί και με κωδικοποιημένο τρόπο ως εξής:

```
1.  Αλγόριθμος Ευκλείδης
2.  Διάβασε x, y
3.  z ← y
4.  Όσο z ≠ 0 επανάλαβε
5.      z ← x mod y
6.      x ← y
7.      y ← z
8.  Τέλος_επανάληψης
9.  Εμφάνισε x
10. Τέλος Ευκλείδης
```

Για την περιγραφή του αλγορίθμου χρησιμοποιείται μια γλώσσα στην οποία το όνομα του αλγορίθμου (Ευκλείδης) καθορίζει την αρχή και το τέλος του.

Οι τιμές εισόδου (x, y) δίνονται με την εντολή Διάβασε, ενώ ο ΜΚΔ είναι η τιμή που παίρνει τελικά η μεταβλητή x, η οποία εμφανίζεται.

Η εύρεση του ΜΚΔ είναι μια επαναληπτική διαδικασία που συνεχίζεται όσο το υπόλοιπο (**mod**) της διαίρεσης x διά του y είναι διάφορο του μηδενός. Η επαναληπτική διαδικασία έχει την έννοια «όσο ισχύει η συνθήκη (δηλαδή όσο $z \neq 0$) επαναλάμβανε τη διαδικασία, αλλιώς μην εκτελείς άλλο τη διαδικασία και συνέχισε στο επόμενο βήμα». Οι εντολές του τύπου $x \leftarrow y$ δεν έχουν την έννοια της ισότητας, αλλά της εκχώρησης τιμής του δεξιού μέλους στη μεταβλητή του αριστερού μέλους. Αυτό σημαίνει ότι μετά την εκτέλεση της εντολής η μεταβλητή του αριστερού μέλους θα έχει τιμή ίση με αυτή του δεξιού μέλους.

Με βάση τα παραπάνω, ο ευκλείδειος αλγόριθμος για τον υπολογισμό του ΜΚΔ δύο θετικών ακεραίων αριθμών, περιγράφεται πλήρως με ακρίβεια και σαφήνεια. Συνεπώς, αν το ζητούμενο είναι να υπολογιστεί με τη χρήση του αλγόριθμου Ευκλείδης, ο ΜΚΔ των αριθμών 27 και 78, τότε θα μπορούσε να αξιοποιηθεί η αρίθμηση των γραμμών του αλγορίθμου και να πραγματοποιηθεί εκτέλεσή του στο χαρτί. Σε κάθε επανάληψη υπολογίζονται οι τιμές των x, y και z, οι οποίες παρουσιάζονται στον πίνακα 2.1. Με την ολοκλήρωση προκύπτει ότι ο ΜΚΔ των αριθμών 27 και 78 είναι ο αριθμός 3.

Πίνακας 2.1. Εικονική εκτέλεση του ευκλείδειου αλγορίθμου

Αριθμός εντολής	x	y	z	z ≠ 0	Έξοδος
2	27	78			
3			78		
4				Αληθής	
5			27		
6	78				
7		27			
4				Αληθής	
5			24		
6	27				
7		24			
4				Αληθής	
5			3		
6	24				
7		3			
4				Αληθής	
5			0		
6	3				
7		0			
4				Ψευδής	
9					3

Ο παραπάνω αλγόριθμος, μπορεί να απαντήσει όχι μόνο στη συγκεκριμένη ερώτηση, «να βρεθεί ο ΜΚΔ των 27 και 78», αλλά σε όλες τις παρόμοιες ερωτήσεις. Λύνει, δηλαδή, ένα πρόβλημα. Κάθε μία από τις ερωτήσεις αυτές λέγεται στιγμιότυπο του προβλήματος. Έτσι, η εύρεση του ΜΚΔ των 27 και 78 είναι ένα στιγμιότυπο του προβλήματος της εύρεσης του ΜΚΔ δύο θετικών ακεραίων. Δηλαδή, αν εκτελεστούν τα βήματα του αλγορίθμου, θα ολοκληρωθεί η διαδικασία έχοντας πάρει τη σωστή απάντηση για οποιοδήποτε ζευγάρι θετικών ακεραίων.

Ωστόσο, ένα θεωρητικό ερώτημα που προκύπτει είναι το ακόλουθο: «γιατί ο αλγόριθμος λύνει οποιοδήποτε στιγμιότυπο του προβλήμα-



Η εκτέλεση ενός αλγορίθμου πραγματοποιείται αφού αριθμηθούν οι γραμμές του αλγορίθμου. Για κάθε εντολή που εκτελείται, καταγράφετε τον αριθμό της γραμμής και το αποτέλεσμα της εκτέλεσης στο αντίστοιχο κελί.

Η αρίθμηση των γραμμών του αλγορίθμου είναι απαραίτητη μόνο για την εκτέλεσή του.

Οι εντολές της γραμμής 1 και 10 είναι δηλωτικές εντολές και δεν αποτυπώνονται στον πίνακα 2.1.



Ο Ευκλείδης έζησε περίπου από το 330 έως το 275 π.Χ. και έγραψε το έργο «Τα Στοιχεία» που αποτελείται από 13 βιβλία, τα οποία επιχειρούν να συστηματοποιήσουν τις μαθηματικές γνώσεις της εποχής του.



Ο ευκλείδειος αλγόριθμος έχει πολλές θεωρητικές και πρακτικές εφαρμογές. Μέσω αυτού μπορούν να δημιουργηθούν αρκετοί παραδοσιακοί μουσικοί ρυθμοί. Χρησιμοποιείται στην κρυπτογραφία, στο ηλεκτρονικό εμπόριο, στην επίλυση διοφαντικών εξισώσεων κ.α.



Χαρακτηριστικά ενός αλγορίθμου

1. Καθοριστικότητα (Definiteness)
2. Περατότητα (Finiteness)
3. Αποτελεσματικότητα (Effectiveness)
4. Είσοδος (Input)
5. Έξοδος (Output)



Όπως θα παρουσιαστεί σε επόμενη παράγραφο, η είσοδος σε ένα αλγόριθμο μπορεί να επιτευχθεί με την εντολή Διάβασε, την εντολή Δεδομένα και την εντολή εκχώρησης. Με την εντολή εκχώρησης δημιουργούνται δεδομένα μέσα στον αλγόριθμο (κενό σύνολο μεταβλητών εισόδου).

τος;» Συνήθως, για να λύνει πραγματικά ο αλγόριθμος ένα πρόβλημα, χρειάζεται να μπορεί να αποδειχτεί η ορθότητά του με αυστηρό τρόπο. Στην περίπτωση του ευκλείδειου αλγορίθμου, αποδεικνύεται από τον ίδιο τον Ευκλείδη στο έβδομο βιβλίο των «Στοιχείων» του.

2.2.2 Χαρακτηριστικά αλγορίθμου

Κάθε αλγόριθμος είναι σημαντικό να έχει ορισμένα χαρακτηριστικά προκείμενου να θεωρείται πλήρης.

Καθοριστικότητα: Κάθε εντολή ενός αλγορίθμου χρειάζεται να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της.

Κατά τη διαίρεση δύο ακεραίων αριθμών, το χαρακτηριστικό της καθοριστικότητας ικανοποιείται αν έχει ληφθεί υπόψη και η περίπτωση που ο διαιρέτης είναι μηδέν.

Περατότητα: Κάθε αλγόριθμος πρέπει να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του.

Ένας αλγόριθμος για να διαθέτει το χαρακτηριστικό της περατότητας, χρειάζεται να προσδιορίζει τη λύση ενός προβλήματος μετά από ένα συγκεκριμένο αριθμό βημάτων και να μην εκτελείται ατέρμονα (δηλαδή χωρίς να τελειώνει).

Αποτελεσματικότητα: Κάθε εντολή ενός αλγορίθμου χρειάζεται να είναι διατυπωμένη απλά και κατανοητά, ώστε να μπορεί να εκτελεστεί επακριβώς και σε πεπερασμένο μήκος χρόνου.

Κατά τη διαίρεση δύο ακέραιων αριθμών, ο αλγόριθμος διαθέτει το χαρακτηριστικό της αποτελεσματικότητας, αφού οι ακέραιοι αναπαρίστανται ακριβώς και υπάρχει αλγόριθμος για τη διαίρεσή τους (Ευκλείδεια Διαίρεση) σε πεπερασμένο χρόνο. Αν όμως επιχειρείται η διαίρεση δύο πραγματικών αριθμών που ο καθένας αναπαρίσταται από άπειρα δεκαδικά ψηφία, τότε ο αλγόριθμος δεν διαθέτει το χαρακτηριστικό της αποτελεσματικότητας, αφού δεν μπορεί να αναπαρασταθεί πλήρως και να εκτελεστεί ακριβώς η συγκεκριμένη διαίρεση.

Είσοδος: Κάθε αλγόριθμος χρειάζεται να δέχεται ένα σύνολο μεταβλητών εισόδου (που μπορεί να είναι και το κενό σύνολο), οι οποίες αποτελούν τα δεδομένα του αλγορίθμου.

Η είσοδος των μεταβλητών μπορεί να επιτευχθεί με κατάλληλες εντολές, οι οποίες θα παρουσιαστούν σε επόμενες παραγράφους. Στην περίπτωση του αλγόριθμου Ευκλείδης που παρουσιάστηκε στο παράδειγμα 2.4, αυτό επιτυγχάνεται με την εντολή **Διάβασε** x, y.

Έξοδος: Κάθε αλγόριθμος χρειάζεται να δημιουργεί κάποιο αποτέλεσμα.

Το αποτέλεσμα του αλγορίθμου, η έξοδός του, είναι μία ή περισσότερες μεταβλητές ή/και σταθερές τιμές. Η έξοδος μπορεί να επιτευχθεί με κατάλληλες εντολές, οι οποίες θα παρουσιαστούν σε επόμενες παραγράφους. Στην περίπτωση του αλγόριθμου Ευκλείδης που παρουσιάστηκε στο παράδειγμα 2.4, αυτό επιτυγχάνεται με την εντολή **Εμφάνισε** x.

2.2.3 Ανάλυση Αλγορίθμων, Θεωρία Υπολογισμού, Πολυπλοκότητα Αλγορίθμων, Υπολογισιμότητα Αλγορίθμων.



Η πιο συνηθισμένη ανάλυση ενός αλγορίθμου αφορά το χρόνο εκτέλεσης. Ο χρόνος συνήθως υπολογίζεται σαν συνάρτηση του αριθμού των στοιχειωδών βημάτων που εκτελούνται στον αλγόριθμο.

Η ανάλυση της συμπεριφοράς ενός αλγορίθμου για συνθήκες παρόμοιες με αυτές που εμφανίζονται στην πράξη και η ικανοποιητική απόδοσή του, παρέχει τη δυνατότητα να πραγματοποιηθεί η υλοποίηση και η εφαρμογή του. Αν δεν επιτυγχάνεται ικανοποιητική απόδοση τότε απαιτείται ο σχεδιασμός ενός τροποποιημένου αλγορίθμου.

Η Θεωρία Υπολογισμού (Theory of computation) είναι το πεδίο της πληροφορικής που ασχολείται τόσο με το πρόβλημα ύπαρξης λύσης ενός προβλήματος όσο και αποδοτικότητας των αλγορίθμων για την επίλυση των προβλημάτων με βάση ένα δεδομένο μοντέλο υπολογισμού.



Η Θεωρία Υπολογισιμότητας ερευνά αν και πόσο αποδοτικά κάποια προβλήματα μπορούν να επιλυθούν με συγκεκριμένα υπολογιστικά μοντέλα.

Το πεδίο της θεωρίας υπολογισμού, διαιρείται σε δύο κλάδους: τη Θεωρία Υπολογισιμότητας (Computability Theory) και τη Θεωρία Πολυπλοκότητας (Computational Complexity Theory). Συνεπώς, για κάθε αλγόριθμο απαιτείται ανάλυση, μέσω της οποίας:

α) τεκμηριώνεται η ορθότητά του, δηλαδή αν ο αλγόριθμος κάνει πραγματικά τη δουλειά για την οποία έχει σχεδιαστεί και
β) μετριέται ποσοτικά η απόδοσή του, σε σχέση με διάφορα είδη υπολογιστικών πόρων, όπως είναι ο χρόνος και η μνήμη που απαιτείται για την εκτέλεσή του.



Η Θεωρία Πολυπλοκότητας μελετά τους πόρους που απαιτούνται για την επίλυση ενός προβλήματος βάσει ενός συγκεκριμένου αλγορίθμου.

Η ανάλυση ενός αλγορίθμου είναι η εκτίμηση του πλήθους των υπολογιστικών πόρων που απαιτεί η εκτέλεση του αλγορίθμου.

Για παράδειγμα, ο αλγόριθμος Ευκλείδης χρησιμοποιεί τόση μνήμη όση χρειάζεται για να αποθηκευτούν οι τρεις ακέραιοι x, y, z. Ωστόσο, η εύρεση του πλήθους των εντολών που εκτελούνται δεν είναι τόσο τετριμμένη. Κάθε φορά που πραγματοποιείται η επανάληψη εκτελούνται τέσσερα βήματα (έλεγχος της συνθήκης Όσο...επανάλαβε, υπολογισμός του z και δύο εντολές εκχώρησης). Επομένως, απαιτούνται 4×α βήματα, όπου α είναι ο αριθμός των επαναλήψεων. Πόσες, όμως, θα είναι οι επαναλήψεις;

Το ότι για x = 27 και y = 78 γίνονται τέσσερις επαναλήψεις, δεν είναι πολύ διαφωτιστικό. Θα ήταν χρήσιμο να μπορεί να υπολογιστεί ο αριθμός των επαναλήψεων για κάθε ζευγάρι ακεραίων x, y.

Έχει υπολογιστεί, ότι ο αριθμός των επαναλήψεων είναι περίπου ίσος με τον λογάριθμο του x (ή του y). Δηλαδή, ο ευκλείδειος αλγόριθμος για να υπολογίσει το ΜΚΔ των x και y, κάνει 4×logx βήματα. Συνήθως παραλείπονται σταθερές, όπως το 4 και έτσι προκύπτει ότι η πολυπλοκότητα του αλγόριθμου είναι O(logx) (διαβάζεται «η πολυπλοκότητα του αλγορίθμου είναι της τάξης logx»).

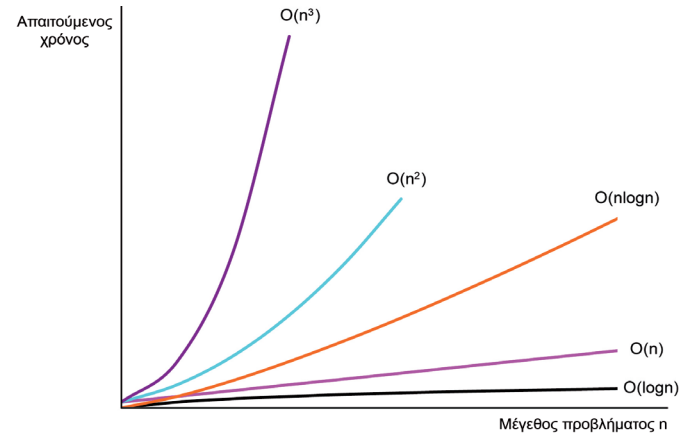
Η **πολυπλοκότητα ενός αλγορίθμου** δίνει ένα μέτρο της χρονικής καθυστέρησης του αλγορίθμου για την επίλυση ενός προβλήματος.

Οι περισσότεροι αλγόριθμοι έχουν πολυπλοκότητα χρόνου που ανήκει σε μια από τις κατηγορίες που φαίνονται στον Πίνακα 2.2. Με n συμβολίζεται το μέγεθος του προβλήματος και εξαρτάται από το πρόβλημα. Για παράδειγμα, αν ζητείται να ταξινομηθούν k αριθμοί, τότε το μέγεθος του προβλήματος είναι k, επομένως n = k.

Πίνακας 2.2: Κατηγορίες πολυπλοκότητας αλγορίθμων

Πολυπλοκότητα	Ονομασία	Παρατηρήσεις
O(1)	Σταθερή	Κάθε βήμα εκτελείται μια φορά ή το πολύ μερικές φορές
O(logn)	Λογαριθμική	Αν δεν σημειώνεται αλλιώς, οι λογάριθμοι είναι δυαδικοί (Δυαδική Αναζήτηση)
O(n)	Γραμμική	Σειριακή αναζήτηση
O(nlogn)		Γρήγορη Ταξινόμηση
$O(n^2)$	Τετραγωνική	Ταξινόμηση με επιλογή
$O(n^3)$	Κυβική	Πολλαπλασιασμός πινάκων
$O(2^n)$	Εκθετική	Μερισμός συνόλου

Στην εικόνα 2.10 έχουν παρασταθεί οι καμπύλες των κυριότερων συναρτήσεων πολυπλοκότητας, ενώ στον Πίνακα 2.3 παρουσιάζονται οι χρόνοι εκτέλεσης για διάφορες πολυπλοκότητες και μεγέθη προβλημάτων.



Εικόνα 2.10: Καμπύλες των κυριότερων συναρτήσεων πολυπλοκότητας.



Ο Lame απέδειξε ότι ο αριθμός των βημάτων του ευκλείδειου αλγορίθμου για δύο αριθμούς είναι πάντα μικρότερος ή ίσος του πενταπλασίου των ψηφίων του μεγαλύτερου από τους δύο αριθμούς.
$s \geq 4{,}785 \times \log_{10} n + 1{,}6723$



Ο συμβολισμός O (O-notation), όπου το O είναι το αρχικό γράμμα της αγγλικής λέξης order και διαβάζεται «όμικρον κεφαλαίο», χρησιμοποιείται για να αναδείξει τη γενική συμπεριφορά (την τάξη) του αλγορίθμου.



Υπολογισιμότητα:
Τι μπορεί να υπολογιστεί;
Μπορεί ένας υπολογιστής να λύσει οποιοδήποτε πρόβλημα δοθέντος αρκετού χρόνου και χώρου;

Πολυπλοκότητα
Πόσο γρήγορα μπορεί να λυθεί ένα πρόβλημα;
Πόσος χώρος (μνήμη) χρειάζεται για να λυθεί ένα πρόβλημα;



Γενικά, τα δεδομένα συνιστούν το μέγεθος της εισόδου ενός αλγορίθμου. Για παράδειγμα στην ταξινόμηση, το πλήθος των αντικειμένων που θα ταξινομηθούν δίνει το μέγεθος του προβλήματος.

Πίνακας 2.3. Χρόνοι εκτέλεσης για διάφορες πολυπλοκότητες και μεγέθη προβλημάτων			
Τάξη αλγορίθμου	**Μέγεθος Προβλήματος**		
	4	**16**	**64**
O(logn)	6×10^{-10} sec	$1{,}6\times10^{-8}$	$1{,}8\times10^{-8}$
O(nlogn)	2×10^{-8} sec	19×10^{-8}	$1{,}2\times10^{-6}$
O(n)	4×10^{-8} sec	16×10^{-8}	64×10^{-8}
$O(n^2)$	2×10^{-8} sec	$2{,}6\times10^{-6}$	4×10^{-5}
$O(n^3)$	64×10^{-8} sec	4×10^{-5}	3×10^{-3}
$O(2^n)$	16×10^{-8} sec	65×10^{-5}	5×10^{3} χρόνια
O(n!)	24×10^{-8} sec	58 ώρες	4×10^{73} χρόνια

Μετά την επίλυση ενός προβλήματος με ένα αλγόριθμο γίνεται προσπάθεια να σχεδιαστεί ένας νέος αλγόριθμος με μεγαλύτερη ταχύτητα εύρεσης της λύσης του προβλήματος. Για παράδειγμα ένας αλγόριθμος ταξινόμησης με συγκρίσεις (ο οποίος περιγράφεται σε επόμενο κεφάλαιο) απαιτεί $O(n^2)$ συγκρίσεις. Υπάρχουν όμως ταχύτεροι αλγόριθμοι, όπως ο αλγόριθμος γρήγορης ταξινόμησης (Quicksort), που απαιτεί O(nlogn) συγκρίσεις.

Η μελέτη αυτού του ζητήματος γίνεται στο πλαίσιο της ***Θεωρίας Πολυπλοκότητας***. Η έρευνα στην περιοχή αυτή προσπαθεί να βρει τους εγγενείς περιορισμούς στην ταχύτητα των αλγορίθμων για τη λύση συγκεκριμένων προβλημάτων. Έτσι αποδεικνύεται, για παράδειγμα, ότι δεν είναι δυνατόν να ταξινομηθούν n ακέραιοι με λιγότερο από nlogn συγκρίσεις και συνεπώς ο αλγόριθμος γρήγορης ταξινόμησης είναι ουσιαστικά βέλτιστος, όσον αφορά την ταχύτητα ταξινόμησης.

Η συνάρτηση O καλείται πολυωνυμική αν είναι έκφραση πολυωνύμου ως προς n, διαφορετικά καλείται μη-πολυωνυμική, αν περιέχει όρους όπως 2^n, n! κτλ. Στην πράξη οι τρεις τελευταίες πολυπλοκότητες χρησιμοποιούνται μόνο για προβλήματα μικρού μεγέθους.

Για το πρόβλημα της ταξινόμησης ενός πίνακα δεν έχει βρεθεί μέχρι σήμερα ταχύτερος αλγόριθμος από τον αλγόριθμο γρήγορης ταξινόμησης, ενώ για το πρόβλημα του υπολογισμού του ΜΚΔ δεν έχει βρεθεί ταχύτερος αλγόριθμος από εκείνον του Ευκλείδη. Αυτό οδηγεί, μοιραία, στο ερώτημα: Μήπως δεν υπάρχει καλύτερος αλγόριθμος, μήπως δηλαδή τα προβλήματα αυτά έχουν μία εγγενή πολυπλοκότητα;

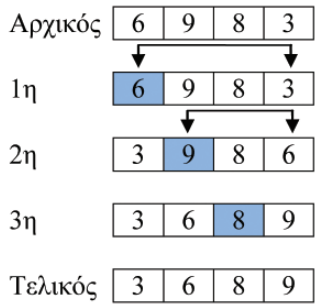
2.2.4 Βασικοί τύποι αλγορίθμων

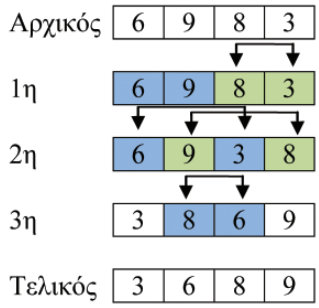
Ο ορισμός του αλγορίθμου που δόθηκε στην αρχή αυτού του κεφαλαίου, συμφωνεί με τη φιλοσοφία των περισσότερων υπολογιστών σήμερα, που διαθέτουν μία Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) στην οποία οι εντολές εκτελούνται με σειρά, η μία μετά την άλλη. Για το λόγο αυτό ονομάζονται ***σειριακοί*** αλγόριθμοι. Όμως η ύπαρξη προβλημάτων στα οποία απαιτείται πολύ μεγάλος χρόνος για τον υπολογισμό της λύσης ενός προβλήματος, δημιούργησε την ανάγκη εύρεσης αλγορίθμων, όπου ορισμένα ή μία σειρά από βήματα αυτών των αλγορίθμων θα μπορούσαν να εκτελούνται παράλληλα (ταυτόχρονα). Σε αυτή την περίπτωση, η εκτέλεση του ενός βήματος δεν εξαρτάται από την ολοκλήρωση της εκτέλεσης του προηγούμενου. Αλγόριθμοι αυτής της μορφής ονομάζονται ***παράλληλοι*** αλγόριθμοι και η υλοποίησή τους γίνεται με την ύπαρξη πολλαπλών ΚΜΕ στο σύστημα του υπολογιστή.

Σειριακοί λέγονται οι αλγόριθμοι που χρησιμοποιούν μία κεντρική μονάδα επεξεργασίας και οι εντολές τους εκτελούνται σε σειρά η μία μετά την άλλη.

Παράλληλοι χαρακτηρίζονται οι αλγόριθμοι που χρησιμοποιούν πολλαπλές κεντρικές μονάδες επεξεργασίας όπου ορισμένες ή μία σειρά από εντολές εκτελούνται παράλληλα (ταυτόχρονα).

Ο πίνακας του παραδείγματος 2.5 είναι μονοδιάστατος πίνακας που έχει τέσσερα στοιχεία, στις θέσεις 1, 2, 3, 4. Στη θέση 1 το περιεχόμενο του πίνακα είναι 6, στη θέση 2 το περιεχόμενο του πίνακα είναι 9 κτλ.







Δεν μπορούν να λυθούν όλα τα προβλήματα κάνοντας χρήση παράλληλου υπολογισμού. Το σε βάθος σκάψιμο για να ανοιχτεί ένα χαντάκι δεν μπορεί να υλοποιηθεί με παράλληλο αλγόριθμο, αφού το δεύτερο μέτρο δεν είναι προσβάσιμο αν δεν τελειώσει πρώτα το πρώτο μέτρο.

Παράδειγμα 2.5. Έστω ότι υπάρχει ένας πίνακας που έχει ως περιεχόμενο τους αριθμούς:

1	2	3	4
6	9	8	3

Στόχος είναι να τοποθετηθούν οι αριθμοί σε αύξουσα σειρά από το μικρότερο στο μεγαλύτερο (αύξουσα ταξινόμηση). Η διαδικασία ταξινόμησης θα επιχειρηθεί με σειριακή και παράλληλη επεξεργασία.

Απάντηση

➢ **Σειριακά**

Εντοπίζεται το μικρότερο στοιχείο του πίνακα (στην περίπτωση αυτή είναι το 3) και αντιμετατίθεται με το στοιχείο της πρώτης θέσης.

Εντοπίζεται το μικρότερο από τα υπόλοιπα στοιχεία του πίνακα (που είναι το 6) και αντιμετατίθεται με το στοιχείο της δεύτερης θέσης.

Εντοπίζεται το μικρότερο από τα υπόλοιπα στοιχεία του πίνακα (που είναι το 8), το οποίο όμως είναι το τρίτο στοιχείο του πίνακα, οπότε η ταξινόμηση έχει ολοκληρωθεί.

➢ **Παράλληλα**

Συγκρίνονται ταυτόχρονα με δύο διαφορετικούς επεξεργαστές το 1ο με το 2ο στοιχείο και το 3ο με το 4ο. Αν δεν είναι σωστά διαταγμένα, αντιμετατίθενται.

Συγκρίνονται ταυτόχρονα με δύο διαφορετικούς επεξεργαστές το 1ο με το 3ο στοιχείο και το 2ο με το 4ο. Αν δεν είναι σωστά διαταγμένα, αντιμετατίθενται.

Τώρα το μικρότερο από όλα τα στοιχεία είναι στη σωστή θέση (την 1η) και το μεγαλύτερο επίσης στη σωστή θέση (την 4η). Ωστόσο τα δύο μεσαία στοιχεία δεν είναι βέβαιο ότι είναι σωστά διαταγμένα. Οπότε απαιτείται μια ακόμη σύγκριση αυτών των δύο από έναν επεξεργαστή και ολοκληρώνεται η ταξινόμηση.

Οι αλγόριθμοι επιλύουν προβλήματα. Υπάρχουν απλά και σύνθετα προβλήματα. Λίγα απλά προβλήματα μπορούν να επιλυθούν με διαδοχική εκτέλεση μερικών βημάτων, αφού τα περισσότερα προβλήματα απαιτούν την εκτέλεση ορισμένων συγκεκριμένων βημάτων πολλές φορές. Αυτοί οι αλγόριθμοι αποκαλούνται ***επαναληπτικοί***.

Παράδειγμα 2.6. Να αναπτυχθεί αλγόριθμος ο οποίος θα διαβάζει τον αριθμό Ν και θα υπολογίζει και θα εμφανίζει το Ν παραγοντικό (συμβολισμός: Ν!).

Το Ν! ορίζεται ως το γινόμενο των ακέραιων αριθμών 1, 2 έως Ν. Δηλαδή
Ν!=1·2·3·….· (Ν-1) ·Ν
Αν Ν = 5, το 5! = 1·2·3·4·5 = 120

Ο αλγόριθμος υπολογισμού του N! με μια επανάληψη βρίσκει το γινόμενο (βλέπε παράδειγμα 2.20).

Όμως το N! μπορεί να οριστεί και με άλλο τρόπο, που αποκαλείται αναδρομικός, ως εξής:

$$\begin{cases} N! = N\cdot(N-1)! \text{ για } N \geq 1 & (1) \\ 0! = 1 & (2) \end{cases}$$

Από τη σχέση (1) φαίνεται ότι το παραγοντικό του N ορίζεται χρησιμοποιώντας το παραγοντικό του (N - 1). Ο όρος αναδρομικότητα εδώ εκφράζει, ότι για να βρεθεί η τιμή του N! πρέπει να βρεθεί η τιμή του (N - 1)!, η τιμή του οποίου χρειάζεται την τιμή του (N - 2)! κ.ο.κ.

Έτσι το 5! κάνει διαδοχικά:

$$5! = 4!\cdot 5 = 3!\cdot 4\cdot 5 = 2!\cdot 3\cdot 4\cdot 5 = 1!\cdot 2\cdot 3\cdot 4\cdot 5 = 0!\cdot 1\cdot 2\cdot 3\cdot 4\cdot 5 = 1\cdot 1\cdot 2\cdot 3\cdot 4\cdot 5 = 120$$

Φαίνεται ότι ο υπολογισμός του N! με αναδρομικό τρόπο είναι πιο πολύπλοκος από τον επαναληπτικό. Ωστόσο σε άλλες περιπτώσεις και ιδίως σε μερικά δύσκολα προβλήματα η αναδρομή διευκολύνει σημαντικά.

2.2.5 Αναπαράσταση αλγορίθμου

Προκειμένου να επιτευχθεί η «ακριβής περιγραφή» ενός αλγορίθμου, χρησιμοποιείται κάποια γλώσσα που μπορεί να περιγράφει σειρές ενεργειών με τρόπο αυστηρό, χωρίς ασάφειες και διφορούμενα. Τέτοιες γλώσσες είναι οι γλώσσες προγραμματισμού (με την προϋπόθεση ότι η σημασιολογία τους είναι αυστηρά διατυπωμένη), μαθηματικά μοντέλα, κάποιες συμβολικές γλώσσες που χρησιμοποιούν αυστηρά καθορισμένους κανόνες περιγραφής, καθώς και κατάλληλα διαμορφωμένα υποσύνολα των φυσικών (ομιλούμενων) γλωσσών.

Η αναπαράσταση των αλγορίθμων μπορεί να πραγματοποιηθεί με:

- **Φυσική γλώσσα** όπου η αναπαράσταση γίνεται με την ομιλούμενη γλώσσα, μέσω της οποίας περιγράφονται τα βήματα επίλυσης του προβλήματος. Ωστόσο, με τη φυσική γλώσσα μπορούν να παρατηρηθούν ασάφειες στις οδηγίες.
- **Ψευδοκώδικα ή ψευδογλώσσα** η οποία είναι μια υποθετική γλώσσα για την αναπαράσταση αλγορίθμων με στοιχεία από κάποιες γλώσσες προγραμματισμού, παραλείποντας λεπτομέρειες που δεν είναι ουσιαστικές για την ανθρώπινη κατανόηση του αλγορίθμου.
- **Γλώσσα προγραμματισμού** η οποία είναι μια τεχνητή γλώσσα, που έχει αναπτυχθεί για να δημιουργεί ή να εκφράζει προγράμματα για τον υπολογιστή. Η αναπαράσταση των αλγορίθμων με γλώσσα προγραμματισμού μπορεί να γίνει είτε με οπτικές είτε με κειμενικές γλώσσες προγραμματισμού.



Ενδιαφέρον ζήτημα αποτελεί ο εντοπισμός του καλύτερου τρόπου υποδιαίρεσης των προβλημάτων, για να είναι εφικτή η επεξεργασία τους από πολλούς επεξεργαστές παράλληλα.



Για παράδειγμα, επαναληπτικοί αλγόριθμοι είναι ο ευκλείδειος αλγόριθμος, καθώς και ο αλγόριθμος ταξινόμησης με επιλογή που περιγράφηκε πιο πριν.



Αλγόριθμοι που υλοποιούν μια αναδρομική σχέση, αποκαλούνται ***αναδρομικοί*** και μερικά παραδείγματα παρουσιάζονται στην παράγραφο 2.2.7.6.



Όταν περιγράφεται στην ομιλούμενη γλώσσα ο τρόπος με τον οποίο θα μπορέσει κάποιος να επισκεφθεί ένα μουσείο, τότε ο αλγόριθμος έχει διατυπωθεί με φυσική γλώσσα.

Οι φυσικές γλώσσες, είναι οι γλώσσες που μιλούν οι άνθρωποι, ενώ οι τεχνητές γλώσσες έχουν αναπτυχθεί κυρίως για διευκόλυνση της επικοινωνίας ιδεών.

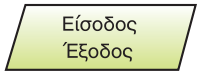
Θέματα Θεωρητικής Επιστήμης των Υπολογιστών

Τα κυριότερα χρησιμοποιούμενα γεωμετρικά σχήματα - σύμβολα στα διαγράμματα ροής είναι τα ακόλουθα:

- Η έλλειψη, που δηλώνει την αρχή και το τέλος του αλγορίθμου.



- Το πλάγιο παραλληλόγραμμο, που δηλώνει είσοδο ή έξοδο στοιχείων.



- Το ορθογώνιο παραλληλόγραμμο, που δηλώνει την εκτέλεση μιας ή περισσότερων πράξεων.



- Ο ρόμβος, που δηλώνει μία ερώτηση με δύο εξόδους για απάντηση.



- Στα διαγράμματα ροής, εκτός των παραπάνω σχημάτων, χρησιμοποιείται και το βέλος, το οποίο δείχνει τη ροή εκτέλεσης του αλγορίθμου.



Ο αλγόριθμος Αντιμετάθεση αντιμεταθέτει το περιεχόμενο των δύο μεταβλητών και παρουσιάζεται με τρεις τρόπους αναπαράστασης.

Στην ψευδογλώσσα έχουν καταγραφεί οι ενέργειες που περιγράφονται στο πλαίσιο της φυσικής γλώσσας σε κωδικοποιημένη μορφή, ενώ τέλος έχει δημιουργηθεί και το αντίστοιχο διάγραμμα ροής.

- Στις **οπτικές γλώσσες προγραμματισμού**, η αναπαράσταση των αλγορίθμων γίνεται μέσα από το γραφικό χειρισμό προγραμματιστικών στοιχείων.
- Στις **κειμενικές γλώσσες προγραμματισμού**, η αναπαράσταση των αλγορίθμων γίνεται με τη χρήση σειρών κειμένου που περιλαμβάνουν λέξεις, αριθμούς και σημεία στίξης.

- **Μεθοδολογίες διαγραμματικής αναπαράστασης αλγορίθμων** που συνιστούν έναν γραφικό τρόπο παρουσίασης του αλγόριθμου. Από τις διάφορες μεθοδολογίες διαγραμματικής αναπαράστασης αλγορίθμων που έχουν επινοηθεί η πιο διαδεδομένη είναι το διάγραμμα ροής, όπου η περιγραφή και η αναπαράσταση των αλγορίθμων γίνεται με τη χρήση γεωμετρικών σχημάτων - συμβόλων, όπου το καθένα δηλώνει μια συγκεκριμένη ενέργεια ή λειτουργία.

Παράδειγμα 2.7. Να αναπτυχθεί αλγόριθμος με φυσική γλώσσα, με διάγραμμα ροής και με ψευδογλώσσα, ο οποίος θα διαβάζει τις τιμές δύο μεταβλητών και θα αντιμεταθέτει το περιεχόμενό τους. Στη συνέχεια θα εμφανίζει ως αποτέλεσμα το περιεχόμενο των μεταβλητών μετά την αντιμετάθεση.

Να εκτελεστεί ο αλγόριθμος για τις τιμές 8 και 12.

Απάντηση

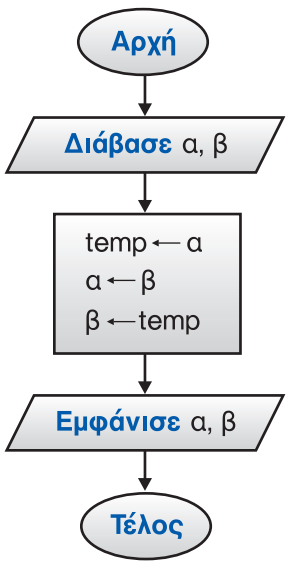
Φυσική γλώσσα: Αφού εισαχθούν οι τιμές δύο μεταβλητών α και β, να δώσετε το περιεχόμενο της μεταβλητής α και σε μία νέα μεταβλητή temp (προσωρινή). Στη συνέχεια, να δώσετε το περιεχόμενο της μεταβλητής β στη μεταβλητή α και τέλος να δώσετε το περιεχόμενο της μεταβλητής temp και στη μεταβλητή β.

Ψευδογλώσσα

1. **Αλγόριθμος** Αντιμετάθεση
2. **Διάβασε** α, β
3. temp ← α
4. α ← β
5. β ← temp
6. **Εμφάνισε** α, β
7. **Τέλος** Αντιμετάθεση

Αρ. Εντ.	α	β	temp	Έξοδος
2	8	12		
3			8	
4	12			
5		8		
6				12 8

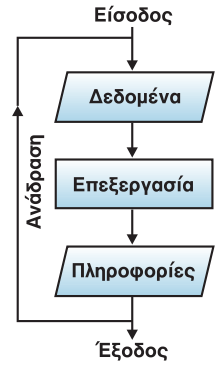
Διάγραμμα ροής



2.2.6 Δεδομένα και αναπαράστασή τους

Ένας αλγόριθμος λαμβάνει κάποια δεδομένα από την είσοδο, τα επεξεργάζεται μέσα από μια σειρά βημάτων και δίνει ως έξοδο τα αποτελέσματα.

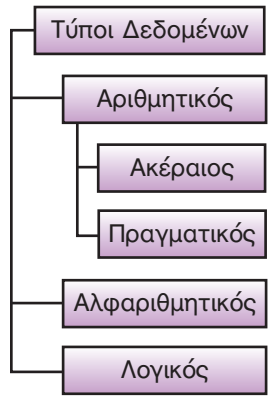
Η επεξεργασία δεδομένων, η οποία στην πράξη πραγματοποιείται μέσω αλγορίθμων, αναφέρεται στην εκτέλεση διαφόρων πράξεων/ λειτουργιών πάνω στα δεδομένα. Το αποτέλεσμα της επεξεργασίας δεδομένων είναι η πληροφορία. Έτσι, θα μπορούσε κανείς γενικεύοντας να πει ότι ένας αλγόριθμος μετατρέπει τα δεδομένα σε πληροφορία. Για παράδειγμα, ένας τηλεφωνικός αριθμός και ένα ονοματεπώνυμο αποτελούν δεδομένα. Δεν παρέχουν όμως καμία πληροφορία. Πληροφορία παράγεται μόνο αν σχετισθούν μεταξύ τους, ότι δηλαδή κάποιο τηλέφωνο ανήκει σε κάποιο συγκεκριμένο συνδρομητή. Με βάση τις πληροφορίες λαμβάνονται αποφάσεις και γίνονται διάφορες ενέργειες. Στη συνέχεια οι ενέργειες αυτές παράγουν νέα δεδομένα, αυτά νέες πληροφορίες, οι τελευταίες νέες αποφάσεις και ενέργειες κ.ο.κ. όπως φαίνεται στην εικόνα 2.11. Η διεργασία αυτή μπορεί να επαναλαμβάνεται, οι πληροφορίες που παρήχθησαν, να επανατροφοδοτούν μέσω ανάδρασης την είσοδο για επανάληψη του κύκλου κ.ο.κ..



Εικόνα 2.11. Σχέση δεδομένων και πληροφοριών

Η θεωρία αλγορίθμων, μελετά τα δεδομένα κυρίως από τις σκοπιές του υλικού και των γλωσσών προγραμματισμού.

Όσον αφορά τις γλώσσες προγραμματισμού, κάθε γλώσσα μπορεί να υποστηρίζει τη χρήση διαφόρων τύπων δεδομένων. Κάθε γλώσσα έχει συγκεκριμένους τύπους δεδομένων, ενώ μπορούν να δημιουργηθούν νέοι τύποι ορισμένοι από το χρήστη. Οι πιο συνήθεις τύποι δεδομένων είναι οι ακόλουθοι (εικόνα 2.12):



Εικόνα 2.12. Τύποι Δεδομένων

Ακέραιος τύπος: για την αναπαράσταση ακεραίων αριθμών.
Πραγματικός τύπος: για την αναπαράσταση πραγματικών αριθμών.
Λογικός τύπος: για την αναπαράσταση λογικών δεδομένων.
Αλφαριθμητικός τύπος: για την αναπαράσταση αλφαριθμητικών δεδομένων.

Σε κάθε τύπο δεδομένων μπορούν να εφαρμοστούν διαφορετικές πράξεις. Επομένως, κατά τον σχεδιασμό ενός αλγορίθμου έχει σημασία το είδος των τύπων δεδομένων που υποστηρίζονται.

Το υλικό επιτρέπει την αποθήκευση των δεδομένων ενός προγράμματος στην κύρια μνήμη ή και στις περιφερειακές συσκευές ενός υπολογιστή με διάφορες μορφές. Με σκοπό την πρόσβαση στα δεδομένα που βρίσκονται στη βοηθητική μνήμη είναι πιθανό να χρησιμοποιούνται διαφορετικοί αλγόριθμοι για την επεξεργασία τους. Επομένως, το υλικό του υπολογιστή έχει επίδραση στο είδος των αλγορίθμων που θα χρησιμοποιηθούν.



Ο σκληρός δίσκος και η μνήμη flash, αποτελούν παραδείγματα βοηθητικής μνήμης του υπολογιστή.

Τα δεδομένα μπορεί να είναι απλές μεταβλητές, οι οποίες λαμβάνουν μία τιμή κάθε φορά (απλά δεδομένα) ή μπορούν να αποθηκεύονται ως μία δομή δεδομένων.

Δομή δεδομένων (data structure) είναι ένα σύνολο αποθηκευμένων δεδομένων, τα οποία είναι έτσι οργανωμένα, ώστε να υπόκεινται σε συγκεκριμένες απαιτούμενες επεξεργασίες.

Ο όρος αναφέρεται σε ένα σύνολο δεδομένων μαζί με ένα σύνολο λειτουργιών που επιτρέπονται στα δεδομένα αυτά. Οι δομές δεδομένων είναι πολύ στενά συνδεδεμένες με την έννοια του αλγορίθμου. Είναι πολύ χαρακτηριστική η ακόλουθη «σχέση» που διατύπωσε ο Νικλάους Βιρθ (Niklaus Wirth), δημιουργός της γλώσσας Pascal:

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

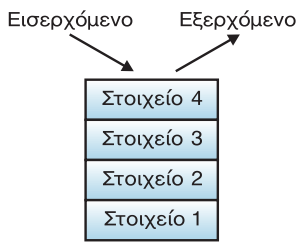
Η ανωτέρω σχέση έχει την έννοια ότι αν κάποιος διαθέτει τον κατάλληλο αλγόριθμο και τις δομές δεδομένων, οι οποίες θα χρησιμοποιηθούν, είναι εντελώς άμεση η μετατροπή και υλοποίησή του σε πρόγραμμα σε γλώσσα υπολογιστή. Κάθε δομή δεδομένων αποτελείται, στην πιο γενική της μορφή, από ένα σύνολο στοιχείων ή κόμβων.

Οι πιο ευρέως χρησιμοποιούμενες δομές δεδομένων είναι ο πίνακας, η στοίβα, η ουρά, η λίστα, το δένδρο και ο γράφος.

Δισδιάστατος Πίνακας Α 4x3		
Στοιχείο 1,1	Στοιχείο 1,2	Στοιχείο 1,3
Στοιχείο 2,1	Στοιχείο 2,2	Στοιχείο 2,3
Στοιχείο 3,1	Στοιχείο 3,2	Στοιχείο 3,3
Στοιχείο 4,1	Στοιχείο 4,2	Στοιχείο 4,3

Εικόνα 2.13. Δισδιάστατος πίνακας

Ο ***πίνακας*** (table) αποτελείται από ένα σύνολο ομοειδών απλών στοιχείων, καθένα από τα οποία καθορίζεται με τη βοήθεια ενός ή περισσοτέρων δεικτών. Ένας πίνακας μπορεί να είναι μίας, δύο ή περισσοτέρων διαστάσεων, ανάλογα με το πλήθος δεικτών που χρειάζονται για να καθοριστεί η θέση του. Στην εικόνα 2.13, παρουσιάζεται ένας δισδιάστατος πίνακας Α με 4 γραμμές και 3 στήλες. Στο παράδειγμα 2.5 είχε παρουσιαστεί ένας μονοδιάστατος πίνακας.



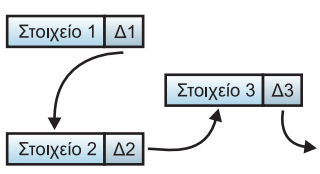
Εικόνα 2.14. Στοίβα

Μία ***στοίβα*** (stack) είναι μια γραμμική διάταξη στοιχείων, στην οποία εισάγονται και εξάγονται στοιχεία μόνο από το ένα άκρο (εικόνα 2.14). Η λειτουργία της εισαγωγής αποκαλείται ***ώθηση*** (push) και της εξαγωγής ***απώθηση*** (pull ή pop). Η φιλοσοφία εισαγωγής και εξαγωγής των στοιχείων ονομάζεται ***LIFO*** (Last In, First Out), δηλαδή το τελευταίο εισαγόμενο δεδομένο εξέρχεται και πρώτο.



Εικόνα 2.15. Ουρά

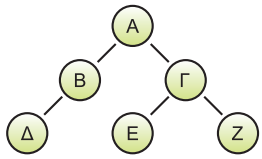
Μια ***ουρά*** (queue) αποτελεί μια γραμμική διάταξη στοιχείων, στην οποία εισάγονται νέα στοιχεία από ένα άκρο και εξάγονται υπάρχοντα στοιχεία από το άλλο άκρο (εικόνα 2.15). Η λειτουργία της ουράς αποκαλείται ***FIFO*** (First In, First Out), δηλαδή το στοιχείο το οποίο εισάγεται πρώτο στην ουρά εξέρχεται και πρώτο.



Εικόνα 2.16. Λίστα

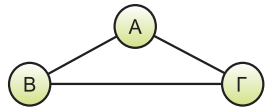
Σε μια (συνδεσμική) ***λίστα*** (linked list) τα στοιχεία φαίνονται «λογικά» ότι είναι γραμμικά διατεταγμένα, χωρίς όμως αυτό να σημαίνει ότι βρίσκονται σε συνεχόμενες θέσεις της μνήμης του υπολογιστή (εικόνα 2.16). Ανεξάρτητα από τη θέση που καταλαμβάνει στη μνήμη ένα δεδομένο, συσχετίζεται με το επόμενό του με τη βοήθεια κάποιου ***δείκτη*** (pointer).

Το ***δένδρο*** (tree) είναι μη γραμμική δομή που αποτελείται από ένα σύνολο κόμβων, οι οποίοι συνδέονται με ακμές (εικόνα 2.17). Υπάρχει μόνο ένας κόμβος, από τον οποίο μόνο ξεκινούν ακμές, που ονομάζεται ***ρίζα*** (root). Σε όλους τους άλλους κόμβους καταλήγει μία ακμή και ξεκινούν καμία, μία ή περισσότερες. Οι κόμβοι στους οποίους καταλήγουν μόνο ακμές, ονομάζονται ***φύλλα***.



Εικόνα 2.17. Δένδρο

Ο ***γράφος*** (graph) αποτελεί τη πιο γενική δομή δεδομένων μια και αποτελείται από κόμβους και ακμές χωρίς όμως κάποια ιεράρχηση.



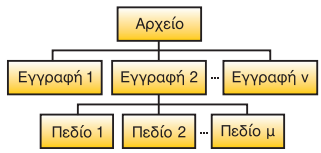
Εικόνα 2.18. Γράφος

Υπάρχουν διάφοροι τρόποι διάκρισης των δομών δεδομένων. Διακρίνονται σε ***στατικές*** και ***δυναμικές***. Οι στατικές δομές έχουν σταθερό μέγεθος και μπορούν να κατακρατήσουν συγκεκριμένο πλήθος στοιχείων. Αντίθετα οι δυναμικές δομές δεν έχουν σταθερό μέγεθος και το πλήθος των στοιχείων τους μπορεί να μεγαλώνει ή να μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται άλλα.

Οι δομές δεδομένων διακρίνονται επίσης σε ***γραμμικές*** και ***μη γραμμικές***. Στις γραμμικές δομές μπορεί να ορισθεί κάποια σχέση διάταξης για δύο οποιαδήποτε διαδοχικά στοιχεία τους. Αυτό σημαίνει ότι κάποιο στοιχείο θα είναι πρώτο και κάποιο τελευταίο. Οποιοδήποτε από τα υπόλοιπα θα έπεται από το προηγούμενό του και θα προηγείται από το επόμενό του. Στις μη γραμμικές δομές δεν μπορεί να οριστεί μια σχέση διάταξης όπως η παραπάνω. Τέτοιες δομές είναι τα δένδρα και οι γράφοι. Στα δένδρα ένας κόμβος έχει έναν προηγούμενο αλλά πιθανόν πολλούς επόμενους. Στους γράφους κάθε κόμβος μπορεί να έχει πολλούς προηγούμενους και πολλούς επόμενους.

Τέλος διάκριση των δομών μπορεί να γίνει και ανάλογα με το είδος της χρησιμοποιούμενης μνήμης (κύρια ή βοηθητική). Οι δομές δεδομένων βοηθητικής μνήμης αποκαλούνται ***αρχεία δεδομένων*** (data files). Ένα αρχείο απαρτίζεται από έναν αριθμό ομοειδών εγγραφών (records). Κάθε εγγραφή διαθέτει ορισμένα πεδία (fields), που περιέχουν δεδομένα για μια οντότητα (π.χ. μαθητής), όπως φαίνεται στην εικόνα 2.19.

Για παράδειγμα, η εγγραφή ενός μαθητή μπορεί να αποτελείται από το όνομα, το επώνυμο, τον αριθμό κινητού τηλεφώνου, τη διεύθυνση αλληλογραφίας, την ηλεκτρονική διεύθυνση, τη φωτογραφία του κ.ά., που καλούνται πεδία της εγγραφής.



Εικόνα 2.19. Δομή αρχείου

Αλφάβητο

Το σύνολο των χαρακτήρων που χρησιμοποιούνται στην ψευδογλώσσα περιλαμβάνει:

- όλα τα γράμματα της ελληνικής ή αγγλικής αλφαβήτου πεζά και κεφαλαία
- τους αριθμητικούς χαρακτήρες 0-9
- τους επόμενους ειδικούς χαρακτήρες

"	εισαγωγικά (διπλά)
( )	παρενθέσεις
[ ]	αγκύλες
*	αστερίσκος
+	συν
,	κόμμα
-	μείον
.	τελεία
/	κάθετος
!	θαυμαστικό
<	μικρότερο από
=	ίσον
>	μεγαλύτερο από
≤	μικρότερο ή ίσο

2.2.7 Εντολές και δομές αλγορίθμου

Στην παράγραφο αυτή δίδονται διάφορα παραδείγματα αλγορίθμων, όπου εξετάζονται τα συστατικά μέρη ενός αλγορίθμου και οι τρεις συνιστώσες του (δομή ακολουθίας, δομή επιλογής και δομή επανάληψης) ξεκινώντας από τις απλούστερες και προχωρώντας προς τις συνθετότερες. Στα περιθώρια παρουσιάζονται ορισμένα βασικά εισαγωγικά στοιχεία της χρησιμοποιούμενης ψευδογλώσσας.

Κάθε αλγόριθμος διατυπωμένος σε ψευδογλώσσα ξεκινά με τη γραμμή

Αλγόριθμος *όνομα_αλγορίθμου*
και τελειώνει με τη γραμμή
Τέλος *όνομα_αλγορίθμου*

≥ μεγαλύτερο ή ίσο
≠ διάφορο
^ άνω βέλος
_ κάτω παύλα
κενό

- και ένα γραφικό σύμβολο το ← (αριστερό βέλος)

Σταθερές
Οι σταθερές στην ψευδογλώσσα μπορεί να είναι αριθμητικές, αλφαριθμητικές ή λογικές. Για το σχηματισμό μιας αριθμητικής σταθεράς χρησιμοποιούνται οι αριθμητικοί χαρακτήρες και πιθανά ένας από τους χαρακτήρες +, -. Επίσης, μπορεί να χρησιμοποιηθεί το κόμμα για το δεκαδικό σημείο. Π.χ. 5, 123,27, -1, 1000000 κ.λπ.
Για το σχηματισμό μιας αλφαριθμητικής σταθεράς χρησιμοποιούνται οποιοιδήποτε χαρακτήρες περικλειόμενοι σε διπλά εισαγωγικά.
Μια σταθερά μπορεί να έχει οποιοδήποτε πλήθος αριθμητικών ή αλφαριθμητικών χαρακτήρων αντίστοιχα.
Οι λογικές σταθερές είναι δύο, η Αληθής και Ψευδής.

Μεταβλητές
Για το σχηματισμό του ονόματος μιας μεταβλητής χρησιμοποιείται οποιοσδήποτε αριθμός αλφαβητικών ή αριθμητικών χαρακτήρων και ο χαρακτήρας κάτω παύλα. Ο πρώτος χαρακτήρας της μεταβλητής πρέπει να είναι αλφαβητικός και δεν μπορεί να χρησιμοποιηθεί δεσμευμένη λέξη ως όνομα μεταβλητής.
Οι μεταβλητές χαρακτηρίζονται ως αριθμητικές, αλφαριθμητικές ή λογικές ανάλογα με την τιμή που θα αποδοθεί σε αυτές. Πριν από την απόδοση κάποιας τιμής σε μια μεταβλητή (με εντολή εισόδου ή εκχώρησης) η μεταβλητή έχει απροσδιόριστη τιμή.
Οι σταθερές και οι μεταβλητές καλούνται και τελεστέοι.

Μεταξύ αυτών των δύο γραμμών γράφονται οι εντολές του αλγορίθμου. Οι εντολές είναι λέξεις (συνήθως ρήματα σε προστακτική) ή συμβολισμοί που προσδιορίζουν μία σαφή ενέργεια. Οι λέξεις που έχουν αυστηρά καθορισμένο νόημα στην ψευδογλώσσα καλούνται δεσμευμένες λέξεις και στο πλαίσιο του βιβλίου θα γράφονται με έντονα μπλε γράμματα.

Οι εντολές γράφονται σε ξεχωριστές γραμμές. Επεξηγηματικά σχόλια μπορούν να γράφονται οπουδήποτε στο σώμα του αλγορίθμου. Ένα σχόλιο αρχίζει με το χαρακτήρα θαυμαστικό (!) και στο πλαίσιο του βιβλίου θα γράφεται με πλάγια γράμματα.

Στη συνέχεια επεξηγούνται οι διάφορες εντολές που μπορούν να χρησιμοποιηθούν για τη σύνταξη ενός αλγορίθμου.

2.2.7.1 Εκχώρηση, Είσοδος και Έξοδος τιμών

Η γενική μορφή της εντολής εκχώρησης είναι:

Μεταβλητή ← Έκφραση

και η λειτουργία της είναι «εκτελούνται οι πράξεις στην έκφραση και η τιμή της εκχωρείται (αποδίδεται, μεταβιβάζεται) στη μεταβλητή».

Στην εντολή χρησιμοποιείται το αριστερό βέλος προκειμένου να δείχνει τη φορά της εκχώρησης. Για το σκοπό αυτό χρησιμοποιούνται διάφορα σύμβολα από τις γλώσσες προγραμματισμού. Για παράδειγμα στην Pascal και Delphi χρησιμοποιείται το :=, ενώ στην Basic και τη C το =. Προσοχή, λοιπόν, το σύμβολο της ισότητας (=) στις γλώσσες προγραμματισμού ή το αριστερό βέλος (←) στην ψευδογλώσσα, δεν είναι σύμβολο εξίσωσης. Το σύμβολο = χρησιμοποιείται ως τελεστής σύγκρισης.

Αριστερά του συμβόλου ← υπάρχει πάντα μόνο μια μεταβλητή, ενώ δεξιά μπορεί να υπάρχει σταθερά, μεταβλητή ή έκφραση.

Η εκχώρηση τιμών επιτυγχάνεται και με τις εντολές εισόδου. Η εντολή

Διάβασε *λίστα_μεταβλητών*

επιτρέπει την είσοδο τιμών και την εκχώρηση αυτών στις μεταβλητές που αναφέρονται στη λίστα_μεταβλητών.

Η εντολή **Διάβασε** διαφέρει από την εντολή εκχώρησης, γιατί στη δεύτερη οι τιμές των μεταβλητών προσδιορίζονται κατά τη συγγραφή του αλγορίθμου, ενώ στην πρώτη κατά την εκτέλεση του αλγορίθμου.

Για την έξοδο τιμών (αποτελεσμάτων) μπορούν να χρησιμοποιηθούν οι εντολές **Γράψε, Εμφάνισε** ή **Εκτύπωσε** με ίδια σύνταξη. Κάθε μία από αυτές τις εντολές συνοδεύεται από μια λίστα μεταβλητών ή σταθερών. Π.χ. **Γράψε** "Τιμή:", αξία.

2.2.7.2 Δομή ακολουθίας

Η δομή ακολουθίας χρησιμοποιείται για την αντιμετώπιση προβλημάτων στα οποία οι εντολές εκτελούνται η μία μετά την άλλη από πάνω προς τα κάτω.

Παράδειγμα 2.8. Είσοδος και έξοδος αριθμών

Να διαβαστούν δύο αριθμοί και να υπολογιστεί και να εμφανιστεί το άθροισμά τους.

Αλγόριθμος Άθροισμα
Διάβασε α, β
Σ ← α + β
Εμφάνισε Σ
Τέλος Άθροισμα

Η πρώτη ενέργεια που γίνεται είναι η εισαγωγή δεδομένων. Αυτό επιτυγχάνεται με τη χρήση της εντολής Διάβασε. Μετά την εκτέλεση της εντολής αυτής στις μεταβλητές α και β έχουν εκχωρηθεί τιμές, οπότε υπάρχει η δυνατότητα επεξεργασίας των τιμών. Εδώ απαιτείται η πρόσθεση των δύο αριθμών και η απόδοση του αθροίσματος σε μια άλλη μεταβλητή, τη Σ, που επιτυγχάνεται με την επόμενη εντολή εκχώρησης. Τελευταία ενέργεια αποτελεί η εμφάνιση (ή εκτύπωση) του αποτελέσματος.

Παράδειγμα 2.9. Υπολογισμός τελικής αξίας είδους

Να γραφεί αλγόριθμος, ο οποίος να διαβάζει την καθαρή αξία ενός είδους και το ποσοστό ΦΠΑ και να υπολογίζει και να εκτυπώνει την τελική αξία.

Αλγόριθμος Υπολογισμός
Διάβασε ΚΑ, ΠΦΠΑ
ΤΑ ← ΚΑ + ΚΑ * ΠΦΠΑ / 100
Εκτύπωσε "Τελική Αξία:", ΤΑ
Τέλος Υπολογισμός

Η τελική αξία (ΤΑ) ενός είδους βρίσκεται, αν στην καθαρή αξία (ΚΑ) προστεθεί η αξία ΦΠΑ. Αυτό επιτυγχάνεται με την εντολή εκχώρησης.

Ο ΦΠΑ (Φόρος Προστιθέμενης Αξίας) είναι ένας φόρος που επιβάλλεται σε κάθε προϊόν που πωλείται ή σε κάθε παρεχόμενη υπηρεσία. Σήμερα για τα περισσότερα είδη το ποσοστό του ΦΠΑ είναι 23%, για την εστίαση και είδη πρώτης ανάγκης είναι 13% και για τα βιβλία 6,5%. Όμως στα νησιά του Αιγαίου (εκτός της Κρήτης) εφαρμόζονται μειωμένοι συντελεστές. Τα ποσοστά του ΦΠΑ αλλάζουν με κυβερνητική απόφαση.

Οι εντολές εισόδου/εξόδου μπορούν να συνδυάζονται προκειμένου να είναι πιο κατανοητή η ενέργεια που απαιτείται από το χρήστη του προγράμματος που θα υλοποιεί έναν αλγόριθμο.

Εμφάνισε "Δώστε τιμές για τα α και β"
Διάβασε α, β

Τελεστές
Τελεστές είναι τα σύμβολα και οι λέξεις που χρησιμοποιούνται στις διάφορες πράξεις. Υπάρχουν οι επόμενοι τελεστές:

➢ **Αριθμητικοί**
Οι αριθμητικοί τελεστές χρησιμοποιούνται για την εκτέλεση αριθμητικών πράξεων. Είναι οι:

+	για πρόσθεση
-	για αφαίρεση
*	για πολλαπλασιασμό
/	για διαίρεση
mod	για το υπόλοιπο ακέραιας διαίρεσης
div	για το πηλίκο ακέραιας διαίρεσης
^	για ύψωση σε δύναμη

➢ **Σχεσιακοί**
Οι σχεσιακοί τελεστές χρησιμοποιούνται για τη σύγκριση δύο τιμών. Το αποτέλεσμα μιας σύγκρισης είναι είτε Αληθής είτε Ψευδής. Οι σχεσιακοί ή συγκριτικοί τελεστές είναι οι επόμενοι:

<	μικρότερο
>	μεγαλύτερο
=	ίσο
≤	μικρότερο ή ίσο
≥	μεγαλύτερο ή ίσο
≠	διάφορο

Αντί των τριών τελευταίων τελεστών μπορεί να χρησιμοποιηθούν και οι συνδυασμοί των χαρακτήρων <=, >= και < > αντίστοιχα.

➢ **Λογικοί**

Οι λογικοί τελεστές υλοποιούν τις λογικές πράξεις. Το αποτέλεσμα μιας λογικής πράξης είναι Αληθής ή Ψευδής. Λογικοί τελεστές είναι:

όχι	πράξη άρνησης
και	πράξη σύζευξης
ή	πράξη διάζευξης

➢ **Συναρτησιακοί τελεστές ή Συναρτήσεις**

Μια συνάρτηση χρησιμοποιείται για να εκτελέσει μια προκαθορισμένη λειτουργία. Κάθε συνάρτηση έχει ένα όνομα ακολουθούμενο από ζεύγος παρενθέσεων που περικλείουν μια μεταβλητή ή μια σταθερά ή γενικότερα μια έκφραση. Στην ψευδογλώσσα μπορούν να χρησιμοποιηθούν όλες οι συνηθισμένες συναρτήσεις, όπως οι τριγωνομετρικές **ΗΜ**(x), **ΣΥΝ**(x), **ΕΦ**(x), οι μαθηματικές **Α_Τ**(x) για την απόλυτη τιμή, **Ε**(x) για την e^x, **ΛΟΓ**(x) για το δεκαδικό λογάριθμο, **ΛΝ**(x) για το φυσικό λογάριθμο, **Τ_Ρ**(x) για την τετραγωνική ρίζα, και **Α_Μ**(x) για το ακέραιο μέρος.

Εναλλακτική είσοδος και έξοδος τιμών παρέχεται με τη χρήση των εντολών Δεδομένα και Αποτελέσματα. Η εντολή **Δεδομένα** γράφεται δεύτερη (μετά την εντολή Αλγόριθμος) και περιγράφει εντός των συμβόλων // .... // τα δεδομένα του αλγορίθμου, δηλαδή τις μεταβλητές ***που έχουν ήδη κάποια τιμή***. Αντίστοιχα η εντολή Αποτελέσματα γράφεται προτελευταία και περιέχει τις μεταβλητές εξόδου. Οι επόμενοι δύο αλγόριθμοι για τις ίδιες τιμές εισόδου έχουν την ίδια τιμή εξόδου.

Αλγόριθμος Άθροισμα_1	**Αλγόριθμος** Άθροισμα_2
Διάβασε α, β	**Δεδομένα** // α, β //
Σ ← α + β	Σ ← α + β
Γράψε Σ	**Αποτελέσματα** // Σ //
Τέλος Άθροισμα_1	**Τέλος** Άθροισμα_2

Η χρήση των εντολών Δεδομένα και Αποτελέσματα γενικά προτιμάται προκειμένου ο αλγόριθμος να απαλλαγεί από τις λεπτομέρειες εισόδου/εξόδου και να επικεντρωθεί στο πρόβλημα που επιλύει (εκτός βέβαια αν το πρόβλημα είναι η εισαγωγή δεδομένων). Επίσης η χρήση τους συνιστάται στην περίπτωση που τα δεδομένα εισόδου ή/και εξόδου είναι πολυπληθή, όπως για παράδειγμα σε προβλήματα επεξεργασίας πινάκων. Τέλος η χρήση τους επιβάλλεται, στην περίπτωση που ένας αλγόριθμος καλείται από άλλον (βλ. παρ. 2.2.7.5).



Αν και η χρήση της μιας ή της άλλης μεθόδου αφήνεται γενικά στην ευχέρεια του συντάκτη του αλγορίθμου, ο μαθητής χρειάζεται να είναι προσεκτικός στη χρήση των δύο εντολών. Έτσι αν η εκφώνηση ενός θέματος λέει «Να γραφεί αλγόριθμος ο οποίος να διαβάζει ....»», τότε είναι απαραίτητο να χρησιμοποιηθεί η εντολή Διάβασε. Αντίθετα αν η εκφώνηση λέει «Δίδεται ένας πίνακας Α. Να γραφεί αλγόριθμος ο οποίος ...», τότε χρειάζεται να χρησιμοποιηθεί η εντολή Δεδομένα.

2.2.7.3 Δομή επιλογής

Στην πράξη πολύ λίγα προβλήματα μπορούν να επιλυθούν με τον προηγούμενο τρόπο της σειριακής/ακολουθιακής δομής ενεργειών. Συνήθως τα προβλήματα έχουν κάποιες ιδιαιτερότητες και δεν μπορούν να εκτελεστούν τα ίδια βήματα για κάθε περίπτωση. Τις πιο πολλές φορές λαμβάνονται κάποιες αποφάσεις με βάση κάποια κριτήρια που μπορεί να είναι διαφορετικά για κάθε στιγμιότυπο ενός προβλήματος. Για παράδειγμα το πρόβλημα της εξόδου (από το σπίτι) σχετίζεται με τις καιρικές συνθήκες. Έτσι κάποιος μπορεί να πει ότι, «αν βρέχει, θα πάρω ομπρέλα».

Με τη δομή επιλογής μπορεί να τροποποιηθεί η σειρά εκτέλεσης των εντολών ενός αλγορίθμου. Η διαδικασία επιλογής περιλαμβάνει τον έλεγχο μιας συνθήκης που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής) και ακολουθεί η απόφαση εκτέλεσης εντολών με βάση την τιμή αυτής της συνθήκης. Ως συνθήκη εννοείται μια λογική έκφραση στην οποία

Εκφράσεις

Μια έκφραση μπορεί να είναι μια σταθερά, μια μεταβλητή, μια συνάρτηση ή ένας συνδυασμός σταθερών, μεταβλητών, συναρτήσεων, τελεστών και παρενθέσεων.
Σε μία έκφραση που αποτελείται από συνδυασμό στοιχείων, εκτελούνται οι πράξεις επί των σταθερών και μεταβλητών που ορίζουν οι τελεστές. Οι χρησιμοποιούμενοι τελεστές έχουν διαφορετική ιεραρχία. Αυτό σημαίνει ότι κάποιες πράξεις μπορεί να προηγούνται από κάποιες άλλες σε μια έκφραση.

Η ιεραρχία των πράξεων είναι η ακόλουθη:

Α. Αριθμητικοί τελεστές
Σε κάθε έκφραση που υπάρχουν αριθμητικοί τελεστές, ακολουθείται η προσδιορισμένη από τα μαθηματικά ιεραρχία των πράξεων.
1. Ύψωση σε δύναμη
2. Πολλαπλασιασμός, Διαίρεση, Πηλίκο ακέραιας διαίρεσης, Υπόλοιπο ακέραιας διαίρεσης
3. Πρόσθεση, Αφαίρεση

Β. Σχεσιακοί τελεστές

Γ. Λογικοί τελεστές
1. όχι
2. και
3. ή

Αν οι πράξεις είναι ίδιας ιεραρχίας, τότε εκτελούνται από τα αριστερά προς τα δεξιά.
Οι τελεστέοι ενός αριθμητικού τελεστή πρέπει να είναι αριθμητικές εκφράσεις. (π.χ. α + β^3)
Στις λογικές εκφράσεις μπορούν να χρησιμοποιηθούν όλοι οι τελεστές. Αν μία λογική έκφραση περιλαμβάνει τελεστές, τότε ένας τουλάχιστον πρέπει να είναι λογικός ή συγκριτικός. (π.χ. α + β^3 > 5 και γ / 3 < 2)
Οι συγκριτικοί τελεστές συνδυάζονται με εκφράσεις ιδίου τύπου, ενώ οι λογικοί τελεστές μόνο με λογικές εκφράσεις. (π.χ. α + β > 3, "ΑΒ" < "Γ")

Οι συγκρίσεις λογικών εκφράσεων έχουν νόημα μόνο στην περίπτωση του = και ≠.

➢ Αριθμητικές εκφράσεις

Στη συνέχεια παρουσιάζονται μερικά παραδείγματα και διευκρινίσεις που αφορούν τις αριθμητικές εκφράσεις.

❑ Στην έκφραση 5 + 12 / 3 * 2 - 1 οι πράξεις εκτελούνται με την επόμενη σειρά
1. 12 / 3 (= 4)
2. 4 * 2 (= 8)
3. 5 + 8 (= 13)
4. 13 - 1 (= 12)

Σε μια έκφραση μπορούν να χρησιμοποιηθούν και παρενθέσεις. Οι παρενθέσεις μπορεί να μεταβάλλουν την προτεραιότητα των πράξεων.

❑ Στην έκφραση 4 * (1 + 2) εκτελείται πρώτα η πρόσθεση (1 + 2 = 3) και μετά ο πολλαπλασιασμός (4 * 3 = 12)

❑ Παρατίθεται ένας πίνακας με τη γραφή μερικών μαθηματικών τύπων ως εκφράσεις της ψευδογλώσσας.

Πίνακας 2.4. Μαθηματικοί τύποι ως εκφράσεις της ψευδογλώσσας

Μαθηματικός τύπος	Έκφραση ψευδογλώσσας
$\frac{x - y}{z}$	(x - y) / z
$\frac{xy}{z + 1}$	x * y / (z + 1)
$(x^2)^3$	(x^2)^3
x^{2^3}	x^(2^3)
x(-y)	x * (-y)

Στην ακέραια διαίρεση οι τελεστέοι των τελεστών div και mod είναι υποχρεωτικά θετικοί ακέραιοι αριθμοί.

➢ Λογικές εκφράσεις

Οι σχεσιακοί τελεστές χρησιμοποιούνται για τη σύγκριση δύο τιμών. Το αποτέλεσμα μιας σύγκρισης μπορεί να είναι είτε Αληθής είτε Ψευδής.

❑ Στην έκφραση x + y < (z - 1) / t το αποτέλεσμα είναι Αληθής, αν το αποτέλεσμα της πράξης x + y είναι μικρότερο από το αποτέλεσμα της πράξης z - 1 διαιρούμενο δια t.

Οι σχεσιακοί τελεστές μπορούν να χρησιμοποιηθούν και με αλφαριθμητικούς τελεστέους. Η σύγκριση αλφαριθμητικών εκφράσεων πραγματοποιείται βάσει διεθνών προτύπων που στοχεύουν στην κωδικοποίηση όλων των συστημάτων γραφής. Ο υπολογισμός του αποτελέσματος, λοιπόν, της σύγκρισης αλφαριθμητικών εκφράσεων που περιέχουν οποιονδήποτε χαρακτήρα είναι πέρα από τους στόχους του μαθήματος. Για το λόγο αυτό, στην ψευδογλώσσα θα γίνεται σύγκριση αλφαριθμητικών εκφράσεων που περιέχουν μόνο τα κεφαλαία γράμματα του ελληνικού αλφαβήτου, στα οποία ισχύει η αλφαβητική σειρά. Για παράδειγμα η λογική έκφραση "ΑΔΓ" > "ΑΒΚ" είναι αληθής, διότι κατά τη σύγκριση χαρακτήρα προς χαρακτήρα από αριστερά προς τα δεξιά εντοπίζεται ότι το γράμμα Δ είναι διαφορετικό και μεγαλύτερο του γράμματος Β.

Οι λογικοί τελεστές πραγματοποιούν τις λογικές πράξεις σε μια έκφραση. Το αποτέλεσμα μιας λογικής πράξης είναι πάντα Αληθής ή Ψευδής, σύμφωνα με τον επόμενο πίνακα τιμών, όπου με Χ και Υ εννοούνται δύο λογικές εκφράσεις, στις οποίες χρησιμοποιούνται μόνο αριθμητικοί και σχεσιακοί τελεστές.

Πίνακας 2.5. Πίνακας τιμών δύο λογικών εκφράσεων (όχι, και, ή)

Χ	Υ	όχι Χ	Χ και Υ	Χ ή Υ
Αληθής	Αληθής	Ψευδής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Αληθής	Ψευδής	Ψευδής

Σε μια λογική έκφραση οι λογικές πράξεις εκτελούνται μετά τις αριθμητικές και συγκριτικές.

❑ Η σχέση $0 < x < 2k + 1$ γράφεται στην ψευδογλώσσα $x > 0$ **και** $x < 2 * k + 1$ και είναι αληθής, αν x θετικό και ταυτόχρονα μικρότερο του $2 * k + 1$.

Ο τελεστής **όχι** έχει έναν τελεστέο, ενώ οι **και, ή** έχουν δύο (ή περισσότερους).

Προσοχή. *Ο συγκριτικός τελεστής ≤ είναι ένας και διαβάζεται «μικρότερο ή ίσο». Δεν πρέπει να αναλύεται και σε μια λογική έκφραση με χρήση του λογικού τελεστή* ***ή****, διότι υπάρχει μεγάλη πιθανότητα να προκληθεί λάθος, ιδιαίτερα αν υπάρχουν πολλοί τελεστές σε μια σύνθετη έκφραση.*

❑ **Η σχέση** $0 < x \leq 10$ γράφεται στην ψευδογλώσσα $x > 0$ **και** $x \leq 10$

Στην πιο πάνω έκφραση δεν μπορούμε να αναλύσουμε την $x \leq 10$ σε $x = 10$ ή $x < 10$, διότι θα λάβουμε την $x > 0$ **και** $x = 10$ **ή** $x < 10$ η οποία για $x = -1$ θα δώσει τιμή Αληθής. Συνιστάται ανεπιφύλακτα να χρησιμοποιούνται παρενθέσεις όταν σε έκφραση υπάρχουν πολλοί λογικοί τελεστές.

υπάρχει τουλάχιστον ένας σχεσιακός τελεστής (δηλαδή η συνθήκη δεν μπορεί να απαρτίζεται από μόνο μια μεταβλητή ή μια σταθερά ή μια αριθμητική παράσταση).

Σύγκριση αριθμών με απλή επιλογή

Παράδειγμα 2.10. Να διαβαστεί ένας αριθμός και να εμφανιστεί η απόλυτη τιμή του.

Η απόλυτη τιμή ενός αριθμού είναι ο ίδιος ο αριθμός, αν είναι θετικός ή ο αντίθετός του, αν είναι αρνητικός. Έτσι για να υπολογιστεί η απόλυτη τιμή ενός αριθμού αρκεί να ελεγχθεί, αν τυχόν ο δεδομένος αριθμός είναι αρνητικός και αν ναι, να βρεθεί ο αντίθετός του. Ο συλλογισμός αυτός οδηγεί στον επόμενο αλγόριθμο.

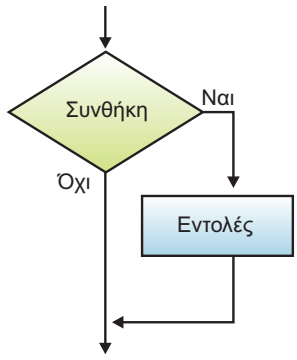
```
Αλγόριθμος Απόλυτη_τιμή1
Διάβασε α
Αν α < 0 τότε
        α ← α * (-1)
        ! α ←-α
Τέλος_αν
Εμφάνισε α
Τέλος Απόλυτη_τιμή1
```

```
Αλγόριθμος Απόλυτη_τιμή2
Διάβασε α
! Η εμφάνιση της απόλυτης τιμής
! μπορεί να γίνει με τη χρήση της
! συνάρτησης Α_Τ(α)
Εμφάνισε Α_Τ(α)
Τέλος Απόλυτη_τιμή2
```

Απλή εντολή επιλογής

Αν Συνθήκη **τότε**
Εντολές
Τέλος_αν

Αν η συνθήκη είναι αληθής, τότε εκτελούνται οι εντολές. Οι εντολές μπορούν να είναι μία ή περισσότερες.



Εικόνα 2.20. Διάγραμμα ροής της απλής εντολής επιλογής

Παράδειγμα 2.11. Να αναπτύξετε αλγόριθμο ο οποίος με δεδομένα τα μήκη τριών ευθυγράμμων τμημάτων θα υπολογίζει και θα εμφανίζει το εμβαδόν του τριγώνου που μπορούν να σχηματίσουν, με βάση τον τύπο του Ήρωνα $E = \sqrt{\tau\,(\tau - \alpha)\,(\tau - \beta)\,(\tau - \gamma)}$, όπου τ είναι η ημιπερίμετρος του τριγώνου $\tau = (\alpha + \beta + \gamma) / 2$ και α, β, γ τα μήκη των ευθυγράμμων τμημάτων. Σε περίπτωση που τα ευθύγραμμα τμήματα δεν μπο-

ρούν να σχηματίσουν τρίγωνο, εμφανίζεται κατάλληλο μήνυμα. Για να σχηματιστεί τρίγωνο θα πρέπει το άθροισμα των μηκών δύο οποιονδήποτε ευθυγράμμων τμημάτων να είναι μεγαλύτερο από το μήκος του άλλου τμήματος.

```
Αλγόριθμος Εμβαδό
Δεδομένα // α, β, γ //
Αν α + β > γ και β + γ > α και γ + α > β τότε
        τ ← (α + β + γ) / 2
        Εμβ ← T_P(τ * (τ - α) * (τ - β) * (τ - γ))
        Εμφάνισε Εμβ
αλλιώς
        Εμφάνισε "Δεν σχηματίζεται τρίγωνο"
Τέλος_αν
Τέλος Εμβαδό
```

Παράδειγμα 2.12. Το όζον (O_3) αποτελεί έναν από τους ρύπους που προκαλούν μόλυνση στην ατμόσφαιρα. Σε περιπτώσεις που ο ρύπος αυτός ξεπεράσει τα 300 μg/m^3 τότε πρέπει να ληφθούν μέτρα. Να αναπτυχθεί αλγόριθμος ο οποίος θα διαβάζει την τιμή του O_3 και θα εκτυπώνει το αντίστοιχο μήνυμα σύμφωνα με τον παρακάτω πίνακα:

Τιμές O_3 (μg/m^3)	Μήνυμα
Τιμή > 250	Προειδοποίηση
Τιμή > 300	Μέτρα Α
Τιμή > 500	Μέτρα Β

Επιπλέον, σε περίπτωση που έχουν ξεπεραστεί τα όρια, θα εκτυπώνει κατά πόσο τα ξεπέρασε.

```
Αλγόριθμος Όζον1
Διάβασε τ
Αν τ > 250 και τ ≤ 300 τότε
    Εκτύπωσε "Προειδοποίηση"
αλλιώς_αν τ > 300 και τ ≤ 500 τότε
    πο ← τ - 300
    Εκτύπωσε "Μέτρα Α", πο
αλλιώς_αν τ > 500 τότε
    πο ← τ - 300
    Εκτύπωσε "Μέτρα Β", πο
Τέλος_αν
Τέλος Όζον1
```

```
Αλγόριθμος Όζον2
Διάβασε τ
Αν τ > 500 τότε
    πο ← τ - 300
    Εκτύπωσε "Μέτρα Β", πο
αλλιώς_αν τ > 300 τότε
    πο ← τ - 300
    Εκτύπωσε "Μέτρα Α", πο
αλλιώς_αν τ > 250 τότε
    Εκτύπωσε "Προειδοποίηση"
Τέλος_αν
Τέλος Όζον2
```

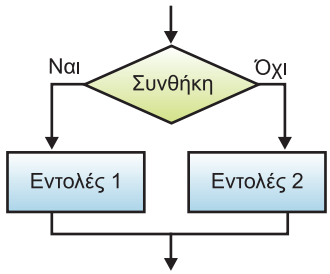
Εμφωλευμένες εντολές επιλογής

Σε όλες τις προηγούμενες περιπτώσεις όπου αναφέρεται εντολή ή εντολές, τίποτα δεν απαγορεύει αυτές οι εντολές να είναι επίσης εντολές επιλογής. Αναφερόμαστε τότε σε εμφωλευμένες εντολές επιλογής.

Σύνθετη εντολή επιλογής

```
Αν Συνθήκη τότε
        Εντολές_1
αλλιώς
        Εντολές_2
Τέλος_αν
```

Αν η συνθήκη είναι αληθής, τότε εκτελούνται οι εντολές 1, αλλιώς (δηλαδή αν η συνθήκη είναι ψευδής) εκτελούνται οι εντολές 2.



Εικόνα 2.21. Διάγραμμα ροής της σύνθετης εντολής επιλογής

Πολλαπλή

Αν συνθήκη_1 **τότε**
εντολές_1
αλλιώς_αν *συνθήκη_2* **τότε**
εντολές_2
.....
αλλιώς_αν *συνθήκη_ν* **τότε**
εντολές_ν
αλλιώς
εντολές_αλλιώς
Τέλος_αν

Αν η συνθήκη_*k* είναι αληθής, εκτελούνται οι εντολές_*k* και η συνέχεια είναι η επόμενη εντολή από το Τέλος_αν. Εφόσον καμία συνθήκη δεν είναι αληθής, τότε εκτελούνται οι εντολές_αλλιώς. Οι εντολές_αλλιώς χρησιμοποιούνται κατά περίσταση.



Σημειώνεται ότι, αν σε μια πολλαπλή εντολή επιλογής υπάρχουν πάνω από μία συνθήκες αληθείς, τότε εκτελούνται οι εντολές που ανήκουν στην πρώτη αληθή συνθήκη κατά σειρά.

Παράδειγμα 2.13. Αριθμομηχανή

Να αναπτυχθεί αλγόριθμος, ο οποίος:
1. Θα διαβάζει πρώτα έναν αριθμό α, στη συνέχεια έναν από τους χαρακτήρες +, -, *, /, ανάλογα με την πράξη που θα εκτελέσει και τέλος έναν αριθμό β.
2. Θα εκτελεί την αντίστοιχη πράξη και θα τυπώνει το αποτέλεσμα. Σε περίπτωση που έχει επιλεγεί η πράξη της διαίρεσης, ο αλγόριθμος πρέπει να ελέγχει αν το β είναι μηδέν και τότε να τυπώνει το μήνυμα «Προσοχή, διαίρεση με το μηδέν» και να οδηγείται στο τέλος του.
3. Θα εκτυπώνει το μήνυμα «Λάθος πράξη», αν για το χαρακτήρα της πράξης δοθεί άλλο σύμβολο.

Αλγόριθμος Αριθμομηχανή
Διάβασε α, πράξη, β
Αν πράξη = "+" **τότε**
    Εμφάνισε α + β
αλλιώς_αν πράξη = "-" **τότε**
    Εμφάνισε α - β
αλλιώς_αν πράξη = "*" **τότε**
    Εμφάνισε α * β
αλλιώς_αν πράξη = "/" **τότε**
    Αν β ≠ 0 **τότε**
        Εμφάνισε α / β
    αλλιώς
        Εμφάνισε "Προσοχή, διαίρεση με το μηδέν"
    Τέλος_αν
αλλιώς
    Εμφάνισε "Λάθος πράξη"
Τέλος_αν
Τέλος Αριθμομηχανή

2.2.7.4 Δομή επανάληψης

Λίγοι αλγόριθμοι χρησιμοποιούν μόνο τις δομές ακολουθίας και επιλογής. Στα ρεαλιστικά προβλήματα χρειάζεται συνήθως μια σειρά εντολών να επαναληφθεί πολλές φορές. Άλλωστε σε τέτοια προβλήματα «αξίζει τον κόπο» να εκπονηθεί κάποιος αλγόριθμος και στη συνέχεια να υλοποιηθεί ένα αντίστοιχο πρόγραμμα υπολογιστή.

Οι επαναληπτικές διαδικασίες μπορεί να έχουν διάφορες μορφές και να εμπεριέχουν συνθήκες επιλογών, όπως αυτές που περιγράφηκαν στις προηγούμενες παραγράφους.

Παράδειγμα 2.14. Να εκπονηθεί αλγόριθμος ο οποίος με δεδομένο ένα θετικό ακέραιο αριθμό θα εμφανίζει τους ακέραιους αριθμούς από το 1 μέχρι και τον δεδομένο αριθμό Ν.

Οι ζητούμενοι αριθμοί μπορούν να παραχθούν με ένα συστηματικό τρόπο, αφού ο καθένας δημιουργείται από τον προηγούμενό του προσθέτοντας το 1. Με την αξιοποίηση αυτού του γεγονότος, ο αλγόριθμος δημιουργεί κάθε νέο αριθμό σε μια μεταβλητή, έστω i. Αυτό μπορεί να γίνει με τη χρήση της εντολής εκχώρησης: i ← i + 1.

Οπότε προκύπτει το ακόλουθο:

```
i ← 1
Εμφάνισε i     ! Εμφανίζεται το 1
i ← i + 1
Εμφάνισε i     ! Εμφανίζεται το 2
i ← i + 1
Εμφάνισε i     ! Εμφανίζεται το 3
…..
```

Το ζεύγος των εντολών i ← i + 1 και Εμφάνισε i επαναλαμβάνεται αυτούσιο. Αν μπορούσαν οι εντολές αυτές να γραφούν μία φορά και να εκτελεστούν Ν φορές, τότε το πρόβλημα θα λυνόταν. Αυτό επιτυγχάνεται με τις εντολές επανάληψης. Το πόσες φορές μπορούν να εκτελεστούν οι εντολές επανάληψης καθορίζεται με διαφορετικούς τρόπους. Στο παράδειγμα οι εντολές εκτελούνται όσο διάστημα η μεταβλητή i είναι μικρότερη ή ίση της μεταβλητής Ν.

Κατόπιν αυτών ο αλγόριθμος γίνεται

```
Αλγόριθμος Σειρά_αριθμών
Δεδομένα // N //
i ← 1
Όσο i ≤ N επανάλαβε
        Εμφάνισε i
        i ← i + 1
Τέλος_επανάληψης
Τέλος Σειρά_αριθμών
```

Συχνά η μεταβλητή i αποκαλείται *μετρητής*, επειδή αυξάνεται κατά 1. Σε άλλες περιπτώσεις όμως το βήμα αύξησης μπορεί να είναι οποιοδήποτε.

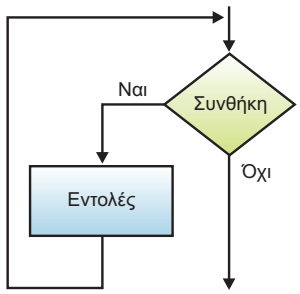
Παράδειγμα 2.15. Να γραφεί αλγόριθμος ο οποίος διαβάζει το όνομα ενός μαθητή, τους βαθμούς του σε τρία μαθήματα και υπολογίζει και τυπώνει το μέσο όρο του. Ο αλγόριθμος να σταματάει, όταν για όνομα μαθητή δοθεί το κενό εμφανίζοντας το πλήθος των μαθητών για τους οποίους υπολογίστηκε ο μέσος όρος.

```
Αλγόριθμος Μέσος_όρος
π ← 0
Διάβασε όνομα
Όσο όνομα ≠ " " επανάλαβε
```

Εντολή
Όσο ... επανάλαβε

```
Όσο Συνθήκη επανάλαβε
      Εντολές
Τέλος_επανάληψης
```

Εκτελούνται οι εντολές όσο η συνθήκη είναι αληθής



Εικόνα 2.22. Διάγραμμα ροής της Όσο...επανάλαβε

Η εντολή i ← i + 1 ***δεν είναι εξίσωση*** (γιατί και να ήταν δεν έχει λύση), αλλά δρα ως εξής: κάθε φορά που εκτελείται, το περιεχόμενο της μεταβλητής *i αυξάνεται κατά 1.*

Όλες οι εντολές επανάληψης μπορούν να πραγματοποιούν την εκτέλεση ενός συνόλου εντολών πάνω από μια φορά.

Στην εντολή Όσο… επανάλαβε οι εμπεριεχόμενες εντολές μπορεί να μην εκτελεστούν ποτέ, αφού η συνθήκη τερματισμού ελέγχεται στην αρχή.



Οι εντολές που συγκροτούν μια εντολή επανάληψης αποκαλούνται ***βρόχος*** (αγγλ. loop, γαλ. boucle).

Προσοχή: βρόχος, όχι βρόγχος. Βρόχος = θηλιά, βρόγχος = πνευμόνι.

```
        Διάβασε α, β, γ
        Εμφάνισε (α + β + γ) / 3
        π ← π + 1
        Διάβασε όνομα
Τέλος_επανάληψης
Εμφάνισε π
Τέλος Μέσος_όρος
```

Παράδειγμα 2.16. Σε ένα σουπερμάρκετ κάθε πελάτης δικαιούται μια δωροεπιταγή 6 € αν συμπληρώσει 200 πόντους. Να αναπτυχθεί αλγόριθμος ο οποίος θα διαβάζει τους πόντους που κερδίζει ένας συγκεκριμένος πελάτης σε κάθε επίσκεψη στο σουπερμάρκετ και θα εμφανίζει μετά από πόσες επισκέψεις παίρνει τη δωροεπιταγή και ποιος είναι ο μέσος όρος πόντων σε κάθε επίσκεψη.

```
Αλγόριθμος Παράδειγμα
Σ ← 0
π ← 0
Όσο Σ < 200 επανάλαβε
        Διάβασε πόντοι
        Σ ← Σ + πόντοι
        π ← π + 1
Τέλος_επανάληψης
ΜΟ ← Σ / π
Εμφάνισε π, ΜΟ
Τέλος Παράδειγμα
```

Παράδειγμα 2.17. Κατάλογος επιλογών

Πολύ συχνά στις εφαρμογές προβάλλεται στην οθόνη ένας κατάλογος από δυνατές επιλογές (menu) και στη συνέχεια ζητείται από το χρήστη να διαλέξει μία μόνο από αυτές. Στην πιο απλή περίπτωση, οι δυνατές επιλογές είναι αριθμημένες, οπότε απλά ζητείται η εισαγωγή ενός ακέραιου αριθμού.

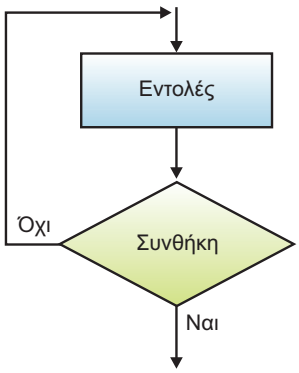
```
…
Επανάλαβε
        Εμφάνισε "1. Ενημέρωση"
        Εμφάνισε "2. Εκτύπωση"
        Εμφάνισε "3. Έξοδος"
        Εμφάνισε "Επιλογή:"
        Διάβασε Επιλογή
Μέχρις_ότου Επιλογή = 1 ή Επιλογή = 2 ή Επιλογή = 3
```

Στο παραπάνω τμήμα αλγορίθμου, ο βρόχος επαναλαμβάνεται μέχρι να δοθεί 1, 2 ή 3. Με τον τρόπο αυτό προστατεύεται το πρόγραμμα εφαρμογής από τυχόν λανθασμένη εισαγωγή τιμών από τον χρήστη. Ας σημειωθεί με την ευκαιρία, ότι τα προγράμματα εισαγωγής δεδομένων

Εντολή
Επανάλαβε ... Μέχρις_ότου

```
Επανάλαβε
        Εντολές
Μέχρις_ότου Συνθήκη
```

Εκτελούνται οι εντολές μέχρις ότου η συνθήκη γίνει αληθής.



Εικόνα 2.23. Διάγραμμα ροής της Επανάλαβε...Μέχρις_ότου

είναι συνήθως τα πιο δύσκολα και μακροσκελή λόγω της ανάγκης να είναι φιλικά προς το χρήστη.

Υποτίθεται ότι ανάλογα με την επιλογή, ο πιο πάνω αλγόριθμος διακλαδίζεται σε άλλα σημεία, όπου υπάρχουν οι εντολές υλοποίησης κάθε λειτουργίας.

Παράδειγμα 2.18. Να εκπονηθεί αλγόριθμος ο οποίος θα διαβάζει 100 αριθμούς και θα υπολογίζει και θα εμφανίζει το άθροισμά τους.

```
Αλγόριθμος Άθροισμα_Αριθμών
Σ ← 0
Για i από 1 μέχρι 100
        Διάβασε α
        Σ ← Σ + α
Τέλος_επανάληψης
Εμφάνισε "Άθροισμα:", Σ
Τέλος Άθροισμα_Αριθμών
```

Σε αυτό το παράδειγμα η εντολή **Για…από…μέχρι** περιλαμβάνει όλα τα απαραίτητα δεδομένα για την επανάληψη, δηλαδή αρχική τιμή της μεταβλητής i, τελική τιμή και το βήμα μεταβολής που είναι 1 και παραλείπεται. Ο βρόχος εκτελείται για όλες τις τιμές της μεταβλητής i.

Η εντολή εκχώρησης Σ ← Σ + α δεν είναι εξίσωση και καλύτερα να διαβάζεται ως «η νέα τιμή της μεταβλητής Σ είναι η παλιά συν α».

Συχνά η μεταβλητή Σ αποκαλείται αθροιστής, γιατί της εκχωρείται το τρέχον και τελικό άθροισμα των αριθμών και δεν πρέπει να λησμονείται ότι απαιτείται ο μηδενισμός του πριν από την έναρξη της επαναληπτικής διαδικασίας.

Η χρήση της εντολής **Για…από…μέχρι** γενικά προτιμάται όταν είναι γνωστός ο αριθμός των φορών που θα γίνει μια επανάληψη, με άλλα λόγια όταν είναι γνωστά τα τ1, τ2 και β.

Εμφωλευμένες εντολές επανάληψης

Σε όλες τις προηγούμενες περιπτώσεις όπου αναφέρεται εντολή ή εντολές, τίποτα δεν απαγορεύει αυτές οι εντολές να είναι επίσης εντολές επανάληψης. Αναφερόμαστε τότε σε εμφωλευμένες εντολές επανάληψης.

2.2.7.5 Κλήση αλγόριθμου από αλγόριθμο

Ένας αλγόριθμος μπορεί να κληθεί από έναν άλλο αλγόριθμο με χρήση της εντολής **Κάλεσε**.

Η επικοινωνία μεταξύ των δύο αλγορίθμων γίνεται ως εξής:

Εντολή Για ... από ... μέχρι

```
Για μεταβλητή από τ1 μέχρι τ2 [με_βήμα β]
        Εντολές
Τέλος_επανάληψης
```

Εκτελούνται οι εντολές με αρχική τιμή της *μεταβλητής* τ1 μέχρι και την τελική τιμή της *μεταβλητής* τ2.

Στη δομή αυτή τ1, τ2 είναι αριθμητικές σταθερές, μεταβλητές ή εκφράσεις. Πρέπει τ1 ≤ τ2, αν β > 0 και τ1 ≥ τ2, αν β < 0. Το βήμα β, αν είναι 1, παραλείπεται. Οι τιμές των τ1, τ2 και β μπορεί να είναι ακέραιες ή πραγματικές.

Αν τ1 > τ2 και β = 0 δεν θα εκτελεστούν οι εμπεριεχόμενες εντολές της Για, ενώ αν τ1 ≤ τ2 και β = 0 η εντολή επανάληψης θα εκτελείται άπειρες φορές (ατέρμονας βρόχος).



Γενικά οι εμφωλευμένες εντολές επιτρέπουν το συνδυασμό συνιστωσών ενός αλγορίθμου με οποιαδήποτε σειρά. Για παράδειγμα: Επανάληψη μέσα σε επιλογή, επανάληψη μέσα σε επανάληψη κ.α.

Η εντολή **Κάλεσε** συνοδεύεται με το όνομα του καλούμενου αλγορίθμου ακολουθούμενο πιθανά από λίστα μεταβλητών ή σταθερών μέσα σε παρενθέσεις. Οι τιμές των μεταβλητών ή σταθερών μεταβιβάζονται κατά την κλήση στις αντίστοιχες μεταβλητές της γραμμής Δεδομένα του καλούμενου αλγορίθμου. Όταν ο καλούμενος αλγόριθμος τερματίσει τη λειτουργία του, γίνεται επιστροφή στην αμέσως επόμενη εντολή της Κάλεσε. Κατά την επιστροφή μπορεί επίσης να μεταβιβάζονται τιμές της εντολής Αποτελέσματα.

Παράδειγμα 2.19.

```
Αλγόριθμος Καλών                    Αλγόριθμος Καλούμενος
Διάβασε α, β                        Δεδομένα // x, y //
Κάλεσε Καλούμενος (α, β, γ, δ)      z1 ← x + y
Γράψε "Άθροισμα:", γ                z2 ← x - y
Γράψε "Διαφορά:", δ                 Αποτελέσματα // z1, z2 //
Τέλος Καλών                         Τέλος Καλούμενος
```

Η αντιστοιχία των μεταβλητών των δύο αλγορίθμων γίνεται με τη σειρά που αναφέρονται στις αντίστοιχες γραμμές, δηλ. $\alpha \Rightarrow x$, $\beta \Rightarrow y$ και $\gamma \Leftarrow z1$, $\delta \Leftarrow z2$. Συνώνυμες μεταβλητές διαφορετικών αλγορίθμων δεν έχουν καμία σχέση μεταξύ τους.

Παράδειγμα 2.20. Υπολογισμός συνδυασμών

Να εκπονηθεί αλγόριθμος ο οποίος να υπολογίζει το πλήθος των συνδυασμών των Ν πραγμάτων ανά Κ. Ο συνδυασμός των Ν πραγμάτων ανά Κ συμβολίζεται με $\binom{n}{k}$ και δίνεται από τον τύπο $\frac{n!}{k!(n-k)!}$.

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

```
Αλγόριθμος Συνδυασμοί
Δεδομένα //n, k//
Κάλεσε Παραγοντικό(n, nfact)
Κάλεσε Παραγοντικό(k, kfact)
Κάλεσε Παραγοντικό(n - k, nkfact)
combin ← nfact / (kfact * nkfact)
Αποτελέσματα //combin//
Τέλος Συνδυασμοί
```

Οι συνδυασμοί απαντούν στο ερώτημα κατά πόσους διαφορετικούς τρόπους είναι δυνατό π.χ. να εξαχθούν 5 φύλλα από μια τράπουλα των 52 φύλλων.

Ο αλγόριθμος Παραγοντικό καλείται τρεις φορές από τον αλγόριθμο Συνδυασμοί. Σε κάθε κλήση μεταβιβάζεται η τιμή του αριθμού για τον οποίον ζητείται το παραγοντικό, στη μεταβλητή n του καλούμενου αλγόριθμου Παραγοντικό. Όταν ο τελευταίος τερματίσει, επιστρέφει το αποτέλεσμα nfact.

Οι μεταβλητές n και nfact του αλγόριθμου Συνδυασμοί δεν έχουν καμία σχέση με τις συνώνυμες μεταβλητές του αλγορίθμου Παραγοντικό.

Από τον τύπο των συνδυασμών φαίνεται ότι απαιτείται ο υπολογισμός του παραγοντικού τριών διαφορετικών ποσοτήτων. Για το σκοπό αυτό δημιουργείται ο αλγόριθμος Παραγοντικό, που υπολογίζει το παραγοντικό ενός αριθμού. Σε αυτόν τον αλγόριθμο με μια απλή επανάληψη βρίσκεται το παραγοντικό της μεταβλητής n στη μεταβλητή nfact. Ας προσεχθεί εδώ, ότι επειδή ζητείται γινόμενο, η αρχική τιμή του nfact είναι 1.

Με τον ίδιο τρόπο που καλείται ο αλγόριθμος Παραγοντικό από τον αλγόριθμο Συνδυασμοί, μπορεί να κληθεί και ο τελευταίος από έναν άλλο αλγόριθμο, όπως π.χ. τον επόμενο.

```
Αλγόριθμος Τράπουλα
Εμφάνισε "Αριθμός Φύλλων:"
Διάβασε fylla
deck ← 52
Κάλεσε Συνδυασμοί(deck, fylla, number)
Εμφάνισε "Οι συνδυασμοί", deck, "φύλλων ανά", fylla, "είναι:", number
Τέλος Τράπουλα
```

2.2.7.6 Αναδρομή

Πολλές επιστημονικές εφαρμογές χρησιμοποιούν συναρτήσεις ή σχέσεις γενικότερα που χρησιμοποιούν τις ίδιες στον ορισμό τους. Αυτές οι συναρτήσεις ή σχέσεις ονομάζονται ***αναδρομικές*** (Recursive).

Παράδειγμα 2.21. Ν Παραγοντικό

Είδαμε στην παράγραφο 2.2.4 τον αναδρομικό ορισμό του Ν!. Με βάση αυτόν τον ορισμό, παρατίθεται ο αναδρομικός αλγόριθμος υπολογισμού του Ν!.

```
Αλγόριθμος Ν_Παραγοντικό
Διάβασε Ν
Κάλεσε Factorial (N, N_Fact)
Εμφάνισε Ν, "!=", N_Fact
Τέλος Ν_Παραγοντικό1
```

```
Αλγόριθμος Factorial
Δεδομένα // n //
Αν n > 0 τότε
  Κάλεσε Factorial(n - 1, Fact)
  Fact ← Fact * n
αλλιώς
  Fact ← 1
Τέλος_αν
Αποτελέσματα // Fact //
Τέλος Factorial
```

Παράδειγμα 2.22. Μέγιστος Κοινός Διαιρέτης

Παρατίθεται εδώ η αναδρομική εκδοχή του αλγόριθμου του Ευκλείδη

```
Αλγόριθμος Ευκλείδης
Διάβασε α, β
Κάλεσε ΜΚΔ(α, β)
Τέλος Ευκλείδης
```

```
Αλγόριθμος ΜΚΔ
Δεδομένα // x, y //
Αν y = 0 τότε
  Εμφάνισε x
αλλιώς
  x ← x mod y
  Κάλεσε ΜΚΔ (y, x)
Τέλος_αν
Τέλος ΜΚΔ
```

2.2.8 Βασικές αλγοριθμικές λειτουργίες σε δομές δεδομένων

Όπως αναφέρθηκε ήδη, οι δομές δεδομένων διαθέτουν οργανωμένα δεδομένα, στα οποία μπορούν να γίνουν διάφορες επεξεργασίες. Οι βασικές λειτουργίες ή πράξεις επί των δομών δεδομένων είναι οι ακόλουθες:

- ✓ **Προσπέλαση** (access), πρόσβαση σε δεδομένα με σκοπό την ανάγνωση ή εγγραφή ή μετακίνηση.
- ✓ **Ανάκτηση** (retrieval), η με οποιονδήποτε τρόπο λήψη (ανάγνωση) του περιεχομένου ενός κόμβου.

Από τη μελέτη του παραδείγματος 2.20 αναδεικνύεται και ο σωστός τρόπος αντιμετώπισης προβλημάτων. Κάθε πρόβλημα διασπάται σε μικρότερα προβλήματα, τα οποία μπορούν να επιλυθούν πιο εύκολα. Ο αλγόριθμος επίλυσης ενός προβλήματος πρέπει να επικεντρώνεται στο πρόβλημα που επιλύει και να αδιαφορεί για το πού θα χρησιμοποιηθεί, καθώς και για την είσοδο και έξοδο των δεδομένων. Με τη χρήση των εντολών Δεδομένα και Αποτελέσματα επιτυγχάνεται πολύ εύκολα η μεταφορά των δεδομένων ενός προβλήματος από/ προς τον αλγόριθμο επίλυσης. Στα επόμενα θα γίνεται αποκλειστική χρήση των εντολών αυτών, εκτός από τις περιπτώσεις που η είσοδος δεδομένων και η έξοδος αποτελεσμάτων είναι το προς επίλυση πρόβλημα.



Στο παράδειγμα αυτό, ο αλγόριθμος *Ευκλείδης* καλεί τον αλγόριθμο ΜΚΔ μεταβιβάζοντάς του τις τιμές των μεταβλητών α και β στις μεταβλητές x και y αντίστοιχα. Ο αλγόριθμος ΜΚΔ με την εντολή Κάλεσε καλεί τον εαυτό του δίνοντας άλλες τιμές στις μεταβλητές x και y.

Στην πράξη σπάνια υπάρχουν όλες οι λειτουργίες σε μια δομή. Συνήθως παρατηρείται το φαινόμενο μια δομή δεδομένων να είναι αποδοτικότερη από μια άλλη για κάποια λειτουργία π.χ. την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, όπως π.χ. την εισαγωγή. Αυτό εξηγεί τόσο την ύπαρξη διαφορετικών δομών όσο και τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.

- ✓ **Αναζήτηση** (searching) ενός συνόλου στοιχείων δεδομένων προκειμένου να εντοπιστούν ένα ή περισσότερα στοιχεία, που έχουν μια δεδομένη ιδιότητα.
- ✓ **Εισαγωγή** (insertion), η προσθήκη ή δημιουργία νέων κόμβων σε μια υπάρχουσα δομή.
- ✓ **Μεταβολή** ή **τροποποίηση** (modification), η αλλαγή του περιεχομένου ενός κόμβου.
- ✓ **Διαγραφή** (deletion) ή **ακύρωση** που συνιστά το αντίθετο της εισαγωγής.
- ✓ **Ταξινόμηση** (sorting), όπου τα στοιχεία μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα τάξη.

Άλλες λειτουργίες είναι η ***συγχώνευση*** (merging), κατά την οποία δύο ή περισσότερες ταξινομημένες δομές συνενώνονται σε μια ενιαία δομή, η ***προσάρτηση*** (append), κατά την οποία μία δομή επικολλάται στο τέλος μιας άλλης, η ***αντιγραφή*** (copying), κ.ά..

Η πιο συνηθισμένη και απλή δομή δεδομένων είναι ο πίνακας. Οι πίνακες υποστηρίζονται από όλες σχεδόν τις γλώσσες προγραμματισμού. Αποτελούνται από ένα σύνολο ομοειδών απλών στοιχείων. Το μέγεθος ενός πίνακα, δηλαδή το πλήθος των στοιχείων που περιέχει, συνήθως είναι σταθερό και προκαθορισμένο. Η αναφορά σε ένα στοιχείο του πίνακα γίνεται με τη χρήση ενός συμβολικού ονόματος που έχει αποδοθεί στον πίνακα, μαζί με ένα ή περισσότερα στοιχεία που ονομάζονται δείκτες (indexes). Για παράδειγμα το πέμπτο στοιχείο του πίνακα Α συμβολίζεται με Α[5].



Οι πίνακες με ένα δείκτη λέγονται μονοδιάστατοι, οι πίνακες με 2 δείκτες δισδιάστατοι, ... οι πίνακες με ν δείκτες ν-διάστατοι.

Στη συνέχεια παρουσιάζονται χαρακτηριστικά παραδείγματα επεξεργασίας πινάκων.

Παράδειγμα 2.23. Εισαγωγή στοιχείων σε πίνακα

➢ **Είσοδος δεδομένων αγνώστου πλήθους**

Στο επόμενο τμήμα αλγορίθμου εισάγονται θετικοί αριθμοί στο μονοδιάστατο πίνακα Α. Δεν είναι γνωστός ο αριθμός των στοιχείων που θα εισαχθούν, αλλά έχει συμφωνηθεί ότι το τέλος της εισαγωγής θα καθοριστεί από την είσοδο ενός αρνητικού αριθμού (ή γενικότερα μιας τιμής που δεν μπορεί να ανήκει στο σύνολο τιμών του πίνακα).

Εδώ χρησιμοποιείται η εντολή επανάληψης Όσο συνθήκη επανάλαβε. Απαιτεί ένα αρχικό διάβασμα προκειμένου να ενεργοποιηθεί η συνθήκη συνέχισης της επανάληψης. Κάθε θετική τιμή που διαβάζεται καταχωρείται σε επόμενο στοιχείο του πίνακα με τη χρήση του δείκτη i. Στο τέλος φυλάσσεται στη μεταβλητή n η τρέχουσα τιμή του i, ώστε να είναι γνωστό το πλήθος των στοιχείων του πίνακα στη συνέχεια.

```
i ← 0
Διάβασε K
Όσο K ≥ 0 επανάλαβε
        i ← i + 1
        A[i] ← K
        Διάβασε K
Τέλος_επανάληψης
n ← i
```



Πριν από κάθε επεξεργασία πίνακα απαιτείται να εισαχθούν δεδομένα σε αυτόν. Η εισαγωγή δεδομένων σε πίνακα μπορεί να είναι επίπονη διαδικασία, ιδιαίτερα αν ο πίνακας είναι μεγάλος. Επί πλέον τα λάθη πληκτρολόγησης είναι πολύ συνηθισμένα και πρέπει να υπάρχει τρόπος διόρθωσης κάποιων τιμών.

➢ **Είσοδος δεδομένων γνωστού πλήθους**

Όταν είναι γνωστό το πλήθος n των τιμών μπορεί να χρησιμοποιηθεί η εντολή επανάληψης Για … από …μέχρι. Η πιο απλή εκδοχή είναι η επόμενη.

```
Για i από 1 μέχρι n
        Διάβασε A[i]
Τέλος_επανάληψης
```

Σε όλες τις προηγούμενες περιπτώσεις εισαγωγής τιμών σε πίνακα, ο χρήστης του σχετικού προγράμματος μπορεί να υποβοηθείται με την εμφάνιση κατάλληλων μηνυμάτων.

Παράδειγμα 2.24. Εκτύπωση πίνακα

Η εκτύπωση των στοιχείων του πίνακα, καθώς και της θέσης που υπάρχει το στοιχείο, επιτυγχάνεται με τις επόμενες εντολές:

```
Για i από 1 μέχρι n
        Εμφάνισε i, A[i]
Τέλος_επανάληψης
```

Παράδειγμα 2.25. Τροποποίηση στοιχείων πίνακα

Μετά την εισαγωγή στοιχείων και την εκτύπωση μπορεί να απαιτείται η διόρθωση ενός ή περισσοτέρων στοιχείων.

```
Διάβασε i
Όσο i > 0 και i <= n επανάλαβε
        Εμφάνισε A[i]
        Διάβασε A[i]
        Διάβασε i
Τέλος_επανάληψης
```

Στον αλγόριθμο αυτό η επανάληψη εκτελείται όσο δίνεται τιμή του i εντός ορίων. Κάθε φορά εμφανίζεται το στοιχείο A[i] και ζητείται η εισαγωγή μιας άλλης τιμής που την αντικαθιστά.

Μετά την εισαγωγή δεδομένων σε ένα πίνακα μπορεί να ακολουθήσει η επεξεργασία του. Στη συνέχεια παρουσιάζονται μερικές συνηθισμένες επεξεργασίες πινάκων.

Παράδειγμα 2.26. Άθροισμα στοιχείων πίνακα

Δίδεται ο μονοδιάστατος πίνακας Β που περιέχει Ν βαθμούς μαθητών. Να αναπτυχθεί αλγόριθμος, ο οποίος να υπολογίζει και να εμφανίζει το μέσο όρο βαθμολογίας των μαθητών.

```
Αλγόριθμος Βαθμολογία
Δεδομένα // Β, Ν //
Σ ← 0
Για i από 1 μέχρι Ν
        Σ ← Σ + Β[i]
Τέλος_επανάληψης
ΜΟ ← Σ / Ν
Εμφάνισε "Μ.Ο.:", ΜΟ
Τέλος Βαθμολογία
```

Με το βρόχο **Για** i **από** 1 **μέχρι** Ν διατρέχονται όλα τα στοιχεία του πίνακα και κάθε ένα αθροίζεται στη μεταβλητή Σ, η οποία έχει μηδενιστεί πριν από την έναρξη της επανάληψης. Ο Μέσος όρος είναι το πηλίκο του αθροίσματος όλων των στοιχείων δια του πλήθους των στοιχείων.

Για τη δημιουργία προγράμματος που να υλοποιεί τον παραπάνω αλγόριθμο και προκειμένου να γίνει ο έλεγχος ορθότητας, απαιτείται και η εισαγωγή κάποιων δεδομένων. Σύμφωνα με όσα αναφέρθηκαν στις προηγούμενες παραγράφους αυτό μπορεί να γίνει π.χ. με τον επόμενο αλγόριθμο.

```
Αλγόριθμος Επεξεργασία_Πίνακα
Διάβασε n
Για i από 1 μέχρι n
        Διάβασε Α[i]
Τέλος_επανάληψης
Κάλεσε Βαθμολογία (Α, n)
Τέλος Επεξεργασία_Πίνακα
```

Στις πρώτες γραμμές έχουν γραφεί οι σχετικές εντολές με τις οποίες γίνεται η εισαγωγή δεδομένων στο μονοδιάστατο πίνακα Α, ο οποίος έχει n στοιχεία. Στη συνέχεια προκειμένου να βρεθεί ο μέσος όρος των στοιχείων του Α, καλείται ο αλγόριθμος Βαθμολογία, που δημιουργήθηκε για το σκοπό αυτό. Κατά την κλήση μεταβιβάζονται στον πίνακα Β οι τιμές των στοιχείων του πίνακα Α και στη μεταβλητή Ν η τιμή της μεταβλητής n.

Παράδειγμα 2.27. Μέγιστο στοιχείο πίνακα

Δίδεται ο μονοδιάστατος πίνακας Β που περιέχει Ν βαθμούς μαθητών. Να αναπτυχθεί αλγόριθμος, ο οποίος να βρίσκει και να εμφανίζει την υψηλότερη βαθμολογία.

```
Αλγόριθμος Μέγιστο_πίνακα
Δεδομένα // Β, Ν //
max ← Β[1]
Για i από 2 μέχρι Ν
        Αν Β[i] > max τότε
                max ← Β[i]
        Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε "Μέγιστο:", max
Τέλος Μέγιστο_πίνακα
```

Στη μεταβλητή max εκχωρείται η τιμή του πρώτου στοιχείου του πίνακα. Στη συνέχεια με το βρόχο Για i από 2 μέχρι Ν εξετάζονται όλα τα υπόλοιπα στοιχεία του πίνακα. Για κάθε ένα, αν είναι μεγαλύτερο από το max, τότε αντικαθιστάται η τιμή του max με το νέο στοιχείο. Δηλαδή στη μεταβλητή max εκχωρείται το εκάστοτε μέγιστο στοιχείο, οπότε στο τέλος της επανάληψης θα έχει το μέγιστο στοιχείο του πίνακα.



Η επανάληψη θα μπορούσε να ξεκινά από 1.

Παράδειγμα 2.28. Επεξεργασία δισδιάστατου πίνακα

Να αναπτυχθεί αλγόριθμος ο οποίος με δεδομένο το πλήθος των μαθητών της Γ' γυμνασίου ενός σχολείου και τους βαθμούς των μαθητών στη γυμναστική, σε καθένα από τα 3 τρίμηνα, να υπολογίζει και να επιστρέφει το μέσο όρο βαθμολογίας κάθε μαθητή και το συνολικό μέσο όρο.

Ο πίνακας Β έχει Ν γραμμές και 3 στήλες. Σε κάθε μία γραμμή υπάρχουν οι βαθμολογίες των τριών τριμήνων για κάθε ένα μαθητή.

```
Αλγόριθμος Αθροίσματα
Δεδομένα // Β, Ν //
Σ ← 0
Για i από 1 μέχρι Ν
   ΣΜ ← 0
   Για j από 1 μέχρι 3
      ΣΜ ← ΣΜ + Β[i, j]
   Τέλος_επανάληψης
   ΜΟΒ[i]← ΣΜ / 3
   Σ ← Σ + ΣΜ
Τέλος_επανάληψης
ΜΟ ← Σ / (Ν * 3)
Αποτελέσματα // ΜΟΒ, ΜΟ //
Τέλος Αθροίσματα
```

Με το διπλό βρόχο Για i και Για j σαρώνονται κατά γραμμή όλα τα στοιχεία του πίνακα Β. Σε κάθε μία γραμμή, δηλαδή για κάθε μία τιμή του i, μηδενίζεται το ΣΜ (άθροισμα βαθμών μαθητή) και ακολούθως με τον εσωτερικό βρόχο αθροίζονται όλα τα στοιχεία της γραμμής αυτής. Όταν ολοκληρωθεί ο εσωτερικός βρόχος, στο ΜΟΒ[i] βρίσκεται ο μέσος όρος των στοιχείων της εκάστοτε γραμμής, δηλαδή ο μέσος όρος κάθε μαθητή στα τρία τρίμηνα. Το άθροισμα που υπολογίστηκε ανά γραμμή αξιοποιείται και αυξάνει τη μεταβλητή Σ κατά το άθροισμα γραμμής, η οποία στο τέλος θα έχει το άθροισμα όλων των στοιχείων του πίνακα.

Για την εισαγωγή δεδομένων σε ένα δισδιάστατο πίνακα Α που έχει Ν γραμμές και Μ στήλες, θα μπορούσε να αναπτυχθεί το ακόλουθο τμήμα αλγορίθμου:

```
Για i από 1 μέχρι Ν
  Για j από 1 μέχρι Μ
    Διάβασε Α[i, j]
  Τέλος_επανάληψης
Τέλος_επανάληψης
```

Ο αλγόριθμος αθροίζει τα στοιχεία κάθε γραμμής ενός δισδιάστατου πίνακα Ν γραμμών και 3 στηλών.



Ο μέσος όρος είναι το πηλίκο του αθροίσματος δια του πλήθους των στοιχείων του δισδιάστατου πίνακα. Το πλήθος των στοιχείων του είναι Ν*3.

Παράδειγμα 2.29. Σειριακή αναζήτηση

Σε ένα μονοδιάστατο πίνακα είναι καταχωρημένα τα ονόματα των σχολείων που συμμετέχουν σε έναν διαγωνισμό επιχειρηματικότητας και έχουν κατασκευάσει ένα χώρο παρουσίασης του επιχειρηματικού τους σχεδίου (που εν συντομία θα λέγεται περίπτερο). Ο αριθμός του περιπτέρου κάθε σχολείου είναι η θέση του σχολείου στον πίνακα. Δηλαδή στο πρώτο κελί του πίνακα είναι το όνομα του σχολείου που έχει το περίπτερο με αριθμό 1, στο δεύτερο κελί του πίνακα είναι το όνομα του σχολείου που έχει το περίπτερο με αριθμό 2 κ.ο.κ. Να αναπτυχθεί αλγόριθμος ο οποίος με δεδομένο το πλήθος των σχολείων που συμμετέχουν στο διαγωνισμό με περίπτερο και τον πίνακα με τα ονόματα των σχολείων, να διαβάζει το όνομα ενός σχολείου και να ψάχνει στον πίνακα, αν υπάρχει αυτό το σχολείο. Ο αλγόριθμος θα επιστρέφει το αποτέλεσμα της αναζήτησης, δηλαδή αν υπάρχει ή όχι το αναζητούμενο σχολείο, και αν υπάρχει τη θέση του στον πίνακα, αλλιώς ως θέση την τιμή μηδέν.

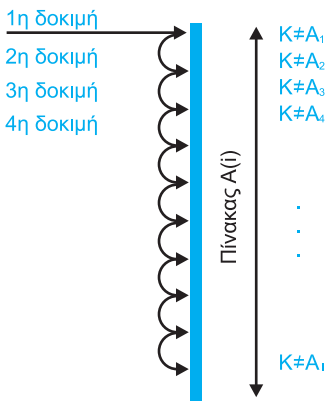
Κάθε σχολείο μπορεί να έχει ένα περίπτερο.

```
Αλγόριθμος Αναζήτηση
Δεδομένα // Α, Ν //
Εμφάνισε "Αναζητούμενο σχολείο:"
Διάβασε Κ
i ← 1
Βρέθηκε ← Ψευδής
Θέση ← 0
```

Στον αλγόριθμο είναι δεδομένος ο πίνακας Α και το πλήθος των στοιχείων του (Ν).

Η λογική μεταβλητή Βρέθηκε έχει την αρχική τιμή Ψευδής, όπου θεωρείται ότι δεν υπάρχει το ζητούμενο σχολείο και η μεταβλητή Θέση έχει την αρχική τιμή 0 για τον ίδιο λόγο.

Η επανάληψη του αλγορίθμου εκτελείται όσο δεν τελείωσε η σάρωση του πίνακα ($i \leq N$) και όσο δεν βρέθηκε το στοιχείο (Βρέθηκε = Ψευδής).



Εικόνα 2.24. Σειριακή αναζήτηση.



Η μέθοδος της ταξινόμησης με επιλογή βασίζεται στα ακόλουθα τρία βήματα:
α. επιλογή του στοιχείου με την ελάχιστη τιμή.
β. ανταλλαγή του με το πρώτο στοιχείο.
γ. επανάληψη των (α), (β) με τα υπόλοιπα στοιχεία.



Ο αλγόριθμος ταξινόμησης με επιλογή είναι πολυπλοκότητας $O(n^2)$

```
Όσο i ≤ N και Βρέθηκε = Ψευδής επανάλαβε
        Αν K = A[i] τότε
                Θέση ← i
                Βρέθηκε ← Αληθής
        αλλιώς
                i ← i + 1
        Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // Βρέθηκε, Θέση //
Τέλος Αναζήτηση
```

Τα αποτελέσματα είναι η μεταβλητή Βρέθηκε, η οποία, αν είναι αληθής σημαίνει ότι η αναζήτηση ήταν επιτυχής και η μεταβλητή Θέση, που έχει τη θέση του στοιχείου στον πίνακα, εφ' όσον βρέθηκε, αλλιώς θα έχουν την τιμή Ψευδής και μηδέν αντίστοιχα. Η τιμή μηδέν δεν μπορεί να είναι θέση πίνακα.

Παράδειγμα 2.30. Ταξινόμηση με επιλογή

Να αναπτυχθεί αλγόριθμος ο οποίος με δεδομένο το πλήθος των μαθητών της Γ' γυμνασίου ενός σχολείου και τον τελικό βαθμό του κάθε μαθητή στο μάθημα «Πληροφορική», να ταξινομεί τον πίνακα των βαθμών σε αύξουσα τάξη, από το μικρότερο στο μεγαλύτερο βαθμό. Στη συνέχεια να επιστρέφει τον ταξινομημένο πίνακα.

```
Αλγόριθμος Ταξινόμηση_με_επιλογή
Δεδομένα // A, N //
Για i από 1 μέχρι N
        j ← i
        min ← A[j]
        Για k από i + 1 μέχρι N
                Αν A[k] < min τότε
                        j ← k
                        min ← A[j]
                Τέλος_αν
        Τέλος_επανάληψης
        temp ← A[i]
        A[i] ← A[j]
        A[j] ← temp
Τέλος_επανάληψης
Αποτελέσματα // A //
Τέλος Ταξινόμηση_με_επιλογή
```

2.2.9 Εκσφαλμάτωση σε λογικά λάθη

Ένας αλγόριθμος χρειάζεται να δοκιμαστεί σε διαφορετικές συνθήκες και με ποικιλία δεδομένων ώστε να συγκριθούν τα παραγόμενα αποτελέσματα με τα αναμενόμενα, και να προβλεφτούν απρόσμενες καταστάσεις λάθους.

Ως **εκσφαλμάτωση** των λογικών λαθών ενός αλγορίθμου προσδιορίζεται η διαδικασία εύρεσης των λογικών λαθών που υπάρχουν σε αυτόν.

Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη διαδικασία ελέγχου και την ανάλυση των αποτελεσμάτων του. Ένα λογικό λάθος είναι ένα λάθος που, ενώ εκτελείται ο αλγόριθμος, τα αποτελέσματά του δεν είναι σωστά. Αυτό μπορεί να οφείλεται είτε σε λανθασμένη προσέγγιση για το πώς θα λυθεί το πρόβλημα, είτε σε λανθασμένη υλοποίηση της προσέγγισης που επιλέχθηκε. Κατά γενική ομολογία, τα λογικά λάθη είναι δύσκολο να εντοπιστούν.

Παράδειγμα 2.31. Λανθασμένος αλγόριθμος

Με σκοπό να αναπτυχθεί αλγόριθμος ο οποίος με δεδομένες τις τιμές δύο μεταβλητών, θα αντιμεταθέτει το περιεχόμενο των δύο μεταβλητών και θα επιστρέφει ως αποτέλεσμα το περιεχόμενο των μεταβλητών μετά την αντιμετάθεση γράφτηκε ο ακόλουθος λανθασμένος αλγόριθμος σε ψευδογλώσσα. Ζητείται να εντοπιστούν τα λογικά λάθη.

1. **Αλγόριθμος** Αντιμετάθεση
2. **Δεδομένα** // α, β //
3. α ← β
4. β ← α
5. **Αποτελέσματα** // α, β //
6. **Τέλος** Αντιμετάθεση

Απάντηση

Το πρώτο βήμα είναι η εκτέλεση του αλγορίθμου με πίνακα παρακολούθησης τιμών για να ελεγχθεί η λειτουργία του.

Αριθμός Εντολής	α	β	Έξοδος
2	8	12	
3	12		
4		12	
5			12 12

Από την εκτέλεση φαίνεται ότι εκτυπώνει σωστά την τιμή της μεταβλητής α, αλλά όχι της β. Αφού διαπιστωθεί το λάθος χρειάζεται να προσδιοριστεί η απαιτούμενη διόρθωση. Από τον αλγόριθμο απουσιάζει η μεταβλητή στην οποία θα εκχωρούταν προσωρινά η τιμή μίας εκ των δύο μεταβλητών.

Το πρώτο βήμα στην εύρεση των λογικών λαθών είναι να γίνει προσπάθεια εντοπισμού των πιθανών πηγών τους. Για να επιτευχθεί αυτό χρειάζεται να δώσετε προσοχή στα αποτελέσματα που έχει ως έξοδο ο αλγόριθμος και να σκεφτείτε πιθανές αιτίες που μπορεί να οδήγησαν σε τέτοιου είδους αποτελέσματα.

Η εκτέλεση του αλγορίθμου με το χέρι είναι πολύ χρήσιμη διαδικασία για τον εντοπισμό των λογικών λαθών.

Για το λόγο αυτό χρειάζεται να δοκιμάζετε εξονυχιστικά τον αλγόριθμό σας με διάφορα παραδείγματα, απλά, σύνθετα και ασυνήθιστα, για να βεβαιωθείτε ότι είναι σωστός.

Αν κάποιο από τα παραδείγματα δεν δίνει το αναμενόμενο αποτέλεσμα τότε υπάρχει παράλειψη στον κώδικα και δεν έχει καλυφθεί η σχετική περίπτωση.

Μόλις βρείτε τι φταίει, εμπλουτίζετε την αρχική σας σχεδίαση ώστε να καλύψετε και αυτό το είδος περιπτώσεων και κάνετε τις απαραίτητες προσθήκες.

Η διαδικασία εκσφαλμάτωσης περιλαμβάνει:

- Τη διαπίστωση του είδους του λάθους.
- Την ανεύρεση του ανεξάρτητου τμήματος του αλγορίθμου που εκτελεί τη λανθασμένη λειτουργία.
- Την ανεύρεση του λάθους μέσα σε αυτό το ανεξάρτητο τμήμα του αλγορίθμου.

Οι λόγοι τεκμηρίωσης είναι:

- η ευκολία ανάπτυξης αλγορίθμου σε περίπτωση που ζητείται αλγόριθμος με παρόμοιες λειτουργίες με υπάρχοντα αλγόριθμο.
- Η ευκολία στις τροποποιήσεις ενός υπάρχοντος αλγορίθμου.
- Ο περιορισμός των ελέγχων μόνο σε συγκεκριμένα σημεία, αφού για τα υπόλοιπα θα υπάρχει αρχείο δοκιμής μαζί με αποτελέσματα και συνοδευτικά σχόλια.



Λέξεις κλειδιά
Αλγόριθμος, Χαρακτηριστικά αλγορίθμου, Ψευδογλώσσα, Διάγραμμα ροής, Μεταβλητές, Εκφράσεις, Εντολή εκχώρησης, Δομή ακολουθίας, Δομή επιλογής, Δομή επανάληψης, Δομές δεδομένων, Πίνακας, Ουρά, Στοίβα, Γράφος, Δένδρο, Αναζήτηση, Ταξινόμηση, Εκσφαλμάτωση, Τεκμηρίωση.



2.2.10 Τεκμηρίωση

Η τεκμηρίωση δεν αποτελεί μια ξεχωριστή φάση στην ανάπτυξη ενός αλγορίθμου, αλλά μία παράλληλη διαδικασία που συμπληρώνει όλα τα στάδια ανάπτυξής του.

Με τον όρο **τεκμηρίωση** εννοείται το σύνολο του γραπτού υλικού που περιγράφει τα συστατικά μέρη και τις λειτουργίες του αλγορίθμου ή τις τροποποιήσεις που έγιναν σε έναν αλγόριθμο.

Η τεκμηρίωση μπορεί να διακριθεί:

- Τεκμηρίωση ανάπτυξης. Αναφέρεται στις προδιαγραφές του προβλήματος, τι πρόκειται να κάνει ο αλγόριθμος και τη σύνδεσή του με άλλους αλγορίθμους.
- Τεκμηρίωση ελέγχου. Αναφέρεται στα δεδομένα ελέγχου που χρησιμοποιήθηκαν προκειμένου να δοκιμαστεί ο αλγόριθμος, αν έχουν διερευνηθεί οι ακραίες περιπτώσεις και ποιες είναι αυτές και τέλος αν έχουν καθιερωθεί κάποιες και ποιες συμβάσεις για τη λύση του προβλήματος.
- Τεκμηρίωση αλγορίθμου. Αναφέρεται στον τρόπο με τον οποίο έχουν λυθεί τα επιμέρους προβλήματα, τις υποθέσεις που έχουν γίνει και τους περιορισμούς που έχουν τεθεί. Ακόμα πληροφορίες για «ειδικές» τεχνικές που έχουν χρησιμοποιηθεί. Πρέπει σε κάθε βήμα να αναφερθεί με λεπτομέρεια, πολλές φορές κουραστική, αλλά δυστυχώς αναγκαία, η λύση που χρησιμοποιήθηκε και τα δεδομένα με τα οποία ελέγχθηκε. Ένα υπόμνημα των μεταβλητών που χρησιμοποιούνται είναι χρήσιμο για την τεκμηρίωση αλλά και για την αρχική ανάπτυξη του αλγόριθμου.

  Η απλούστερη και πλέον στοιχειώδης τεκμηρίωση αυτής της μορφής γίνεται με την εισαγωγή γραμμών σχολίων μέσα στον αλγόριθμο.

Ανακεφαλαίωση

Στις παραγράφους του κεφαλαίου παρουσιάστηκαν η έννοια του αλγορίθμου μαζί με τα χαρακτηριστικά που χρειάζεται να έχει, οι τρόποι παράστασης αλγορίθμων, τα συστατικά τους μέρη, οι δομές δεδομένων και διατυπώθηκαν βασικά στοιχεία ανάλυσης και θεωρίας αλγορίθμων. Επιπλέον, παρουσιάστηκε η ψευδογλώσσα ως βασικότερος τρόπος ανάπτυξης αλγορίθμων, προκειμένου να υπάρχει ανεξαρτησία από τις γλώσσες προγραμματισμού. Παρουσιάστηκαν πολλά παραδείγματα απλών αλγορίθμων, ενώ αναπτύχθηκαν αλγόριθμοι αναζήτησης και ταξινόμησης. Τέλος, στο κεφάλαιο αυτό, επιχειρήθηκε να αναδειχθεί ότι οι αλγόριθμοι και οι δομές δεδομένων έχουν ισχυρότατη σχέση και αποτελούν τα βασικά συστατικά για την ανάπτυξη προγραμμάτων.

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Τι είναι αλγόριθμος; Καταγράψτε ή συζητήστε με τους συμμαθητές σας έναν αλγόριθμο.

2. Μέσα από παιχνίδι ρόλων να υλοποιήσετε και να εκτελέσετε τον αλγόριθμο του παραδείγματος 2.5 σειριακά και παράλληλα.

3. Με ποιους τρόπους γίνεται η αναπαράσταση αλγορίθμων;

4. Ποια η διαφορά δεδομένου και πληροφορίας; Να δώσετε ένα παράδειγμα.

5. Ποιοι είναι οι συνηθέστεροι τύποι δεδομένων; Ποιος τύπος λαμβάνει τις λιγότερες τιμές;

6. Τι είναι η δομή δεδομένων;

7. Ποιες δομές δεδομένων είναι μη γραμμικές;

8. Ποια είναι τα είδη των τελεστών;

9. Ποια είναι η χρήση της δομής ακολουθίας; Να δώσετε έναν αλγόριθμο σε ψευδογλώσσα με τη χρήση της δομής ακολουθίας.

10. Να περιγράψετε τη λειτουργία των τριών εντολών επιλογής.

11. Να περιγράψετε τη λειτουργία των τριών εντολών επανάληψης.

12. Να εξηγήσετε πότε δεν μπορεί να χρησιμοποιηθεί η εντολή επανάληψης Για.

13. Ποιες είναι οι λειτουργίες ή πράξεις επί των δομών δεδομένων;

14. Να περιγράψετε τον αλγόριθμο ταξινόμησης επιλογής σε πίνακα.

15. Σημειώστε τις σωστές απαντήσεις

Α. Ποια από τα παρακάτω αποτελούν αριθμητική σταθερά;

i. 2009 **ii.** "2009"
iii. Εμφάνισε **iv.** "Αλγόριθμος"

Β. Η λογική πράξη ή μεταξύ δύο προτάσεων είναι αληθής όταν:

i. Οποιαδήποτε από τις προτάσεις είναι αληθής
ii. Η πρώτη πρόταση είναι αληθής
iii. Η πρώτη πρόταση είναι ψευδής
iv. Και οι δύο προτάσεις είναι αληθείς

Γ. Αν Α και Β ακέραιες μεταβλητές, ποιες από τις παρακάτω εκφράσεις είναι αριθμητικές;

i. A + B ^ 2 **ii.** A > B + 3
iii. A div 3 ^ B **iv.** A και B > 3

16. Να μετατρέψετε σε εντολές εκχώρησης τις παρακάτω φράσεις:

Α. Η μεταβλητή α έχει διπλάσια τιμή από τη μεταβλητή β

Β. Η μεταβλητή ΜΟ είναι ο μέσος όρος των α, β, γ

Γ. Η μεταβλητή β αυξάνεται κατά 2

Δ. Η μεταβλητή i μειώνεται κατά α και β

Ε. Η μεταβλητή i είναι το μισό του αθροίσματος των α και β

17. Να επιλέξετε την τιμή της y που είναι αποτέλεσμα κάθε εντολής.

Εντολές εκχώρησης	Τιμή της y
i. y←(A_M(7 / 2) mod 3) + 1	
ii. y←10 div 9 - 10 mod 9	**Α.** 1
iii. y ←10 mod 2 + A_T(2 - 3)	**Β.** 0
iv. y ←A_M(9 / 2 / 3)	

18. Αντιστοιχίστε τις εκφράσεις της στήλης Α με τις λογικές σταθερές της στήλης Β με δεδομένο ότι α = 10, β = 5 και γ = 3.

Στήλη Α	Στήλη Β
i. α ≠ β **και** (γ - β) < 0	
ii. (α > β **ή** α > γ) **και** (γ > β)	
iii. α > β **ή** α > γ **και** γ > β	**Α.** Αληθής
iv. **όχι** (α > β) **ή** γ ≤ β	**Β.** Ψευδής
v. α > β **ή** (β - γ) > 0 **και** α < 0	
vi. α > β **ή** (β + 3) < γ **και** α < γ	

19. Τι εμφανίζουν τα επόμενα τμήματα αλγορίθμων;

```
S ← 0
i ← 1
Όσο i ≤ 9 επανάλαβε
    i ← i + 2
    S ← S + i
Τέλος_επανάληψης
Εμφάνισε S
```

```
S ← 0
i ← 1
Όσο i ≤ 9 επανάλαβε
    S ← S + i
    i ← i + 2
Τέλος_επανάληψης
Εμφάνισε S
```

20. Ο νόμος του Νεύτωνα για τη βαρύτητα λέει ότι κάθε σώμα στο σύμπαν έλκει κάθε άλλο σώμα με δύναμη που δίνεται από τον τύπο

$F = G \cdot \frac{m1 \cdot m2}{r^2}$, όπου m1 και m2 είναι οι μάζες των δύο σωμάτων (σε κιλά), r η απόσταση μεταξύ τους (σε μέτρα) και G είναι η παγκόσμια βαρυτική σταθερά. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει τις δύο μάζες, την απόσταση μεταξύ τους και θα υπολογίζει και θα εκτυπώνει τη δύναμη. Δίνεται ότι $G = 6,67 \times 10^{-11}$ N m^2 kg^{-2}.

21. Στην περίοδο των εκπτώσεων αγοράσατε ένα ποδήλατο με έκπτωση 25%. Το ποσό που δώσατε για το ποδήλατο ήταν 100 ευρώ. Να αναπτύξετε αλγόριθμο ο οποίος θα υπολογίζει την αρχική τιμή του ποδηλάτου.

22. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει έναν αριθμό και θα υπολογίζει και θα εμφανίζει το γινόμενο αυτού του αριθμού επί το τελευταίο ψηφίο του. Θεωρήστε ότι ο αριθμός είναι θετικός και ακέραιος.

23. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει έναν πραγματικό αριθμό με 2 δεκαδικά ψηφία και θα τον στρογγυλοποιεί στον πλησιέστερο ακέραιο. Για παράδειγμα, αν διαβαστεί ο αριθμός 4,23, να εμφανίζει 4, ενώ αν είναι ο 4,70 να εμφανίζει 5.

24. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει έναν ακέραιο αριθμό και θα υπολογίζει και θα εμφανίζει τον επόμενο άρτιο.

25. Σε έναν λογαριασμό τραπέζης παρέχεται κλιμακωτά το ακόλουθο επιτόκιο:

Ποσό	Επιτόκιο
≤ 5.000	1,8% το έτος
> 5.000	1,5% το έτος

Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει το ποσό χρημάτων που έχει ο λογαριασμός και θα υπολογίζει και θα εμφανίζει τον τόκο που θα λάβει μετά από ένα έτος, καθώς και το συνολικό ποσό χρημάτων μαζί με τον τόκο.

26. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει το τρέχον έτος και αν αυτό είναι από το 2001 μέχρι και το 2099 να εμφανίζει το μήνυμα «21ος αιώνας». Αν το έτος είναι από το 2002 και πάνω, να εμφανίζει το μήνυμα «Χρήση του €».

27. Ένα επιστημονικό σωματείο έχει 1.200 μέλη. Η γενική συνέλευση του σωματείου είναι σε απαρτία όταν είναι παρόν το 1/3 των μελών του. Για να υπερψηφιστεί μια πρόταση, θα πρέπει περισσότεροι από το 1/2 των παρόντων μελών να ψηφίσουν υπέρ. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει τον αριθμό των παρόντων μελών και αν ο αριθμός επιτρέπει την πραγματοποίηση της ψηφοφορίας, θα διαβάζει τον αριθμό αυτών που ψήφισαν υπέρ της πρότασης και θα εμφανίζει το αποτέλεσμα της ψηφοφορίας, δηλαδή αν υπερψηφίστηκε, αν καταψηφίστηκε ή αν δεν μπορεί να ψηφιστεί.

28. Ένας συνδρομητής μιας εταιρείας κινητής τηλεφωνίας έχει επιλέξει ένα πρόγραμμα με πάγιο 50 ευρώ τον μήνα. Στο πρόγραμμα δικαιούται τις ακόλουθες παροχές:

Παροχές	Πλήθος
Λεπτά ομιλίας/μήνα	1.000
SMS/μήνα	1.000
MB/μήνα	1.000

Ωστόσο, αν ξεπεράσει τον αριθμό 1.000 σε κάποια από τις παραπάνω παροχές, τότε χρεώνεται ως εξής για κάθε παροχή που ξεπερνάει τα 1.000:

Επιπλέον	Πλήθος
Κλήσεις ομιλίας	0,0055 €/δευτερόλεπτο
SMS	0,08 €/SMS
MB	0,05 €/MB

Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει τα λεπτά ομιλίας, το πλήθος των SMS, το πλήθος των MB και ανάλογα θα εμφανίζει τη μηνιαία χρέωση του καταναλωτή.

29. Να αναπτύξετε αλγόριθμο ο οποίος θα εκτυπώνει τις τιμές της συνάρτησης $f(x)=4\log(5+e^{3x+2})$ όταν το x παίρνει τις τιμές στο διάστημα [10, 50] με βήμα 0,5.

30. Να αναπτύξετε αλγόριθμο ο οποίος θα εμφανίζει όλους τους τριψήφιους αριθμούς που το άθροισμα των ψηφίων τους είναι μεγαλύτερο ή ίσο του 12.

31. Να αναπτύξετε αλγόριθμο ο οποίος θα υπολογίζει το ημίτονο ενός αριθμού x με βάση

τον επόμενο τύπο

$$\text{ημ}(x) = \sum_{n=0}^{49} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \quad \text{ή}$$

$$\text{ημ}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots - \frac{x^{49}}{49!}$$

32. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει γράμματα μέχρι να βρει τρεις φορές το γράμμα Α. Όταν σταματήσει το διάβασμα γραμμάτων, ο αλγόριθμος θα εκτυπώνει πόσα συνολικά γράμματα διαβάστηκαν.

33. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει αριθμούς, μέχρι να διαβαστεί ο αριθμός μηδέν. Ο αλγόριθμος θα εκτυπώνει το άθροισμα και το πλήθος των αριθμών που δόθηκαν και ήταν μεγαλύτεροι του 50.

34. Ένα ψηφιακό φωτογραφικό άλμπουμ έχει αποθηκευτικό χώρο N Mbytes. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει το μέγεθος της κάθε φωτογραφίας που επιχειρείται να αποθηκευτεί στο άλμπουμ, μέχρι το άλμπουμ να μη χωράει άλλη φωτογραφία. Ο αλγόριθμος θα επαναλαμβάνεται και θα σταματά αν το μέγεθος της φωτογραφίας που προσπαθεί κάποιος να αποθηκεύσει είναι μεγαλύτερο από τον διαθέσιμο χώρο του άλμπουμ. Όταν η εισαγωγή φωτογραφιών σταματήσει, ο αλγόριθμος θα εκτυπώνει το μήνυμα «Δεν χωράει». Στην περίπτωση που περίσσεψε χώρος να τον εκτυπώνει. Τέλος, να εκτυπώνει το πλήθος των φωτογραφιών που αποθηκεύτηκαν.

35. Τροποποιείστε τον αλγόριθμο του παραδείγματος 2.26, ώστε να βρίσκει το γινόμενο των στοιχείων του πίνακα.

36. Να αναπτύξετε αλγόριθμο ο οποίος:

- **Α.** Θα διαβάζει το πλήθος των μαθητών ενός λυκείου.
- **Β.** Θα διαβάζει το μέσο όρο βαθμολογίας κάθε μαθητή ο οποίος θα εισάγεται σε πίνακα. Ο μέσος όρος βαθμολογίας κάθε μαθητή θα πρέπει να ελέγχεται ώστε να είναι μεγαλύτερος ή ίσος του (ένα) 1 και μικρότερος ή ίσος του είκοσι (20).
- **Γ.** Θα εμφανίζει σε ποια θέση βρίσκεται και ποιος είναι ο μικρότερος μέσος όρος. Θεωρήστε ότι είναι μοναδικός.
- **Δ.** Θα εμφανίζει το μέσο όρο όλων των μαθητών.
- **Ε.** Θα εμφανίζει με κατάλληλα μηνύματα το ποσοστό των μαθητών με μέσο όρο μεγαλύτερο του 18.

37. Ενοποιείστε τους αλγορίθμους των παραδειγμάτων 2.23, 2.24 και 2.25 σε έναν ενιαίο αλγόριθμο, στον οποίο να προτάξετε ένα μενού επιλογής μιας λειτουργίας. (βλ. παρ. 2.17).

38. Στο πλαίσιο των εικονικών μαθητικών επιχειρήσεων, να αναπτύξετε αλγόριθμο ο οποίος:

- **Α.** Θα διαβάζει και θα αποθηκεύει σε πίνακα τις εισπράξεις μιας επιχείρησης για κάθε μήνα ενός σχολικού έτους.
- **Β.** θα ταξινομεί τον πίνακα των εισπράξεων σε φθίνουσα σειρά και θα εμφανίζει τον ταξινομημένο πίνακα.
- **Γ.** θα εμφανίζει τη μικρότερη και τη μεγαλύτερη είσπραξη της επιχείρησης.

39. Τροποποιείστε τον αλγόριθμο του παραδείγματος 2.28, ώστε να βρίσκει το μέσο όρο κάθε μαθήματος (υποδ. Απαιτείται ο υπολογισμός αθροισμάτων κατά στήλη).

40. Ο επόμενος αλγόριθμος εκτελεί σειριακή αναζήτηση του στοιχείου Κ στον πίνακα Α. Εντοπίστε τις διαφορές με τον αλγόριθμο του παραδείγματος 2.29. Ποιος είναι προτιμότερος και γιατί; Να λάβετε υπόψη την περίπτωση επιτυχημένης και αποτυχημένης αναζήτησης.

```
Αλγόριθμος Αναζήτηση2
Δεδομένα // Α, Ν, Κ //
Βρέθηκε ← Ψευδής
Θέση ← 0
Για  i από 1 μέχρι Ν
    Αν Κ = Α[i] τότε
            Θέση ← i
            Βρέθηκε ← Αληθής
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // Βρέθηκε, Θέση //
Τέλος Αναζήτηση2
```

41. Στον αλγόριθμο του παραδείγματος 2.29 είναι γνωστό από την εκφώνηση, ότι δεν μπο-

ρεί να υπάρχει στον πίνακα δεύτερη φορά το ίδιο σχολείο. Αν σε έναν πίνακα δεν είναι γνωστό αν υπάρχουν ή όχι επαναλήψεις των ίδιων στοιχείων, πώς πρέπει να αντιμετωπιστεί το πρόβλημα της αναζήτησης; Δώστε ένα σχετικό αλγόριθμο.

42. Αν ο πίνακας Α του παραδείγματος 2.29 ήταν ταξινομημένος, μήπως αυτή η επί πλέον γνώση μπορεί να επιφέρει κάποια βελτίωση στον αλγόριθμο σειριακής αναζήτησης; Αν ναι, εκπονήστε τον τροποποιημένο αλγόριθμο.

43. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει τις δικαιολογημένες και τις αδικαιολόγητες απουσίες ενός μαθητή καθώς και τον μέσο όρο του στα προφορικά και θα καλεί αλγόριθμο ο οποίος θα δέχεται τα παραπάνω στοιχεία και θα επιστρέφει το μήνυμα:

- «Έχει δικαίωμα εξέτασης τον Ιούνιο», αν ο μαθητής έχει μέχρι 64 απουσίες ή έχει μέχρι 114 απουσίες από τις οποίες οι αδικαιολόγητες δεν ξεπερνούν τις 64 ή έχει μέχρι 164 απουσίες από τις οποίες οι αδικαιολόγητες δεν ξεπερνούν τις 64 και ο μέσος όρος του στα προφορικά είναι πάνω από 15.
- «Έχει δικαίωμα εξέτασης τον Σεπτέμβριο», αν ο μαθητής έχει πάνω από 64 και μέχρι 114 απουσίες και οι αδικαιολόγητες ξεπερνούν τις 64 ή έχει πάνω από 114 και μέχρι 164 απουσίες από τις οποίες οι αδικαιολόγητες δεν ξεπερνούν τις 64 αλλά ο μέσος όρος του στα προφορικά δεν είναι πάνω από 15.
- «Επανάληψη χρονιάς», σε κάθε άλλη περίπτωση.

Ο καλών αλγόριθμος θα εκτυπώνει το μήνυμα.

44. Να αναπτύξετε αλγόριθμο ο οποίος θα διαβάζει έναν ακέραιο θετικό αριθμό, και θα καλεί αλγόριθμο ο οποίος θα υπολογίζει το πλήθος των ψηφίων του αριθμού. Ο καλούμενος αλγόριθμος, θα επιστρέφει το πλήθος των ψηφίων του αριθμού. Ο καλών αλγόριθμος θα εκτυπώνει το πλήθος.

45. Να αναπτύξετε αλγόριθμο ο οποίος:

Α. Θα διαβάζει και θα καταχωρίζει σε έναν πίνακα 1000 στοιχείων ονόματα.

Β. Θα διαβάζει ένα ζητούμενο όνομα.

Γ. Θα καλεί αλγόριθμο, ο οποίος θα αναζητά στον πίνακα το όνομα που διαβάστηκε στο προηγούμενο ερώτημα και θα επιστρέφει το πλήθος των ατόμων που υπάρχουν με το όνομα αυτό. Ο καλών αλγόριθμος θα εκτυπώνει το πλήθος.

46. Να εκτελέσετε το παρακάτω τμήμα αλγορίθμου

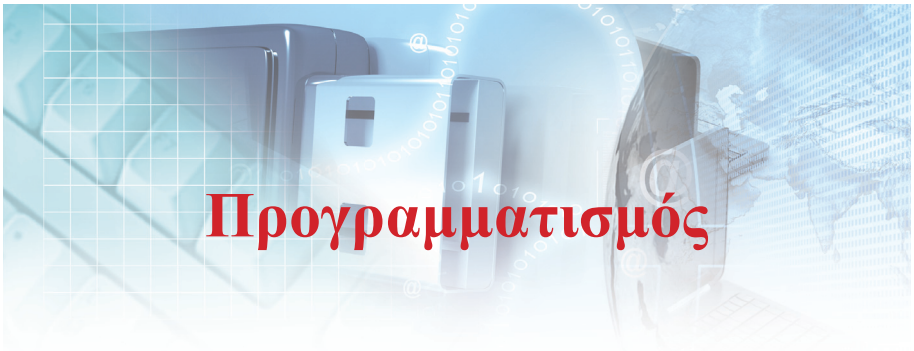
```
x ← 20
Σ ← 0
Για α από 1 μέχρι 10 με_βήμα 3
        x ← x - α
        Σ ← Σ + x
        Αν Σ mod 2 = 0 τότε
                x ← x + 1
        Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // x //
```

47. Το ακόλουθο τμήμα αλγορίθμου αναπτύχθηκε για να υπολογίζει το άθροισμα 5 αριθμών. Ωστόσο περιέχει λογικά λάθη. Να εντοπίσετε τα λογικά λάθη και να τον διορθώσετε.

```
Διάβασε x
Σ ← 0
Για i από 0 μέχρι 5
        Διάβασε x
        Σ ← Σ + x
Τέλος_επανάληψης
Εμφάνισε Σ + x
```

48. Το ακόλουθο τμήμα αλγορίθμου αναπτύχθηκε για να ελέγχει αν ένας αριθμός είναι θετικός. Ωστόσο περιέχει λογικό λάθος. Να το εντοπίσετε.

```
Επανάλαβε
        Διάβασε α
Μέχρις_ότου α < 0
```



Προγραμματισμός

Στόχοι του κεφαλαίου αυτού είναι να μπορούν οι μαθητές:

- ✓ να δημιουργούν ευκρινές γνωσιακό και οργανωμένο νοητικό σχήμα που να περιλαμβάνει τα είδη και τεχνικές προγραμματισμού, με βάση την πρότερη εμπειρία τους.
- ✓ να συνδυάζουν αλγοριθμικές δομές και δεδομένα/δομές δεδομένων για να δημιουργούν κώδικα/πρόγραμμα.
- ✓ να διαπιστώνουν ότι οι σημερινές εφαρμογές είναι αρκετά πολύπλοκες και η δημιουργία τους ακολουθεί συγκεκριμένα μοντέλα ανάπτυξης εφαρμογών λογισμικού που εξελίσσονται σε συγκεκριμένες φάσεις.

2.3.1 Αναφορά σε γλώσσες προγραμματισμού και «Προγραμματιστικά Υποδείγματα»

2.3.1.1 Πρόγραμμα και Γλώσσες Προγραμματισμού

Για να αναπαρασταθούν οι αλγόριθμοι σε μορφή κατανοητή από τον υπολογιστή αναπτύσσονται προγράμματα.

Πρόγραμμα είναι το σύνολο των εντολών που χρειάζεται να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος.

Η εργασία σύνταξης των προγραμμάτων σε κάποια γλώσσα προγραμματισμού ονομάζεται ***προγραμματισμός*** και τα άτομα που γράφουν και συντάσσουν ένα πρόγραμμα ονομάζονται ***προγραμματιστές***. Βασικό στοιχείο του προγράμματος, εκτός από τον αλγόριθμο που υλοποιεί, είναι τα δεδομένα και οι δομές δεδομένων που επεξεργάζεται.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστή). Ο υπολογιστής κάνει στοιχειώδεις ενέργειες σε ακολουθίες των δύο ψηφίων 0 και 1 (***δυαδικά ψηφία***, bits), αλλά αυτές τις ενέργειες τις εκτελεί με ασύλληπτη ταχύτητα. Συγκεκριμένα μπορεί να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.



Προερωτήσεις

- Η δημιουργία του αλγορίθμου αρκεί για να επιλύσεις ένα πρόβλημα στον υπολογιστή;
- Γνωρίζεις κάποιες γλώσσες προγραμματισμού;
- Έχεις ακολουθήσει κάποια μεθοδολογία ή τεχνική για να επιλύσεις ένα πρόβλημα;
- Ποια νομίζεις ότι είναι η δουλειά του προγραμματιστή υπολογιστών;



Εντολές σε γλώσσα μηχανής που καταχωρούν το άθροισμα των τιμών δύο θέσεων μνήμης σε μία άλλη.

0000001001011010
0000101001011110
0000011011011110

Η περιγραφή των παραπάνω εντολών είναι η εξής:

- Μετάφερε στον καταχωρητή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011010.
- Πρόσθεσε στο περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011110.
- Μετάφερε και αποθήκευσε το περιεχόμενο του καταχωρητή στη θέση μνήμης με διεύθυνση 11011110.

Αρχικά τα προγράμματα γράφονταν σε ***γλώσσα μηχανής***, δηλαδή ακολουθίες δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες. Ο συγκεκριμένος τρόπος γραφής προγραμμάτων είναι επίπονος και ελάχιστοι μπορούν να τον κατανοήσουν και να τον υλοποιήσουν, αφού απαιτεί βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Στη συνέχεια αναπτύχθηκαν οι ***συμβολικές γλώσσες*** οι οποίες κάνουν χρήση εντολών που αποτελούνται από συμβολικά ονόματα τα οποία αντιστοιχούν σε εντολές της γλώσσας μηχανής. Το έργο της μετάφρασης των εντολών σε γλώσσα μηχανής το αναλαμβάνει ένα ειδικό πρόγραμμα, ο ***συμβολομεταφραστής*** (assembler).

Οι συμβολικές γλώσσες ήταν σαφώς μια εξέλιξη αλλά παραμένουν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή. Επιπλέον η έλλειψη εντολών σύνθετων λειτουργιών στις παραπάνω γλώσσες οδηγεί σε μακροσκελή προγράμματα που είναι δύσκολο να γραφούν και να συντηρηθούν. Ακόμη, δεν είναι δυνατό να μεταφερθούν και να εκτελεστούν σε υπολογιστή διαφορετικής αρχιτεκτονικής.

Οι παραπάνω ανεπάρκειες και η προσπάθεια για καλύτερη επικοινωνία ανθρώπου – μηχανής οδήγησαν στην εμφάνιση των ***γλωσσών υψηλού επιπέδου***. Σε σχέση με τις συμβολικές γλώσσες στις γλώσσες υψηλού επιπέδου:

- είναι φυσικότερος και πιο ανθρώπινος ο τρόπος έκφρασης των προβλημάτων.
- υπάρχει δυνατότητα μεταφοράς, «μεταφερσιμότητα» δηλαδή, εκτέλεσης των προγραμμάτων σε οποιοδήποτε υπολογιστή.
- είναι εύκολη η εκμάθηση, η διόρθωση των λαθών και η συντήρηση των προγραμμάτων.

Έτσι αναπτύχθηκαν γλώσσες όπως οι ακόλουθες:

- **FORTRAN** (FORrmula TRANslation, Μετάφραση Τύπων). Το 1957 η IBM ανέπτυξε την πρώτη γλώσσα υψηλού επιπέδου. Αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων.
- **COBOL** (COmmon Business Oriented Language, Κοινή Γλώσσα Προσανατολισμένη στις Επιχειρήσεις). Κατάλληλη για ανάπτυξη εμπορικών και γενικά διαχειριστικών εφαρμογών. Χρησιμοποιείται από επιχειρήσεις και από τη δημόσια διοίκηση.
- **LISP** (LISt Processor, Επεξεργαστής Λίστας). Συναρτησιακή γλώσσα η οποία προσανατολίζεται σε χειρισμό λιστών από σύμβολα. Χρησιμοποιείται στο χώρο της τεχνητής νοημοσύνης, σε έμπειρα συστήματα, παιχνίδια, επεξεργασία φυσικών γλωσσών.
- **PROLOG** (PROgramming in LOGic, Λογικός Προγραμματισμός). Η γλώσσα PROLOG χρησιμοποιεί μεθόδους λογικής για



Εντολές σε **συμβολική γλώσσα** που καταχωρούν το άθροισμα των τιμών δύο θέσεων μνήμης σε μία άλλη.

LDA B
ADD C
STA A

Η περιγραφή των παραπάνω εντολών είναι η εξής:

- Μετάφερε στον καταχωρητή το περιεχόμενο της θέσης μνήμης με όνομα B.
- Πρόσθεσε στο περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης με όνομα C.
- Μετάφερε και αποθήκευσε το περιεχόμενο του καταχωρητή στη θέση μνήμης με όνομα A.

Εντολή στη **γλώσσα BASIC** που καταχωρεί το άθροισμα των τιμών δύο μεταβλητών B και C στη μεταβλητή A.

A = B + C

να αναπαραστήσει τη γνώση και να επιλύσει προβλήματα. Χρησιμοποιείται όπως και η LISP στο χώρο της τεχνητής νοημοσύνης, σε έμπειρα συστήματα, παιχνίδια, επεξεργασία φυσικών γλωσσών.

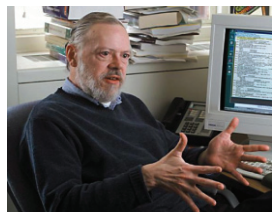
- **BASIC** (Beginner's All Purpose Symbolic Instruction Code, Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους). Γλώσσα που αναπτύχθηκε για την εκπαίδευση αρχαρίων στον προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα τα οποία εκτελούνται με τη βοήθεια διερμηνευτή. Η ανάπτυξη των μικροϋπολογιστών και η τυποποίησή της από τη Microsoft, την καθιέρωσε ως πρότυπο για ανάπτυξη εφαρμογών σε προσωπικούς υπολογιστές.
- **PASCAL**. Είναι μια γλώσσα γενικής χρήσης. Διέπεται από τις αρχές του δομημένου προγραμματισμού. Γνώρισε τεράστια εξάπλωση, και επηρέασε την ανάπτυξη άλλων γλωσσών όπως η ADA.
- **C** και η μετεξέλιξη της **C++**. Γλώσσα η οποία δημιουργήθηκε στα τα εργαστήρια BELL και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος UNIX. Είναι γλώσσα με ισχυρά χαρακτηριστικά. Η C++ είναι γλώσσα αντικειμενοστρεφούς προγραμματισμού.
- **JAVA**. Γλώσσα αντικειμενοστρεφής που αναπτύχθηκε από τη SUN με σκοπό την ανάπτυξη εφαρμογών για το διαδίκτυο.

Η εμφάνιση των γραφικών περιβαλλόντων εργασίας δημιούργησε την ανάγκη για προγράμματα που να εκμεταλλεύονται τον γραφικό τρόπο επικοινωνίας χρήστη – υπολογιστή. Έτσι γλώσσες όπως η BASIC, η C++, η PASCAL που είναι μεν κειμενικές, εξελίχθηκαν (Visual Basic, Visual C++, Delphi) ώστε να διαθέτουν και οπτικό περιβάλλον προγραμματισμού.

Αναπτύχθηκαν όμως και γλώσσες όπως η **SCRATCH – BYOB** και η **Google AppInvertor** που είναι αποκλειστικά οπτικές γλώσσες προγραμματισμού (Visual Programming Languages, VPL). Οι συγκεκριμένες δίνουν τη δυνατότητα στον προγραμματιστή να δημιουργήσει προγράμματα μέσα από το γραφικό χειρισμό προγραμματιστικών στοιχείων (αντί κειμένου).

Η χρήση των υπολογιστών σχεδόν σε όλες τις εκφάνσεις της ανθρώπινης δραστηριότητας δημιούργησε την ανάγκη για γλώσσες κατάλληλες στην επίλυση συγκεκριμένων προβλημάτων. Έτσι αναπτύχθηκαν γλώσσες όπως η **LOGO** ή η **GameMaker** για εκπαιδευτικούς σκοπούς, η **LabView** που χρησιμοποιείται από τους επιστήμονες και τους μηχανικούς στο σχεδιασμό, τον έλεγχο και τη δοκιμή καταναλωτικών προϊόντων κ.ά..

Οι γλώσσες υψηλού επιπέδου χαρακτηρίζονται ως γλώσσες τρίτης γενιάς ενώ οι συμβολικές ως δεύτερης γενιάς ή χαμηλού επιπέδου. Από



Εικόνα 2.25

Ντένις Ρίτσι (Dennis Ritchie). Ο δημιουργός της γλώσσας προγραμματισμού C και ο βασικός συντελεστής στην ανάπτυξη του λειτουργικού συστήματος UNIX. «Το UNIX είναι βασικά ένα απλό λειτουργικό σύστημα, πρέπει όμως να είσαι ιδιοφυία για να καταλάβεις την απλότητά του».



Σε ένα οπτικό περιβάλλον προγραμματισμού υπάρχει η δυνατότητα να δημιουργείται γραφικά ολόκληρο το περιβάλλον της εφαρμογής, όπως για παράδειγμα τα πλαίσια διαλόγου ή τα μενού.

Αρχικά υπήρχαν τα περιβάλλοντα Γραμμής Εντολών, όπου ο χρήστης είχε τη δυνατότητα να πληκτρολογεί τις εντολές και να τις βλέπει στην οθόνη ενώ στα γραφικά περιβάλλοντα εργασίας τα προγράμματα και οι πληροφορίες εμφανίζονται στην οθόνη με γραφικά και σχήματα.



Εικόνα 2.26. Λογότυπο της γλώσσας Google AppInvertor.

την άλλη ένα πρόγραμμα σε γλώσσα μηχανής είναι κωδικοποιημένο σε γλώσσα πρώτης γενιάς.

Οι παραπάνω γενιές γλωσσών προγραμματισμού απευθύνονται μόνο σε προγραμματιστές και ο χρήστης δεν έχει τη δυνατότητα να επιφέρει αλλαγές σε κάποιο πρόγραμμα, προκειμένου να ικανοποιήσει μια νέα ανάγκη του. Σταδιακά όμως πολλές γλώσσες εφοδιάστηκαν με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο χρήστης μπορεί να επιλύει μόνος του μικρά προβλήματα εφαρμογών. Αυτή η αυξανόμενη τάση απόκρυψης της αρχιτεκτονικής του υλικού και της τεχνικής του προγραμματισμού οδήγησε στις γλώσσες τέταρτης γενιάς. Η **SQL** (Structured Query Language, Δομημένη Γλώσσα Ερωτοαποκρίσεων) είναι μία γλώσσα τέταρτης γενιάς η οποία χρησιμοποιείται για την ανάκτηση και τη διαχείριση δεδομένων καθώς και την παραγωγή πληροφοριών σε σχεσιακές βάσεις δεδομένων.

```
SELECT ENAME, JOB, SAL
FROM EMPLOYES
WHERE DEPTNO = 20
AND SAL > 1000;
```

Με την ερώτηση αυτή σε SQL εκτελείται αναζήτηση στη βάση δεδομένων EMPLOYES και επιστρέφει το όνομα, τη θέση και τον μισθό των υπαλλήλων της διεύθυνσης 20 που κερδίζουν πάνω από 1000 ευρώ.

2.3.1.2 Προγραμματιστικά Υποδείγματα

Αναφέρθηκε προηγουμένως ότι κάποιες γλώσσες ακολουθούν τον αντικειμενοστρεφή προγραμματισμό και άλλες είναι συναρτησιακές ή χρησιμοποιούν μεθόδους λογικής για να επιλύσουν προβλήματα. Η ανάπτυξη λοιπόν ενός προγράμματος σε κάποια γλώσσα προγραμματισμού βασίζεται σε ένα πρότυπο ή μία καθορισμένη μεθοδολογία.

Ως «**Προγραμματιστικό Υπόδειγμα**» εννοείται ένα πρότυπο ανάπτυξης προγραμμάτων, δηλαδή μία καθορισμένη μεθοδολογία με βάση την οποία αναπτύσσονται η δομή και τα στοιχεία του προγράμματος.

Οι δυνατότητες και οι μεθοδολογίες ανάπτυξης προγραμμάτων που παρέχει μία γλώσσα προγραμματισμού, καθορίζονται από το προγραμματιστικό υπόδειγμα που ακολουθεί. Υπάρχουν όμως γλώσσες που έχουν σχεδιαστεί να υποστηρίζουν περισσότερα από ένα υποδείγματα.

Τα κυριότερα προγραμματιστικά υποδείγματα είναι:

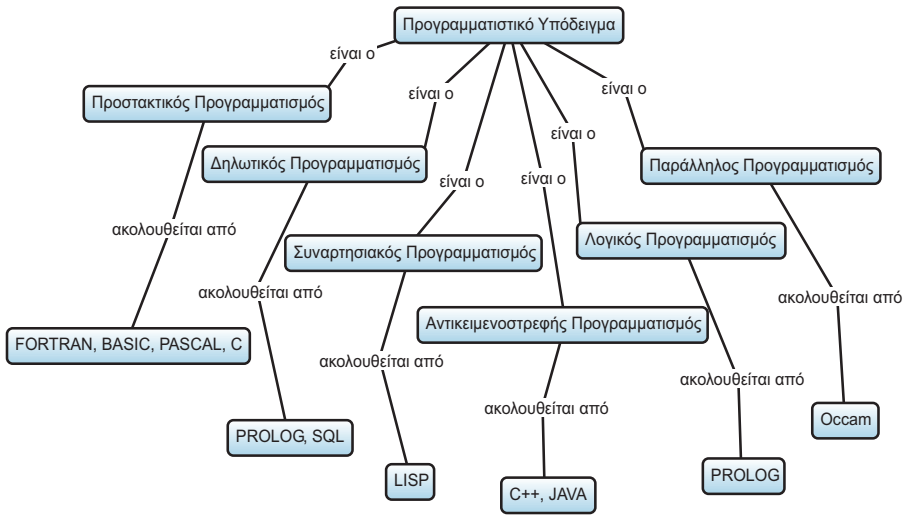
- Ο ***προστακτικός προγραμματισμός*** όπου τα προγράμματα αναπτύσσονται με απλές εντολές σε προστακτική (Διάβασε, Εμφάνισε, Επανάλαβε) που ζητούν από τον υπολογιστή να εκτελέσει συγκεκριμένες ενέργειες και να ακολουθήσει βήματα με μία λογική σειρά για να επιλύσει το πρόβλημα που έχει δοθεί. Γλώσσες, όπως η FORTRAN, η BASIC, η PASCAL, η C, ακολουθούν αυτό το υπόδειγμα.
- Ο ***δηλωτικός προγραμματισμός*** όπου, σε αντίθεση με τον προστακτικό προγραμματισμό, το πρόβλημα επιλύεται δηλώνοντας απλώς τις επιθυμητές ιδιότητες του αποτελέσματος. Το πρόγραμμα περιέχει λογικές εκφράσεις, ενώ κατά την εκτέλεσή του γίνεται έλεγχος για το ποιες ακριβώς ικανοποιούνται. Παραδείγματα γλωσσών που τον ακολουθούν είναι η PROLOG και η SQL.



Εικόνα 2.27

Η Άντα Λάβλεϊς (Ada Lovelace), κόρη του Λόρδου Βύρωνα, έγραψε το πρώτο πρόγραμμα υπολογιστή κατά τον 19ο αιώνα για την Αναλυτική Μηχανή του Τσαρλς Μπάμπατζ (Charles Babbage), πολύ πριν από την εμφάνιση ηλεκτρονικών υπολογιστών. Η γλώσσα προγραμματισμού ADA έχει ονομαστεί έτσι προς τιμήν της.

- Ο ***συναρτησιακός προγραμματισμός*** επιλύει το πρόβλημα με τη χρήση μαθηματικών συναρτήσεων. Οι συναρτήσεις παράγουν αποτελέσματα με βάση τα δεδομένα εισόδου τους. Παράδειγμα συναρτησιακής γλώσσας είναι η LISP.
- Ο ***αντικειμενοστρεφής προγραμματισμός*** βασίζεται στην έννοια του αντικειμένου. Τα αντικείμενα δημιουργούνται από τις κλάσεις. Μία κλάση ορίζει τα χαρακτηριστικά και τη συμπεριφορά ενός τύπου αντικειμένου, λειτουργεί δηλαδή ως πρότυπο. Ένα αντικείμενο είναι μία δομή δεδομένων η οποία περιέχει τόσο τα δεδομένα (χαρακτηριστικά που την περιγράφουν) όσο και τις διαδικασίες (μεθόδους) που επενεργούν σε αυτά. Τα αντικείμενα μπορούν να αλληλεπιδρούν μεταξύ τους. Αντικειμενοστραφείς γλώσσες είναι η C++ και η JAVA.
- Ο ***λογικός προγραμματισμός*** όπου τα προγράμματα είναι γραμμένα ως ένα σύνολο από προτάσεις σε μορφή λογικών εκφράσεων. Το συγκεκριμένο υπόδειγμα βασίζεται στα γεγονότα, στους κανόνες και στις ερωτήσεις και ακολουθείται κυρίως στο πεδίο της Τεχνητής Νοημοσύνης. Παράδειγμα γλώσσας που τον ακολουθεί είναι η PROLOG.
- Ο ***παράλληλος προγραμματισμός*** στον οποίο τα προγράμματα εκμεταλλεύονται την ύπαρξη υπολογιστών που διαθέτουν περισσότερους από έναν επεξεργαστές. Έτσι επιτυγχάνεται η αύξηση των υπολογιστικών επιδόσεων και η μείωση του χρόνου εκτέλεσης της εφαρμογής. Θα πρέπει όμως το πρόβλημα προς επίλυση να διαιρεθεί σε τμήματα που μπορούν να επιλυθούν παράλληλα. Μία γλώσσα που υποστηρίζει τον παράλληλο προγραμματισμό είναι η Occam.



Εικόνα 2.29. Προγραμματιστικά Υποδείγματα.

2.3.1.3 Δομημένος Προγραμματισμός

Μία μεθοδολογία ανάλυσης, σχεδίασης και συγγραφής προγραμμάτων που αναπτύχθηκε και διαδόθηκε ευρύτατα στον χώρο της πληροφορι-



Εικόνα 2.28

Γκρέις Χόπερ (Grace Hopper). Αμερικανίδα καθηγήτρια μαθηματικών και αξιωματικός του αμερικανικού ναυτικού, η οποία ηγήθηκε της ομάδας που ανέπτυξε την πρώτη προηγμένη γλώσσα προγραμματισμού με προορισμό τον επιχειρηματικό κόσμο, την Common Business Oriented Language (COBOL)



Εικόνα 2.30. Νίκλαους Βιρθ (Niklaus Wirth)

Δημιουργός της γλώσσας PASCAL, η οποία ονομάστηκε έτσι προς τιμήν του Γάλλου επιστήμονα Μπλεζ Πασκάλ (Blaise Pascal).



Ο δομημένος προγραμματισμός προτάθηκε ως έννοια το 1966 από την ανάγκη να περιοριστεί η χρήση των εντολών GOTO (Πήγαινε).



Υποπρόγραμμα είναι το τμήμα προγράμματος που επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα.



Στον δομημένο προγραμματισμό γίνεται χρήση υποπρογραμμάτων όπως οι διαδικασίες και οι συναρτήσεις.

Τα υποπρογράμματα που αναπτύσσονται μπορούν να αποτελέσουν σύνθετες εντολές προς τον υπολογιστή.

Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μόνο μία έξοδο. Δηλαδή, όταν ενεργοποιείται, δέχεται κατά βάση κάποια δεδομένα και, αφού ολοκληρώσει τη λειτουργία του, επιστρέφει κάποια αποτελέσματα. Κατά τη διάρκεια όμως της εκτέλεσής του δεν εκτελείται καμία άλλη εντολή του προγράμματος πέρα από αυτές που συμπεριλαμβάνονται στο συγκεκριμένο υποπρόγραμμα.

κής είναι ***ο δομημένος προγραμματισμός*** *(structured programming)* ο οποίος χρησιμοποιεί

- την ιεραρχική σχεδίαση για την ανάπτυξη του αλγορίθμου που επιλύει το πρόβλημα.
- τον τμηματικό προγραμματισμό για τη σχεδίαση του προγράμματος και για τη δημιουργία των ενοτήτων του.
- τρεις βασικές συνιστώσες για τη συγγραφή των επιμέρους ενοτήτων που καθιστούν άσκοπη τη χρήση της εντολής GOTO (Πήγαινε).

Η ***ιεραρχική σχεδίαση*** ή ανάλυση «από πάνω προς τα κάτω» χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα. Αυτά είναι πιο εύκολο να επιλυθούν οδηγώντας τελικά στην επίλυση του αρχικού προβλήματος.

Ο ***τμηματικός προγραμματισμός*** υλοποιεί την ιεραρχική σχεδίαση όπου κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα που ονομάζεται υποπρόγραμμα. Τα υποπρογράμματα πρέπει να χαρακτηρίζονται από όσο το δυνατόν μεγαλύτερο βαθμό ανεξαρτησίας και από υψηλό βαθμό συνεκτικότητας, δηλαδή να υλοποιούν μια μόνο συγκεκριμένη διαδικασία ή λειτουργία. Το πρόγραμμα τελικά είναι η σύνθεση όλων των υποπρογραμμάτων.

Τέλος, όλες οι ενότητες του προγράμματος μπορούν να αναπτυχθούν μόνο με τρεις βασικές συνιστώσες: τη ***δομή ακολουθίας***, τη ***δομή επιλογής*** και τη ***δομή επανάληψης*** ή συνδυασμό τους. Οι τρεις αυτές συνιστώσες ομαδοποιούν τις εντολές και καθορίζουν τη σειρά εκτέλεσης τους. Οι εντολές λοιπόν του προγράμματος οργανώνονται σε μία δομή.

Σε αντίθεση, η χρήση της εντολής GOTO έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, δηλαδή της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη. Έτσι αυξάνεται η δυσκολία στην αρχική σχεδίαση της λύσης, στην παρακολούθηση και κατανόηση του προγράμματος και τέλος στη συντήρησή του.

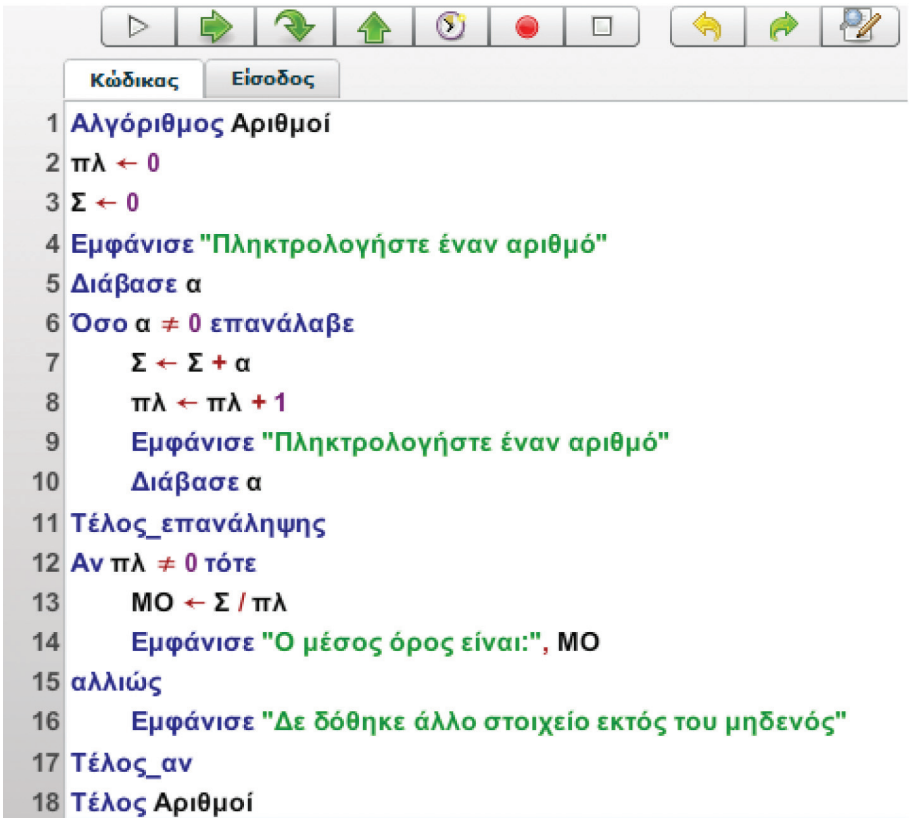
Ο δομημένος προγραμματισμός αποτελεί λοιπόν μια μεθοδολογία σύνταξης προγραμμάτων που έχει σκοπό να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων, να μειώσει τα λάθη, να διευκολύνει την κατανόηση των προγραμμάτων, τις διορθώσεις και τις αλλαγές σε αυτά.

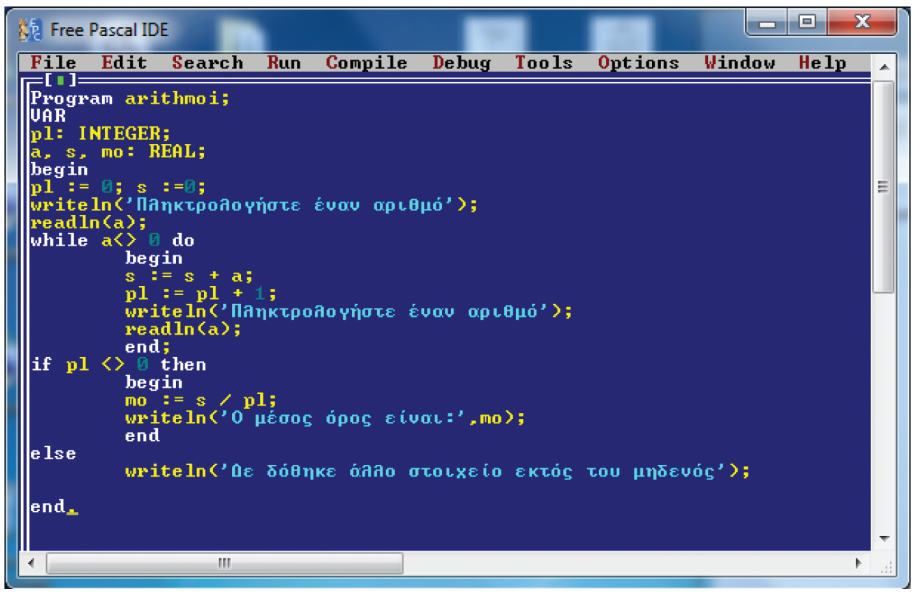
Οι γλώσσες στις οποίες αναπτύσσονται προγράμματα με τη μεθοδολογία του δομημένου προγραμματισμού ακολουθούν κατά βάση το υπόδειγμα του προστακτικού προγραμματισμού.

Παράδειγμα 2.32. Να γραφεί πρόγραμμα το οποίο θα διαβάζει αριθμούς και θα υπολογίζει το μέσο όρο τους. Η διαδικασία εισαγωγής αριθμών θα σταματά αν διαβαστεί ο αριθμός 0, χωρίς να συνυπολογίζεται. Αν ο πρώτος αριθμός που θα διαβαστεί είναι το μηδέν, θα εκτυ-

πώνει το μήνυμα «Δεν δόθηκε άλλο στοιχείο εκτός του μηδενός». Το πρόγραμμα να υλοποιηθεί:

1. στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL).
2. στη γλώσσα προγραμματισμού PASCAL.







Το προγραμματιστικό περιβάλλον Διερμηνευτής **Ψευδογλώσσας (ViALGOL)** υποστηρίζει τις εντολές της ψευδογλώσσας που ορίστηκε στο προηγούμενο κεφάλαιο και ακολουθεί τις αρχές του δομημένου προγραμματισμού.



Στο πρόγραμμα αυτό θα εισαχθεί άγνωστο πλήθος αριθμών προς επεξεργασία. Αυτό θα επιτευχθεί με την εντολή **Όσο**. Επίσης είναι απαραίτητο να ελεγχθεί αν δόθηκε ένα τουλάχιστον στοιχείο διαφορετικό από το μηδέν ή όχι. Για το συγκεκριμένο έλεγχο θα χρησιμοποιηθεί η εντολή σύνθετης επιλογής.



Το προγραμματιστικό περιβάλλον στο οποίο υλοποιήθηκε το πρόγραμμα στη γλώσσα PASCAL μπορεί να βρεθεί στη διεύθυνση http://www.freepascal.org/.

Η εντολή **Όσο** υλοποιείται με την εντολή **While** και η σύνθετη εντολή επιλογής με την εντολή **If…else**

Η σχεδίαση προγραμμάτων στη γλώσσα PASCAL ακολουθεί τον δομημένο προγραμματισμό.



Το πρόγραμμα που γράφεται σε κάποια γλώσσα προγραμματισμού ονομάζεται **πηγαίο πρόγραμμα** (source program).



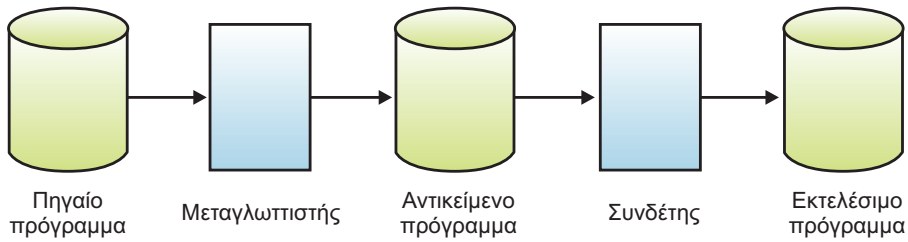
Οι **μεταγλωττιστές** (compilers) δέχονται στην είσοδο ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου και παράγουν ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το πρόγραμμα που παράγεται ονομάζεται **αντικείμενο** (object) πρόγραμμα. Το αντικείμενο πρόγραμμα δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συνδεθεί με άλλα τμήματα προγράμματος τα οποία είτε τα γράφει ο προγραμματιστής, είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται **συνδέτης – φορτωτής** (linker - loader). Το αποτέλεσμα είναι η παραγωγή του **εκτελέσιμου** (executable) προγράμματος (Εικόνα 2.31). Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το πηγαίο πρόγραμμα.

Οι **διερμηνευτές** (interpreters) διαβάζουν μία προς μία τις εντολές του πηγαίου προγράμματος και για καθεμία εκτελούν αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

2.3.2 Σχεδίαση και συγγραφή κώδικα

Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζονται από προγραμματιστικά περιβάλλοντα τα οποία παρέχουν εργαλεία που διευκολύνουν την εργασία του προγραμματιστή. Για τη σύνταξη του ***πηγαίου προγράμματος*** χρησιμοποιείται ένα ειδικό πρόγραμμα το οποίο ονομάζεται ***συντάκτης*** (editor). Στη συνέχεια το πηγαίο πρόγραμμα πρέπει να μεταφραστεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή δηλαδή σε εντολές γλώσσας μηχανής. Το έργο της μετάφρασης το αναλαμβάνουν δύο προγράμματα ο ***μεταγλωττιστής*** ή ο ***διερμηνευτής***.

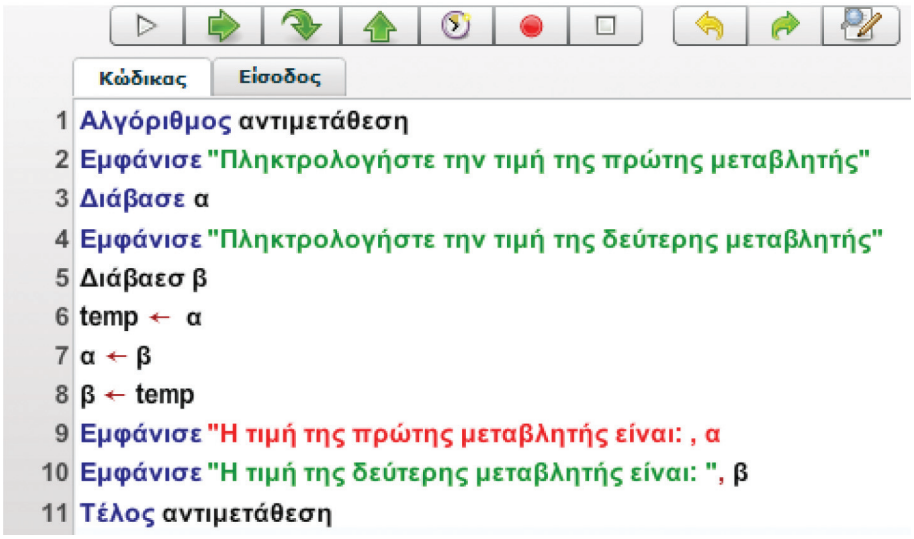
Για να μεταφραστεί το πηγαίο πρόγραμμα σε εντολές γλώσσας μηχανής δεν θα πρέπει να ανιχνευθούν λάθη. Τα λάθη που εμφανίζονται κατά τη μετάφραση ονομάζονται ***συντακτικά***. Τα συντακτικά λάθη μπορεί να οφείλονται σε αναγραμματισμούς, σε λανθασμένη σύνταξη εντολών, παράλειψη δήλωσης μεταβλητών κ.ά.. Ο μεταφραστής ανιχνεύει τα λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα. Στη συνέχεια ακολουθεί η διόρθωσή τους από τον προγραμματιστή.



Εικόνα 2.31. Μεταγλώττιση και Σύνδεση Προγράμματος.

Πέρα όμως από τα συντακτικά λάθη υπάρχουν και τα ***λογικά*** που δεν είναι δυνατό να ανιχνευθούν από τα μεταφραστικά προγράμματα. Τα περισσότερα όμως προγραμματιστικά περιβάλλοντα παρέχουν εργαλεία εκσφαλμάτωσης που βοηθούν τον προγραμματιστή να εκτελέσει το πρόγραμμα εντολή προς εντολή μέχρι συγκεκριμένο σημείο ή να παρακολουθεί τις τιμές των μεταβλητών έτσι ώστε να εντοπίσει τα λάθη στην υλοποίηση του αλγορίθμου.

Παράδειγμα 2.33. Με σκοπό να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών γράφτηκαν οι ακόλουθες εντολές στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL).



Τι θα συμβεί κατά τη μετάφραση του προγράμματος;

Απάντηση

Υπάρχουν δύο συντακτικά λάθη στις εντολές που έχουν γραφεί. Το πρώτο βρίσκεται στην πέμπτη γραμμή του προγράμματος και οφείλεται σε αναγραμματισμό της εντολής Διάβασε και το δεύτερο στην έναρξη γραμμή και οφείλεται στην παράλειψη των διπλών εισαγωγικών στο τέλος της αλφαριθμητικής έκφρασης. Το προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) χρησιμοποιεί διερμηνευτή για τη μετάφραση του πηγαίου προγράμματος. Έτσι αρχικά θα εντοπιστεί το πρώτο συντακτικό λάθος με ένα μήνυμα της μορφής:

Συντακτικό Λάθος - Περίμενα την εντολή εκχώρησης
Βρήκα: β

Ο διερμηνευτής αναγνωρίζει τη λέξη Διάβαεσ ως το όνομα κάποιας μεταβλητής και περιμένει στη συνέχεια να δει το αριστερό βέλος της εντολής εκχώρησης, κάτι το οποίο όμως δεν συμβαίνει. Επίσης εντοπίζει μόνο το πρώτο συντακτικό λάθος. Εφόσον αυτό διορθωθεί, ο διερμηνευτής βρίσκει το επόμενο λάθος και εμφανίζει ένα μήνυμα της μορφής:

Συντακτικό Λάθος - Περίμενα το χαρακτήρα. "

Παράδειγμα 2.34. Με σκοπό να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών γράφτηκαν οι ακόλουθες εντολές στο προγραμματιστικό περιβάλλον FreePascal.



Στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας, το χρώμα των γραμμάτων διαφοροποιείται σε μπλε για τις δεσμευμένες λέξεις, σε μαύρο για τις μεταβλητές και πράσινο για τις αλφαριθμητικές σταθερές ώστε να διακρίνονται εύκολα τα διάφορα στοιχεία του προγράμματος.



Το ερώτημα αν οι μηχανές μπορούν να σκέφτονται έχει την ίδια σημασία με το ερώτημα αν τα υποβρύχια μπορούν να κολυμπούν.

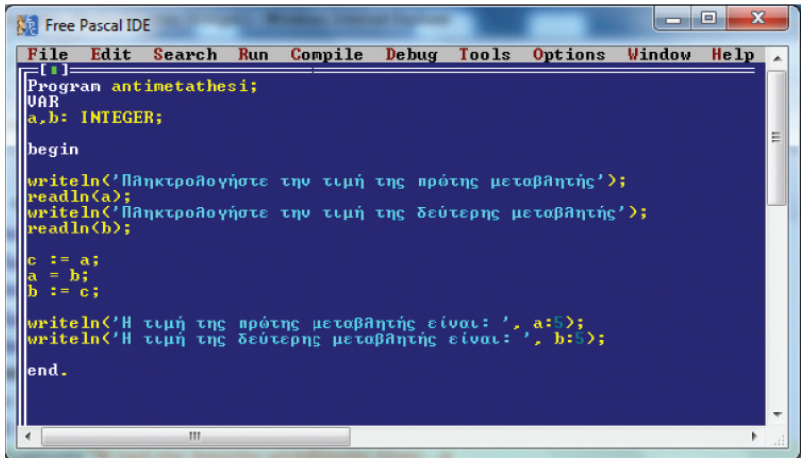
Έντσγκερ Ντάικστρα (Edsger Dijkstra), Επιστήμονας Πληροφορικής



Η γλώσσα PASCAL είναι μία από τις γλώσσες για τη συγγραφή κώδικα στον Πανελλήνιο Διαγωνισμό Πληροφορικής. http://www.pdp.gr



Στη γλώσσα προγραμματισμού PASCAL απαιτείται η δήλωση του τύπου των μεταβλητών που χρησιμοποιούνται από το πρόγραμμα.



Τι θα συμβεί κατά τη μετάφραση του προγράμματος;

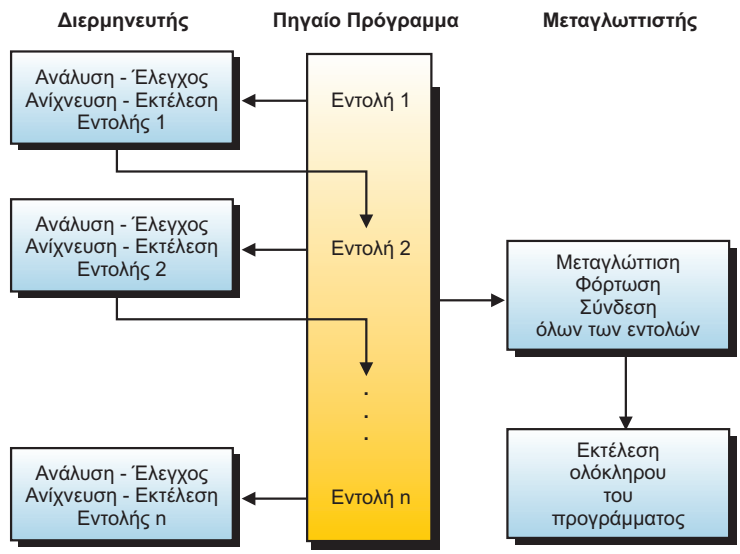
Απάντηση

Το μεταφραστικό πρόγραμμα στο συγκεκριμένο προγραμματιστικό περιβάλλον είναι μεταγλωττιστής. Έτσι θα εμφανιστεί το εξής μήνυμα.

```
antimetathesi.pas(12,3) Error: Identifier not found "c"
antimetathesi.pas(13,6) Error: Illegal expression
antimetathesi.pas(14,7) Error: Identifier not found "c"
antimetathesi.pas(20) Fatal: There were 3 errors compiling module,
antimetathesi.pas(0) Fatal: Compilation aborted
```

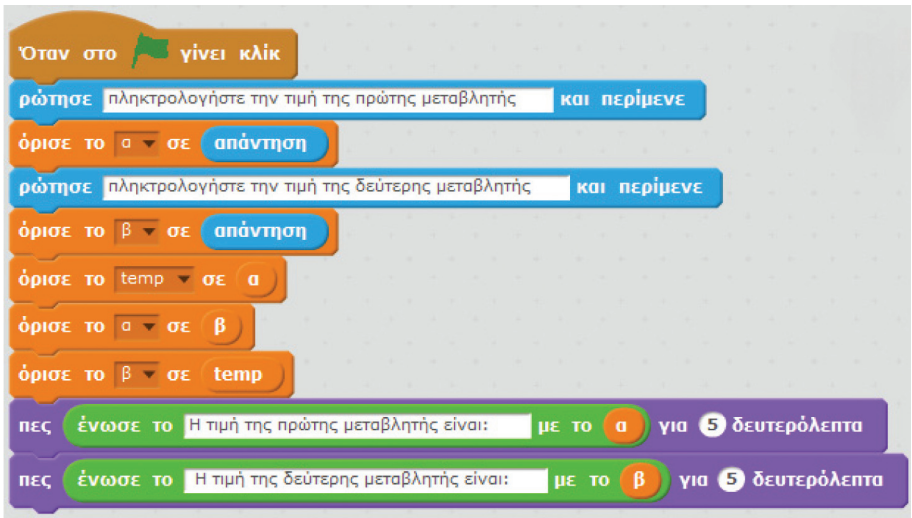
Εντοπίζονται όλα τα συντακτικά λάθη που υπάρχουν. Τα δύο οφείλονται σε παράλειψη δήλωσης της μεταβλητής c και το άλλο οφείλεται σε λανθασμένη σύνταξη της εντολής εκχώρησης της τιμής της μεταβλητής b στη μεταβλητή a.

Η διαδικασία μετάφρασης λοιπόν ενός προγράμματος σε προγραμματιστικά περιβάλλοντα που διαθέτουν είτε διερμηνευτή είτε μεταγλωττιστή φαίνεται στην εικόνα 2.32.



Εικόνα 2.32. Διαδικασία Μετάφρασης και Εκτέλεσης ενός προγράμματος.

Παράδειγμα 2.35. Να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών στο προγραμματιστικό περιβάλλον SCRATCH.



Παράδειγμα 2.36. Εύρεση μέγιστου κοινού διαιρέτη δύο θετικών ακέραιων αριθμών με τον επαναληπτικό αλγόριθμο του Ευκλείδη σε γλώσσα SCRATCH.



Η γλώσσα οπτικού προγραμματισμού SCRATCH έχει δημιουργηθεί στο MIT. http://scratch.mit.edu

Στη γλώσσα SCRATCH ο προγραμματισμός γίνεται με χρήση εντολών που μοιάζουν με κομμάτια ενός παζλ (**Πλακίδια** – blocks). Κάθε εντολή έχει το δικό της χαρακτηριστικό σχήμα και μπορεί να συνδυαστεί με άλλες μόνο με συγκεκριμένο τρόπο που αποκλείει τα συντακτικά λάθη.

Οι εντολές όταν συνδυάζονται δημιουργούν σενάρια ενεργειών που πρέπει να εκτελεστούν από αντικείμενα που λέγονται **μορφές** (sprites). Η πιο χαρακτηριστική μορφή είναι η γάτα.



Εικόνα 2.33. Η μορφή της γάτας.

Η εντολή **Όσο** της ψευδογλώσσας και η εντολή **While** της PASCAL δεν υπάρχει στη γλώσσα SCRATCH. Υπάρχει όμως η εντολή **επανάλαβε ώσπου** στην οποία οι εντολές εκτελούνται όσο η συνθήκη είναι ψευδής.

Το πρόγραμμα αρχικά εξασφαλίζει ότι οι τιμές που πληκτρολογούνται είναι θετικές.

Παράδειγμα 2.37. Να γραφεί πρόγραμμα για την εύρεση του μέγιστου κοινού διαιρέτη δύο θετικών ακεραίων αριθμών με τον αναδρομικό αλγόριθμο του Ευκλείδη στη γλώσσα LOGO. Ο κώδικας να αναπτυχθεί ως διαδικασία στο περιβάλλον MicroWorlds Pro.

```
για μκδ :χ :ψ
ΑνΔιαφορετικά :ψ = 0
[
  κάνε "ζ :χ
  ανακοίνωση (φρ[ο μέγιστος κοινός διαιρέτης είναι] :ζ)
]
[
  κάνε "χ υπόλοιπο :χ :ψ
  μκδ :ψ :χ
]
τέλος
```

Πολλές φορές κάποια προγράμματα μπορούν να χρησιμοποιήσουν κώδικα που έχει γραφτεί προηγουμένως. Η επαναχρησιμοποίηση κώδικα είναι αρκετά συνηθισμένη πρακτική η οποία περιορίζει τα λάθη και μειώνει το χρόνο που απαιτείται για τη συγγραφή του προγράμματος. Προγραμματιστικά γίνεται συνήθως με τη δημιουργία κατάλληλων υποπρογραμμάτων, δηλαδή διαδικασιών ή συναρτήσεων ανάλογα με τη γλώσσα προγραμματισμού. Αρκετές φορές ο προγραμματιστής μπορεί να γράψει ή να χρησιμοποιήσει ***βιβλιοθήκες*** (libraries). Οι βιβλιοθήκες μιας γλώσσας προγραμματισμού είναι μία συλλογή από έτοιμα υποπρογράμματα που μπορούν να χρησιμοποιούνται κατά τη συγγραφή νέων προγραμμάτων. Μία βιβλιοθήκη μπορεί να χρησιμοποιηθεί τόσο από τον δημιουργό της όσο και από άλλους προγραμματιστές. Αποτελεί επίσης πρακτική πολλών προγραμματιστών να βελτιώνουν υπάρχουσες βιβλιοθήκες, ώστε να επεκτείνουν τις δυνατότητες των προγραμμάτων τους.

Παράδειγμα 2.38. Να γραφεί πρόγραμμα που θα δέχεται δύο θετικούς ακέραιους αριθμούς και θα εμφανίζει το μέγιστο κοινό διαιρέτη και το ελάχιστο κοινό πολλαπλάσιο τους. Η υλοποίηση να γίνει στο περιβάλλον MicroWorlds Pro.

```
για μκδ :χ :ψ
ΑνΔιαφορετικά :ψ = 0
[
  κάνε "ζ :χ
]
[
  κάνε "χ υπόλοιπο :χ :ψ
  μκδ :ψ :χ
]
τέλος
```



Αρκετές φορές οι αλγόριθμοι χρειάζεται να τροποποιηθούν ώστε να μπορούν να εκτελεστούν από κάποια γλώσσα προγραμματισμού.

Για να εκτελεστεί το διπλανό πρόγραμμα και να ανακοινώσει το μέγιστο κοινό διαιρέτη των αριθμών 27 και 6, αρκεί να πληκτρολογήσει ο χρήστης στο κέντρο εντολών του περιβάλλοντος την εντολή:

μκδ 27 6

Οι μεταβλητές χ και ψ ονομάζονται παράμετροι.

Η σύνθετη επιλογή υλοποιείται με την εντολή

ΑνΔιαφορετικά.



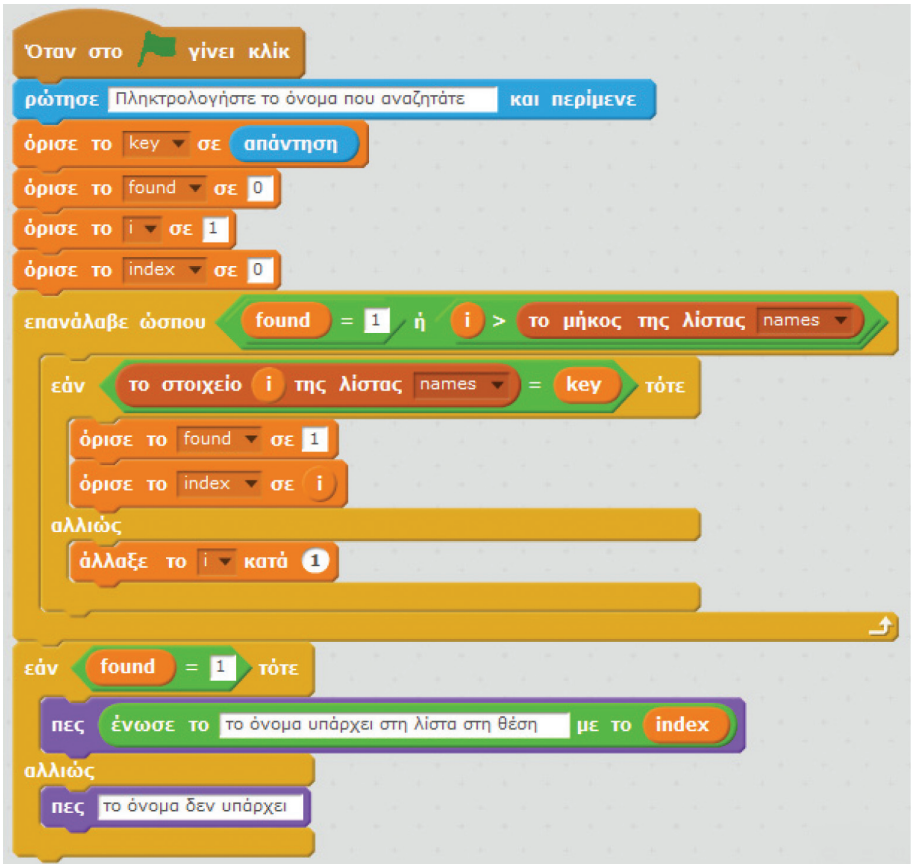
Αν χ, ψ είναι δύο θετικοί ακέραιοι τότε ΜΚΔ(χ,ψ)*ΕΚΠ(χ,ψ) = χ*ψ

Το πρόγραμμα στη γλώσσα LOGO περιλαμβάνει τρεις διαδικασίες: μία για τον υπολογισμό του μέγιστου κοινού διαιρέτη η οποία επαναχρησιμοποιείται, μία για τον υπολογισμό του ελάχιστου κοινού πολλαπλάσιου που καλεί τη διαδικασία του μέγιστου κοινού διαιρέτη και μία που καλεί τη διαδικασία υπολογισμού του ελάχιστου κοινού πολλαπλάσιου και εμφανίζει τα αποτελέσματα.

```
για εκπο :χ :ψ
μκδ :χ :ψ
κάνε "εκ :χ * :ψ / :ζ
τέλος

για υπολογισμός :χ :ψ
εκπο :χ :ψ
ανακοίνωση (φρ[ο μέγιστος κοινός διαιρέτης είναι] :ζ)
ανακοίνωση (φρ[το ελάχιστο κοινό πολλαπλάσιο είναι] :εκ)
τέλος
```

Παράδειγμα 2.39. Να γραφεί πρόγραμμα σε γλώσσα SCRATCH το οποίο θα αναζητά αν υπάρχει ένα όνομα σε μία λίστα και θα εμφανίζει κατάλληλο μήνυμα σχετικά με την εύρεσή του.



Παράδειγμα 2.40. Σε ένα ραδιοφωνικό σταθμό το κόστος ενός διαφημιστικού μηνύματος σε σχέση με τα δευτερόλεπτα μετάδοσης, υπολογίζεται κλιμακωτά σύμφωνα με τον παρακάτω πίνακα.

Χρονική Διάρκεια διαφήμισης (δευτερόλεπτα)	Κόστος (€ / δευτερόλεπτο)
Μέχρι 15	90
Μέχρι 30	75
Πάνω από 30	50

Οι τιμές των μεταβλητών στη LOGO είναι διαθέσιμες σε όλες τις διαδικασίες. Οι μεταβλητές αυτές ονομάζονται ***καθολικές***. Σε άλλες γλώσσες προγραμματισμού υπάρχουν και οι ***τοπικές*** μεταβλητές, οι οποίες μπορούν να προσπελαστούν μόνο στο πρόγραμμα ή το υποπρόγραμμα το οποίο δηλώνονται.



Ο τρόπος επικοινωνίας μεταξύ των ενοτήτων ή υποπρογραμμάτων (διαδικασιών ή συναρτήσεων) και των προγραμμάτων διαφέρει από γλώσσα σε γλώσσα προγραμματισμού. Κατά βάση αυτή η επικοινωνία επιτυγχάνεται με τη χρήση των παραμέτρων, δηλαδή μεταβλητών που επιτρέπουν το πέρασμα της τιμής τους από ένα τμήμα προγράμματος σε ένα άλλο.

Η δομή δεδομένων, που είναι αποθηκευμένα τα ονόματα στο παράδειγμα 2.39, είναι μία λίστα με το όνομα names.

Ο αλγόριθμος που κωδικοποιείται είναι αυτός της σειριακής αναζήτησης κατάλληλα τροποποιημένος.



Η χρήση της δομής του πίνακα, όταν αυτή δεν κρίνεται απαραίτητη, επαφίεται στη λογική του προγραμματιστή. Κατά βάση όμως στις εφαρμογές λογισμικού τα δεδομένα αποθηκεύονται σε δομές δεδομένων.

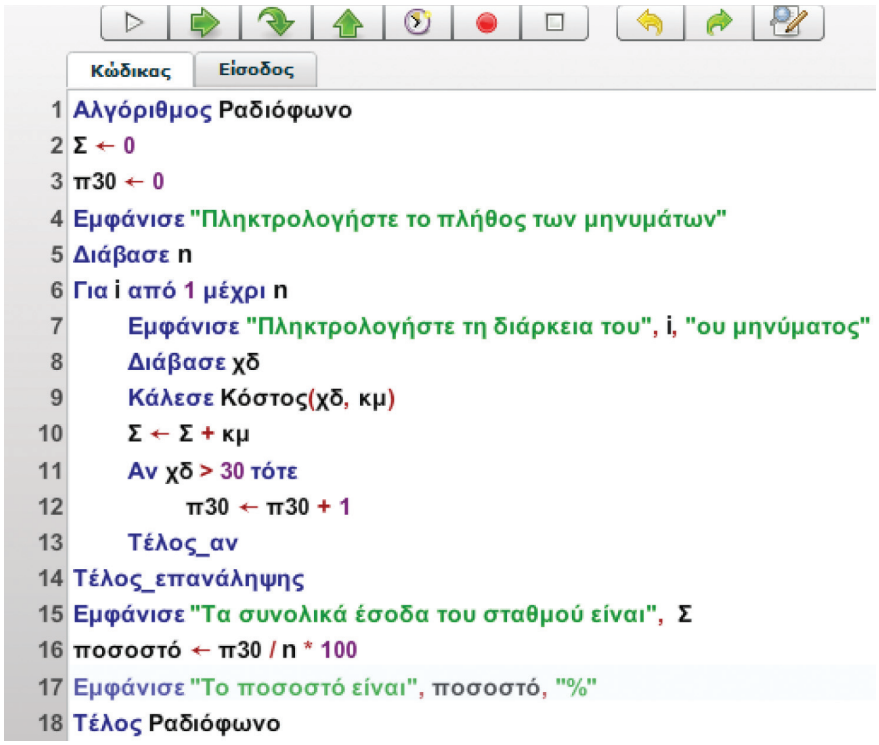


Κατά την κλήση του αλγορίθμου Κόστος, η μεταβλητή χδ μεταβιβάζει την τιμή της στη μεταβλητή δ και όταν ολοκληρώνεται η εκτέλεση του αλγορίθμου η μεταβλητή κ μεταβιβάζει την τιμή της στη μεταβλητή κμ.

A. Να αναπτύξετε πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο
   i. να διαβάζει το πλήθος των διαφορετικών μηνυμάτων που πρόκειται να μεταδώσει ο σταθμός την επόμενη εβδομάδα
   ii. να διαβάζει τη χρονική διάρκεια κάθε μηνύματος και να υπολογίζει καλώντας κατάλληλο υποπρόγραμμα το κόστος του
   iii. να εμφανίζει με κατάλληλο μήνυμα τα συνολικά έσοδα του σταθμού καθώς και το ποσοστό (%) των μηνυμάτων με χρονική διάρκεια άνω των 30 δευτερολέπτων.

B. Να γράψετε τον αλγόριθμο για τον υπολογισμό του κόστους του κάθε μηνύματος.

Ο κώδικας του προγράμματος είναι:



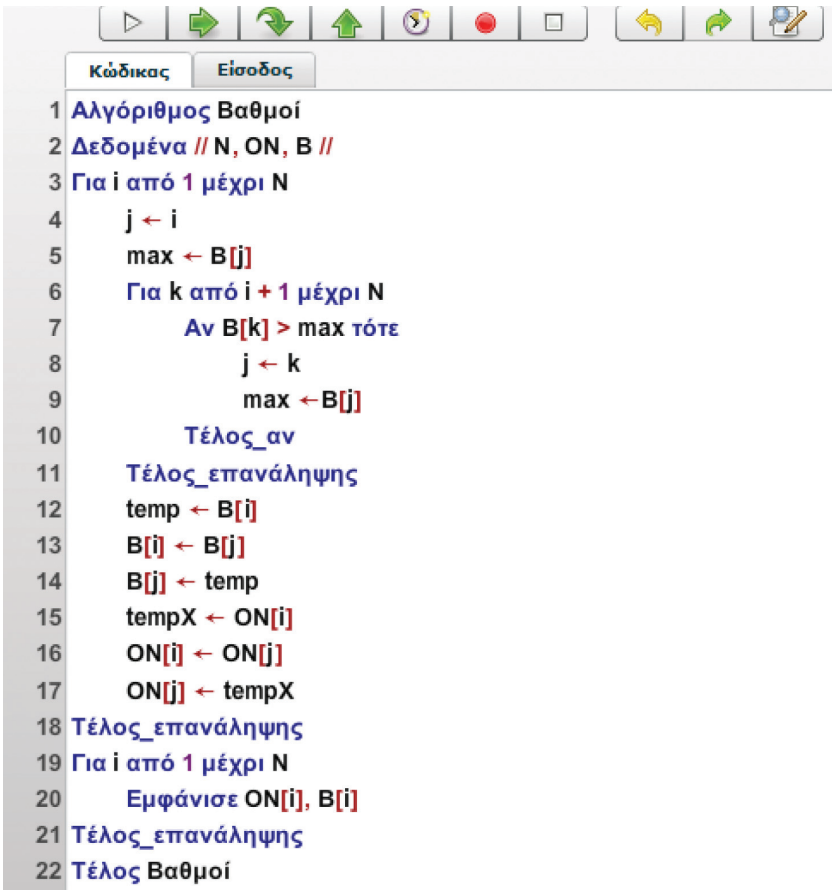
Κώδικας | Είσοδος

```
1  Αλγόριθμος Ραδιόφωνο
2  Σ ← 0
3  π30 ← 0
4  Εμφάνισε "Πληκτρολογήστε το πλήθος των μηνυμάτων"
5  Διάβασε n
6  Για i από 1 μέχρι n
7        Εμφάνισε "Πληκτρολογήστε τη διάρκεια του", i, "ου μηνύματος"
8        Διάβασε χδ
9        Κάλεσε Κόστος(χδ, κμ)
10       Σ ← Σ + κμ
11       Αν χδ > 30 τότε
12             π30 ← π30 + 1
13       Τέλος_αν
14 Τέλος_επανάληψης
15 Εμφάνισε "Τα συνολικά έσοδα του σταθμού είναι",  Σ
16 ποσοστό ← π30 / n * 100
17 Εμφάνισε "Το ποσοστό είναι", ποσοστό, "%"
18 Τέλος Ραδιόφωνο
```

Ο κώδικας του υποπρογράμματος είναι

```
20 Αλγόριθμος Κόστος
21 Δεδομένα // δ //
22 Αν δ ≤ 15 τότε
23       κ ← δ * 90
24 αλλιώς_αν δ ≤ 30 τότε
25       κ ← 15 * 90 + (δ - 15) * 75
26 αλλιώς
27       κ ← 15 * 90 + 15 * 75 + (δ - 30) * 50
28 Τέλος_αν
29 Αποτελέσματα // κ//
30 Τέλος Κόστος
```

Παράδειγμα 2.41. Να γραφεί πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο να δέχεται έναν πίνακα που περιέχει τα ονόματα των μαθητών ενός τμήματος και έναν παράλληλο πίνακα με το βαθμό απολυτηρίου τους. Το πρόγραμμα να ταξινομεί τον πίνακα με τα ονόματα σε φθίνουσα σειρά με βάση τους βαθμούς. Τέλος να εμφανίζει το όνομα του κάθε μαθητή και δίπλα τον βαθμό του.



Παράδειγμα 2.42. Να αναπτυχθεί πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο να διαβάζει τους βαθμούς των μαθητών ενός τμήματος και να εμφανίζει με κατάλληλα μηνύματα το μέσο όρο των βαθμών καθώς και πόσοι από αυτούς είναι μεγαλύτεροι από το μέσο όρο. Να γίνεται έλεγχος ότι οι βαθμοί που εισάγονται είναι ακέραιες τιμές μεταξύ του 0 και του 20. Μετά την εισαγωγή κάθε βαθμού το πρόγραμμα να ερωτά τον χρήστη αν θέλει να συνεχίσει την καταχώρηση εμφανίζοντας το μήνυμα «Θα συνεχίσετε; Πληκτρολογήστε Ν ή Ο» και ανάλογα με την απάντησή του (αποδεκτές τιμές μόνο τα γράμματα του ελληνικού αλφαβήτου «Ν», «ν», «Ο» και «ο»), να συνεχίζει την καταχώριση ή να εμφανίζει τα αποτελέσματα της επεξεργασίας.



Οι πίνακες στους οποίους χρησιμοποιούνται αντίστοιχοι δείκτες θέσης για την αποθήκευση συσχετιζόμενων τιμών ονομάζονται παράλληλοι πίνακες.



Με την εντολή **Δεδομένα** //N, ON, B// γνωστοποιούνται το μέγεθος και οι τιμές των πινάκων στο πρόγραμμα.

Στο πρόγραμμα του παραδείγματος 2.41 εφαρμόζεται ο αλγόριθμος της ταξινόμησης με επιλογή. Έχει τροποποιηθεί όμως σε σχέση με τον αλγόριθμο που παρουσιάστηκε στο προηγούμενο κεφάλαιο, ώστε να βρίσκει κάθε φορά το μέγιστο στοιχείο.



Κατά την ανάπτυξη αυτού του προγράμματος θα χρησιμοποιηθεί η δομή δεδομένων του πίνακα, γιατί διαφορετικά δεν θα υπάρχει η δυνατότητα να γίνει η επεξεργασία για τον υπολογισμό του πλήθους αυτών που είναι πάνω από το μέσο όρο.



Γενικά η χρήση πινάκων είναι απαραίτητη αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης.

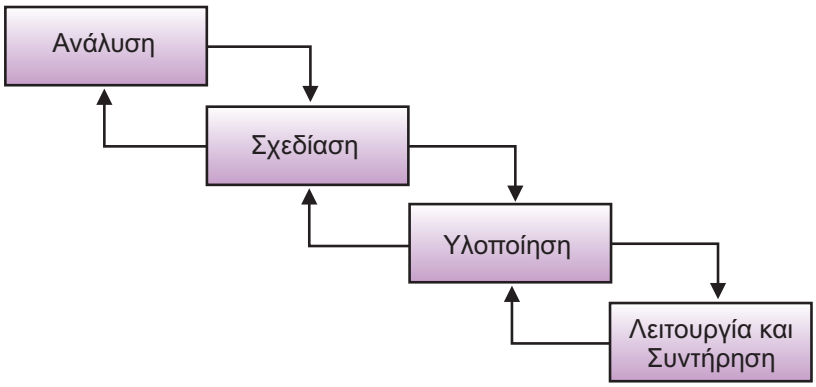


Κώδικας | Είσοδος

```
1  Αλγόριθμος Βαθμοί
2  i ←0
3  Σ ← 0
4  Επανάλαβε
5        i ← i + 1
6        Εμφάνισε "Πληκτρολογήστε τον", i, "ο βαθμό"
7        Επανάλαβε
8              Διάβασε Β[i]
9              Αν Β[i] < 0 ή Β[i] > 20 ή Α_Μ(Β[i]) ≠  Β[i]τότε
10                   Εμφάνισε "Λάθος βαθμός."
11                   Εμφάνισε "Πληκτρολογήστε ξανά τον", i, "ο βαθμό"
12             Τέλος_αν
13       Μέχρις_ότου Β[i] ≥ 0 και Β[i] ≤ 20 και Α_Μ(Β[i]) = Β[i]
14       Σ ← Σ + Β[i]
15       Επανάλαβε
16             Εμφάνισε "Θέλετε να συνεχίσετε; Ναι(Ν ή ν) - Όχι(Ο ή ο)"
17             Διάβασε απ
18       Μέχρις_ότου απ = "Ν" ή απ = "ν" ή απ = "Ο" ή απ = "ο"
19 Μέχρις_ότου απ = "Ο" ή απ = "ο"
20 Ν ← i
21 ΜΟ ←Σ / Ν
22 πλ ← 0
23 Για i από 1 μέχρι Ν
24       Αν Β[i] > ΜΟ τότε
25             πλ ← πλ + 1
26       Τέλος_αν
27 Τέλος_επανάληψης
28 Εμφάνισε "Ο μέσος όρος είναι: ", ΜΟ
29 Εμφάνισε "Υπάρχουν", πλ, "βαθμοί πάνω από το μέσο όρο"
30 Τέλος Βαθμοί
```

2.3.3 Κύκλος ζωής εφαρμογής λογισμικού

Ένα πρόγραμμα αρχίζει την ζωή του από την στιγμή που θα καθοριστούν οι απαιτήσεις του, οι προδιαγραφές του και παύει να ζει όταν εξαντληθούν όλα τα περιθώρια συντήρησής του (προσθήκες, αλλαγές, βελτιώσεις). Ο ***Κύκλος Ζωής*** ενός προγράμματος περιλαμβάνει διάφορες φάσεις οι οποίες αλληλεπιδρούν μεταξύ τους. Οι φάσεις απεικονίζονται στο διάγραμμα της εικόνας 2.34.



Εικόνα 2.34. Κύκλος Ζωής Προγράμματος.

Στη φάση ***Ανάλυσης*** που ακολουθεί τον προσδιορισμό του προβλήματος από τον πελάτη, καταγράφονται αναλυτικά τα δεδομένα και τα ζητούμενα του προβλήματος και ζητούνται οι απαραίτητες διευκρινήσεις από τον πελάτη, σε όσα σημεία οι προδιαγραφές παρουσιάζουν ασάφεια.

Στη φάση της ***Σχεδίασης*** καθορίζεται η δομή του προγράμματος, οι ενότητες (υποπρογράμματα) από τις οποίες θα αποτελείται το πρόγραμμα και αναζητούνται έτοιμες ενότητες (modules) από παλιότερα προγράμματα που μπορούν να χρησιμοποιηθούν και σ' αυτό το πρόγραμμα. Ακόμη επιλέγονται οι αλγόριθμοι και οι δομές δεδομένων που θα χρησιμοποιηθούν σε κάθε ενότητα.

Στην επόμενη φάση της ***Υλοποίησης*** του προγράμματος επιλέγεται η κατάλληλη γλώσσα προγραμματισμού για το συγκεκριμένο πρόγραμμα όπου συντάσσεται το πηγαίο πρόγραμμα και μεταφράζεται από έναν μεταγλωττιστή, ώστε αυτό να γίνει κατανοητό από τον υπολογιστή. Η μετάφραση θα εντοπίσει πιθανά συντακτικά λάθη. Τα λάθη διορθώνονται και ακολουθεί ξανά μετάφραση του προγράμματος, έως την οριστική εξάλειψή τους. Το πρόγραμμα που προκύπτει είναι το εκτελέσιμο πρόγραμμα.

Στην επόμενη φάση της ***Λειτουργίας και Συντήρησης*** θα γίνουν όλες οι προσαρμογές και βελτιώσεις που χρειάζονται προκειμένου το πρόγραμμα να συνεχίσει να χρησιμοποιείται. Αυτές ζητούνται όταν διαφοροποιούνται τα δεδομένα του προβλήματος, όταν ο χρήστης ζητήσει νέες λειτουργίες ή προκύψουν κάποια λογικά λάθη που δεν είχαν διαπιστωθεί στον έλεγχο του αλγορίθμου και μπορεί, για παράδειγμα, να αφορούν την επικοινωνία των ενοτήτων. Για την υλοποίηση των αλλαγών μπορεί να επαναληφθεί η εκτέλεση της φάσης της ανάλυσης και σχεδίασης και άρα όλων των υπολοίπων φάσεων. Έτσι πραγματοποιείται συνολικός έλεγχος του προγράμματος που έχει ως συνέπεια και την καταγραφή σχετικών σχολίων για την τεκμηρίωση.



Στα διάφορα πακέτα λογισμικού, είτε είναι γλώσσες προγραμματισμού, είτε πακέτα εφαρμογών, είτε λειτουργικά συστήματα, δίπλα στο εμπορικό όνομα του λογισμικού, συνηθίζεται να υπάρχει ο αριθμός της έκδοσής του. Ο αριθμός έκδοσης του πακέτου λογισμικού δείχνει ακριβώς τις αλλαγές που έχουν πραγματοποιηθεί από την αρχική του εμφάνιση.

Όταν οι αλλαγές είναι σημαντικές, δηλαδή έχουν προστεθεί νέες λειτουργίες, εντολές, προγράμματα, ο αριθμός έκδοσης αυξάνει κατά ακέραιο αριθμό (GeoGebra 3.0, GeoGebra 4.0), ενώ, όταν οι αλλαγές είναι μικρότερες, αυξάνεται κατά δέκατα ή εκατοστά (GeoGebra 4.1, GeoGebra 4.2). http://www.geogebra.org

Οι εκδόσεις beta μιας εφαρμογής λογισμικού που αναπτύσσει μια εταιρεία δίνονται για έλεγχο συνήθως στους πελάτες της. Οι beta testers παίρνουν την τελική έκδοση δωρεάν ή με σημαντική έκπτωση.



Εικόνα 2.35. Λογότυπο του Λειτουργικού Συστήματος Android.

Στην ιστοσελίδα http://www.code.org υπάρχουν αρκετές δραστηριότητες κατάλληλες για την εκμάθηση οπτικού προγραμματισμού από μαθητές.

Ο συντονιστής του έργου ανάπτυξης μιας εφαρμογής λογισμικού (project manager) δημιουργεί και ενημερώνει το πλάνο εργασίας, στελεχώνει το έργο, και παρακολουθεί και ελέγχει την πρόοδό του.

Πληροφορίες σχετικά με το Λογισμικό Ανοικτού Κώδικα μπορούν να βρεθούν στις διευθύνσεις:
http://www.ellak.gr/
http://www.fsf.org/
http://opensource.org/



Εικόνα 2.36. Λίνους Τόρβαλντς (Linus Torvalds)

Δημιουργός και συντονιστής της ομάδας εργασίας του λειτουργικού συστήματος ανοιχτού κώδικα Linux http://www.linux.org/

Παράδειγμα 2.43. Ανάπτυξη μιας εφαρμογής αριθμομηχανής για κινητά τηλέφωνα με λειτουργικό Android.

Αρχικά στο στάδιο της ανάλυσης θα πρέπει να καταγραφούν οι απαιτήσεις από την αριθμομηχανή, δηλαδή τι είδους πράξεις θα είναι σε θέση να διεκπεραιώσει. Στη συνέχεια, στη φάση της σχεδίασης θα καθοριστούν οι ενότητες από τις οποίες θα αποτελείται η εφαρμογή, όπως ενότητα που θα υλοποιεί τις απλές αριθμητικές πράξεις ή ενότητα που θα υπολογίζει τριγωνομετρικούς αριθμούς ή τιμές μαθηματικών συναρτήσεων. Επίσης θα επιλεγούν οι αλγόριθμοι και οι δομές δεδομένων που θα χρησιμοποιηθούν σε κάθε ενότητα. Στη φάση της υλοποίησης θα πρέπει να επιλεγεί αρχικά η γλώσσα προγραμματισμού στην οποία θα συνταχθεί το πρόγραμμα. Μία γλώσσα οπτικού προγραμματισμού κατάλληλη για ανάπτυξη εφαρμογών για Android είναι η Google AppInventor. Αφού συνταχθούν όλες οι ενότητες, θα γίνει έλεγχος για τη σωστή τους επικοινωνία. Η αριθμομηχανή είναι έτοιμη για να δοθεί στον τελικό χρήστη. Σε περίπτωση που στη φάση της Λειτουργίας και της Συντήρησης κάποιος χρήστης ζητήσει την επέκταση των δυνατοτήτων της αριθμομηχανής, όπως να προστεθεί στατιστική επεξεργασία δεδομένων, τότε θα πρέπει ο προγραμματιστής πιθανότατα να επαναλάβει όλες τις φάσεις ώστε να καλύψει τις νέες απαιτήσεις.

Ωστόσο μια εφαρμογή λογισμικού με σύνθετες λειτουργίες υλοποιείται κατά βάση από μία ομάδα προγραμματιστών που εργάζονται παράλληλα ώστε να μειωθεί ο χρόνος και το κόστος της υλοποίησης. Το κάθε μέλος της ομάδας ανάπτυξης του λογισμικού αναλαμβάνει τη συγγραφή συγκεκριμένων ενοτήτων. Υπάρχουν ταυτόχρονα και μέλη τα οποία είναι υπεύθυνα για το συντονισμό της διαδικασίας ανάπτυξης, τη συνένωση των ενοτήτων σε μία ενιαία εφαρμογή ή για τους ελέγχους του προϊόντος. Τα μεγάλα έργα πληροφορικής εμπλέκουν πολλά διαφορετικά και εξειδικευμένα πρόσωπα, ενώ αποτελούνται από δραστηριότητες των οποίων η σειρά είναι σημαντική.

Οι εφαρμογές που αναπτύσσονται από εταιρείες λογισμικού είναι συνήθως ***Κλειστού Κώδικα*** (proprietary – closed source). Δεν υπάρχει η δυνατότητα για τον χρήστη να δει τον κώδικα ή να επιφέρει κάποια αλλαγή σε αυτόν. Οι βελτιώσεις στα κλειστά λογισμικά γίνονται μέσω αναβαθμίσεων που παρέχονται κατά καιρούς από τις εταιρείες κατασκευής τους, κυρίως μέσω διαδικτύου. Επίσης υπάρχει συνήθως κόστος απόκτησης και απαγόρευση της αναδιανομής της εφαρμογής.

Αντίθετα στις εφαρμογές ***Ελεύθερου Λογισμικού / Λογισμικού Ανοικτού Κώδικα*** (ΕΛ / ΛΑΚ free / open source) ο κώδικας είναι διαθέσιμος, συνεπώς ο καθένας μπορεί ελεύθερα να χρησιμοποιεί, να μελετά τον τρόπο λειτουργίας, να αντιγράφει, να διανέμει και να τροποποιεί την εφαρμογή προσθέτοντας δικές του βελτιώσεις ή νέες λειτουργίες. Με βάση αυτήν τη φιλοσοφία δημιουργούνται ομάδες προγραμμα-

τιστών που μοιράζονται τις αλλαγές που κάνουν στον κώδικα με σκοπό τη βελτίωσή του. Έτσι δημιουργείται ένα παγκόσμιο ανοικτό δίκτυο προγραμματιστών, οι οποίοι συνεργάζονται κυκλοφορώντας νέες εκδόσεις λογισμικού και συμβάλλοντας καθημερινά στη δημιουργία νέων κοινών αγαθών. Το Διαδίκτυο αποτελεί το βασικό μέσο συνεργασίας των προγραμματιστών αλλά και του τρόπου πρόσβασης στο διαθέσιμο Ελεύθερο Λογισμικό.

Ανακεφαλαίωση

Στην ενότητα αυτή επιχειρήθηκε να εξηγηθεί τι είναι πρόγραμμα και έγινε αναφορά στις διάφορες γενιές γλωσσών προγραμματισμού και στα βασικά Προγραμματιστικά Υποδείγματα. Επισημάνθηκαν οι αρχές του Δομημένου Προγραμματισμού και τα πλεονεκτήματα αυτής της μεθοδολογίας συγγραφής προγραμμάτων. Επίσης αναλύθηκε η διαδικασία μετάφρασης των προγραμμάτων και οι τρόποι εύρεσης και διόρθωσης συντακτικών λαθών. Με παραδείγματα ανάπτυξης προγραμμάτων σε διάφορα προγραμματιστικά περιβάλλοντα παρουσιάστηκε η επαναχρησιμοποίηση κώδικα μέσω διαδικασιών ή συναρτήσεων. Ακόμη χρησιμοποιήθηκαν βασικοί αλγόριθμοι του προηγούμενου κεφαλαίου οι οποίοι κωδικοποιήθηκαν σε προγράμματα. Τέλος περιγράφηκε ο Κύκλος Ζωής ενός προγράμματος και έγινε διάκριση μεταξύ των εφαρμογών κλειστού και ανοικτού κώδικα.

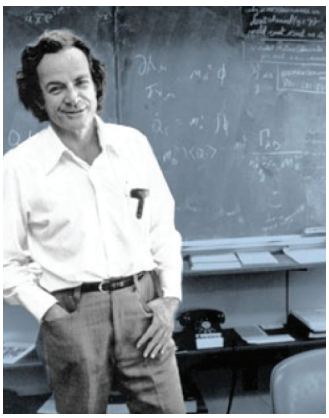
Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Τι είναι πρόγραμμα;
2. Ποια είναι τα πλεονεκτήματα των γλωσσών υψηλού επιπέδου σε σχέση με τις γλώσσες προηγούμενης γενιάς;
3. Ποια η διαφορά των γλωσσών του οπτικού προγραμματισμού με αυτές που διαθέτουν οπτικό περιβάλλον προγραμματισμού;
4. Τι εννοούμε με τη φράση «Προγραμματιστικό Υπόδειγμα»;
5. Τι είναι ο δομημένος προγραμματισμός και ποια είναι τα τρία κύρια χαρακτηριστικά του;
6. Ποια είναι τα πλεονεκτήματα του Δομημένου Προγραμματισμού;
7. Σε τι χρησιμεύουν τα μεταφραστικά προγράμματα;
8. Ποιες οι διαφορές του μεταγλωττιστή και του διερμηνευτή στην εύρεση των συντακτικών λαθών;
9. Τι εργαλεία πρέπει να περιέχει ένα προγραμματιστικό περιβάλλον;
10. Πώς επιτυγχάνεται η επαναχρησιμοποίηση κώδικα και ποια τα πλεονεκτήματά της;
11. Ποιες είναι φάσεις του Κύκλου Ζωής ενός προγράμματος;
12. Ποιες δυνατότητες έχει ο χρήστης με τα Λογισμικά Ανοικτού Κώδικα;
13. Να χαρακτηρίσετε με Σωστό ή Λάθος τις παρακάτω προτάσεις:
    - **Α.** Οι εντολές στις συμβολικές γλώσσες αποτελούνται από ακολουθίες 0 και 1.



Εικόνα 2.37. Σέυμουρ Παπέρτ (Seymour Papert)

Επινόησε τη γλώσσα προγραμματισμού LOGO ως ένα τεχνολογικό εργαλείο που θα βελτιώνει τους τρόπους με τους οποίους τα παιδιά σκέπτονται και επιλύουν προβλήματα.



Εικόνα 2.38. Ρίτσαρντ Φάινμαν (Richard Feynman) Βραβείο Νόμπελ Φυσικής.

«Η επιστήμη υπολογιστών δεν είναι τόσο παλιά όσο η φυσική, υστερεί χρονικά μερικούς αιώνες. Ωστόσο, αυτό δεν σημαίνει ότι υπάρχουν λιγότερα στο πιάτο του επιστήμονα των υπολογιστών απ' ότι σε αυτό του φυσικού: μπορεί να είναι νεότερη αλλά είχε μια πολύ πιο έντονη γέννηση!»



Λέξεις κλειδιά
Πρόγραμμα, Γλώσσες Προγραμματισμού, Προγραμματιστικό Υπόδειγμα, Δομημένος Προγραμματισμός, Συντάκτης, Μετάφραση Προγράμματος, Μεταγλωττιστής, Διερμηνευτής, Αντικείμενο Πρόγραμμα, Συνδέτης, Εκτελέσιμο Πρόγραμμα, Συντακτικά Λάθη, Διαδικασίες, Συναρτήσεις, Βιβλιοθήκες, Κύκλος Ζωής, Ανοικτό Λογισμικό

Β. Η μεταφερσιμότητα είναι χαρακτηριστικό των γλωσσών υψηλού επιπέδου.
Γ. Η SQL αποκρύπτει τις τεχνικές του προγραμματισμού.
Δ. Η γλώσσα LOGO αναπτύχθηκε για εκπαιδευτικού σκοπούς.
Ε. Μία εφαρμογή λογισμικού με σύνθετες λειτουργίες υλοποιείται κατά βάση από μία ομάδα προγραμματιστών.

14. Να επιλέξετε για κάθε μία από τις παρακάτω φράσεις το γράμμα που οδηγεί σε σωστή πρόταση:
Α. Οι εντολές ενός προγράμματος γράφονται σε ένα πρόγραμμα που ονομάζεται:
i. Συντάκτης **ii.** Μεταγλωττιστής
iii. Διερμηνευτής **iv.** Συνδέτης
Β. Ο μεταγλωττιστής επισημαίνει:
i. όλα τα λάθη του προγράμματος
ii. μόνο τα λογικά λάθη του προγράμματος
iii. μόνο τα συντακτικά λάθη του προγράμματος
iv. μόνο τα λάθη που οφείλονται σε αναγραμματισμούς εντολών
Γ. Σε μία εφαρμογή ανοικτού κώδικα:
i. δεν είναι δυνατόν να προσπελάσει ο χρήστης τον κώδικα
ii. πρέπει ο χρήστης να περιμένει τις αναβαθμίσεις από την εταιρεία κατασκευής της.
iii. ο χρήστης δεν μπορεί να προσθέσει λειτουργίες.
iv. ο χρήστης μπορεί να δει τον κώδικα και να προσθέσει λειτουργίες.

15. Να πραγματοποιήσετε συζήτηση στην τάξη για το πώς θα μπορούσε να υλοποιηθεί προγραμματιστικά το υπολογιστικό πρόβλημα που περιγράφεται στο παράδειγμα 2.1. στο κεφάλαιο 2.1.

16. Να πραγματοποιήσετε συζήτηση στην τάξη σχετικά με το επάγγελμα του προγραμματιστή. Αναζητήστε πληροφορίες στη διεύθυνση http://www.eoppep.gr

17. Να αναπτύξετε πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο
Α. να δέχεται τα ονόματα 20 εταιρειών καθώς και τα έσοδα τους για το προηγούμενο έτος.
Β. να εκτυπώνει το μέσο όρο των εσόδων.
Γ. να εκτυπώνει το όνομα της εταιρείας με τα μεγαλύτερα έσοδα.
Δ. να διαβάζει το όνομα μιας εταιρείας και να εμφανίζει τα έσοδα της. Αν η εταιρεία δεν υπάρχει να εμφανίζει κατάλληλο μήνυμα.

18. Να αναπτύξετε κατάλληλα υποπρογράμματα για τα ερωτήματα Β και Γ της δραστηριότητας 17 και να γράψετε τον αντίστοιχο κώδικα του προγράμματος. Τα υποπρογράμματα θα πρέπει να επιστρέφουν το μέσο όρο και το όνομα της εταιρείας αντίστοιχα.



ΕΝΟΤΗΤΑ 3η

Θέματα Εφαρμοσμένης Επιστήμης των Υπολογιστών

ΚΕΦΑΛΑΙΑ

3.1. Λειτουργικά Συστήματα

3.2. Πληροφοριακά Συστήματα

3.3. Δίκτυα

3.4. Τεχνητή Νοημοσύνη

Λειτουργικά Συστήματα

Στόχος του κεφαλαίου αυτού είναι οι μαθητές να εντάξουν τις γνώσεις τους για τα Λειτουργικά Συστήματα στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.

3.1.1 Λογισμικό και Υπολογιστικό Σύστημα

Ένα σύγχρονο υπολογιστικό σύστημα αποτελείται από: το ***υλικό***, τα ηλεκτρονικά μέρη του υπολογιστή και το ***λογισμικό***, το σύνολο των προγραμμάτων που αξιοποιούν και διαχειρίζονται τις λειτουργίες του υλικού του υπολογιστή. Το λογισμικό χωρίζεται στο ***λειτουργικό σύστημα***, το οποίο θα μελετηθεί στη συνέχεια και στο ***λογισμικό εφαρμογών***, τα οποία αποτελούν το σύνολο των προγραμμάτων που επιλύουν τα προβλήματα των χρηστών.

3.1.2 Το Λειτουργικό Σύστημα και οι Αρμοδιότητές του

Λειτουργικό Σύστημα (Λ.Σ.) (Operating System – OS) είναι το σύνολο των προγραμμάτων ενός υπολογιστικού συστήματος το οποίο λειτουργεί ως σύνδεσμος ανάμεσα στα προγράμματα του χρήστη και το υλικό. Το Λ.Σ. είναι υπεύθυνο για τη δημιουργία του περιβάλλοντος επικοινωνίας του χρήστη με το σύστημα, τη διαχείριση και το συντονισμό των εργασιών του συστήματος, καθώς και για την κατανομή των διαθέσιμων πόρων.

Τα Λ.Σ. παρέχουν ένα περιβάλλον στο οποίο εκτελούνται διάφορα προγράμματα, τα οποία στοχεύουν στην ομαλή λειτουργία του υπολογιστικού συστήματος. Οι βασικές αρμοδιότητες ενός Λ.Σ. είναι να:

- Λειτουργεί ως ***ενδιάμεσος μεταξύ του ανθρώπου και της μηχανής***, μεταφέροντας εντολές ή απαιτήσεις του χρήστη στο υπολογιστικό σύστημα.
- ***Διαχειρίζεται τους διαθέσιμους πόρους*** και να τους κατανέμει στις διάφορες διεργασίες.
- Ελέγχει την ***εκτέλεση των προγραμμάτων***.



Προερωτήσεις

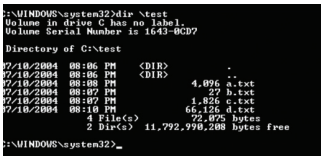
- Πώς συντονίζεται το υλικό σε ένα σύγχρονο υπολογιστικό σύστημα;
- Πώς γίνεται η διαχείριση των πληροφοριών, της μνήμης, της Κεντρικής Μονάδας Επεξεργασίας και των περιφερειακών συσκευών;
- Ποια είναι τα πιο γνωστά Λειτουργικά Συστήματα;
- Ποια είναι τα βασικά χαρακτηριστικά του Λειτουργικού Συστήματος που χρησιμοποιείτε συχνότερα;

Κατά την εξέλιξή τους έχουν εμφανιστεί διάφορες κατηγορίες Λ.Σ.: Τα Λ.Σ. ***ομαδικής επεξεργασίας*** (1η γενιά - δεκαετία ‘50), ***πολυπρογραμματισμού*** (2η γενιά - δεκαετία ‘60), ***καταμερισμού χρόνου*** (3η γενιά - δεκαετία ‘70) και ***κατανεμημένης επεξεργασίας*** (4η γενιά - δεκαετία ‘80 μέχρι σήμερα).

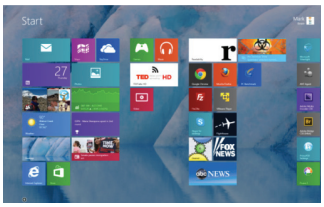


Εικόνα 3.1. Διαστρωμάτωση ενός Λ.Σ.

- Διαχειρίζεται τη ***λειτουργία των συσκευών εισόδου και εξόδου*** και να ελέγχει τη ροή των δεδομένων και την έξοδο των πληροφοριών.
- ***Οργανώνει*** και να ***διαχειρίζεται τα αρχεία*** του συστήματος.
- ***Ανιχνεύει*** και να ***εντοπίζει*** πιθανά λάθη ή δυσλειτουργίες του υπολογιστικού συστήματος και να ενημερώνει τον χρήστη.
- Εφαρμόζει μηχανισμούς που βελτιώνουν την ***ασφάλεια*** του υπολογιστικού συστήματος από διάφορους κινδύνους.



Εικόνα 3.2. Επικοινωνία με απευθείας εντολές (Λ.Σ. MS DOS)



Εικόνα 3.3. Επικοινωνία με γραφικό περιβάλλον Διεπαφής (Λ.Σ. **MS Windows**)

3.1.3 Η Δομή και η Ιεραρχία του Λειτουργικού Συστήματος

Τα σύγχρονα Λ.Σ. είναι δομημένα σε ιεραρχικά τοποθετημένα ***επίπεδα*** (layers). Κάθε επίπεδο εκτελεί μια συγκεκριμένη εργασία και συνεργάζεται με τα δύο γειτονικά του. Στα κατώτερα επίπεδα γίνεται η διαχείριση της μνήμης και της επικοινωνίας με τις περιφερειακές συσκευές του υπολογιστή, ενώ στα ανώτερα γίνεται η διαχείριση των προγραμμάτων που εκτελούν οι χρήστες. Σε ένα Λ.Σ. υπάρχουν τα ακόλουθα επίπεδα (Εικόνα 3.1):

- Ο **Πυρήνας** (Kernel), βρίσκεται πλησιέστερα προς το υλικό και αποτελεί τον ενδιάμεσο για να επιτευχθεί η επικοινωνία των προγραμμάτων με το υλικό. Ο πυρήνας «φορτώνεται» πρώτος στην κύρια μνήμη όταν ξεκινάει ο υπολογιστής.
- Το **Σύστημα Αρχείων** (File System) διαχειρίζεται τα αρχεία (δίνοντάς τους ονομασία, καταχωρώντας τα, κτλ.) και φροντίζει για τη διάθεσή τους στους χρήστες.
- Ο **Διερμηνευτής Εντολών** (Command Interpreter) ή **Φλοιός** (Shell) είναι το σύνολο των προγραμμάτων, το οποίο επιτρέπει στο χρήστη και τις εφαρμογές του να επικοινωνεί με το Λ.Σ. Η επικοινωνία γίνεται είτε με ***απευθείας εντολές*** (command mode - Εικόνα 3.2) είτε μέσω ενός ***γραφικού περιβάλλοντος διεπαφής*** (GUI - Graphical User Interface – Εικόνα 3.3).

3.1.4 Βασικές Εργασίες ενός Λ.Σ.

Εργασίες ενός Λ.Σ. αποτελούν η ***Διαχείριση της Κεντρικής Μονάδας Επεξεργασίας (Κ.Μ.Ε.)***, η ***Διαχείριση της Κεντρικής Μνήμης***, η ***Διαχείριση του Συστήματος Αρχείων*** και η ***Διαχείριση των Λειτουργιών Εισόδου/Εξόδου***.

3.1.4.1 Διαχείριση της Κ.Μ.Ε.

Το χαρακτηριστικό των σύγχρονων Λ.Σ. είναι ο ***πολυπρογραμματισμός*** (multiprogramming) και η ***πολυδιεργασία*** (multitasking). Στην πρώτη περίπτωση το Λ.Σ. μπορεί να εκτελεί ταυτόχρονα περισσότερα από ένα προγράμματα (για παράδειγμα, την ώρα που η Κ.Μ.Ε. περιμένει



Μια από τις βασικές έννοιες στα Λ.Σ. είναι η έννοια της ***διεργασίας*** (process).

Μια διεργασία είναι ένα πρόγραμμα ή ένα αυτόνομο τμήμα προγράμματος υπό εκτέλεση από το σύστημα.
Το ***πρόγραμμα*** διαφέρει από τη διεργασία στο ότι είναι μικρότερο, μιας και η διεργασία περιέχει, εκτός από το πρόγραμμα, και στοιχεία συστήματος για την εκτέλεση του κώδικα.

απάντηση από μια περιφερειακή συσκευή, αυτός ο χρόνος «αναμονής» μπορεί να αξιοποιηθεί από ένα άλλο πρόγραμμα που είναι φορτωμένο στην κύρια μνήμη), ενώ στη δεύτερη περίπτωση το Λ.Σ. μπορεί να εκτελεί ταυτόχρονα περισσότερες από μία εργασίες (για παράδειγμα, μπορεί να γίνονται παράλληλα εκτυπώσεις και υπολογισμοί). Η όλη διαδικασία βασίζεται σε έναν ***αλγόριθμο χρονοπρογραμματισμού***, ο οποίος στοχεύει στη μεγιστοποίηση της αποδοτικότητας και της «***δίκαιης***» χρήσης της Κ.Μ.Ε. από το μέγιστο αριθμό των επεξεργαζόμενων διεργασιών.

3.1.4.2 Διαχείριση της Μνήμης



Κάθε πρόγραμμα προτού εκτελεστεί από την κεντρική μνήμη βρίσκεται στην περιφερειακή μνήμη (σκληρός δίσκος), ενώ κατά τη διάρκεια εκτέλεσής του στην κεντρική μνήμη το πρόγραμμα διαβάζει από αυτήν και γράφει δεδομένα σε αυτήν.

Η πολυδιεργασία προϋποθέτει ότι στην κεντρική μνήμη είναι φορτωμένα περισσότερα του ενός προγράμματα προς εκτέλεση από την Κ.Μ.Ε. Θα πρέπει, λοιπόν, να γίνει η διαχείριση της κύριας μνήμης με τέτοιο τρόπο, ώστε να επιτευχθεί ο αποτελεσματικός διαμοιρασμός της μεταξύ των διαφόρων προγραμμάτων. Το τμήμα του Λ.Σ. που διαχειρίζεται την κύρια μνήμη είναι ο ***διαχειριστής μνήμης*** (memory manager). Οι εργασίες που επιτελεί ο διαχειριστής μνήμης είναι:

- Η διάθεση τμημάτων μνήμης σε διεργασίες.
- Η παρακολούθηση της κατάστασης χρήσης της μνήμης, ώστε να γνωρίζει τα ελεύθερα ή μη τμήματα κάθε στιγμή και να τα διανέμει σε διεργασίες.
- Η ελευθέρωση μνήμης από διεργασίες που δεν τη χρειάζονται.
- Η ***ανταλλαγή*** (swapping) ***δεδομένων*** μεταξύ της κύριας μνήμης και της περιοχής του δίσκου (περιφερειακή μνήμη) που χρησιμοποιείται ως βοηθητική περιοχή της κύριας μνήμης.

3.1.4.3 Διαχείριση του Συστήματος Αρχείων



Σ' έναν υπολογιστή οι πληροφορίες αποθηκεύονται σε περιφερειακές μονάδες (π.χ. σε σκληρούς δίσκους) όπου τα στοιχεία διατηρούνται και μετά τη διακοπή της παροχής ρεύματος. Εκεί αποθηκεύονται τα στοιχεία ως συλλογές δεδομένων (αρχεία-files), τη διαχείριση των οποίων αναλαμβάνει μέρος του Λ.Σ. που καλείται ***Σύστημα Αρχείων*** (file system).

Το σύστημα αρχείων είναι το μέρος του Λ.Σ. με το οποίο ο χρήστης έρχεται σε άμεση επαφή. Το Λ.Σ. συνήθως οργανώνει τα αρχεία του σε ***καταλόγους*** ή ***φακέλους*** (directories ή folders). Κάθε κατάλογος αποτελείται από αρχεία, υποκαταλόγους ή υποφακέλους, δημιουργώντας μία ***δενδροειδή μορφή***.

Ένα σύστημα αρχείων του Λ.Σ. προσφέρει στον χρήστη ένα εικονικό περιβάλλον διαχείρισης, το οποίο του δίνει τη δυνατότητα να εκτελεί μία σειρά από πράξεις όπως η δημιουργία (με προσδιορισμό ονόματος και τύπου), η διαγραφή, η μετονομασία, η αντιγραφή και το κλείσιμο αρχείων. Επιπλέον, είναι δυνατή η τροποποίηση του περιεχομένου ή η αντιγραφή του περιεχομένου ενός αρχείου σε ένα άλλο.

3.1.4.4 Διαχείριση Λειτουργιών Εισόδου/Εξόδου

Με τον όρο ***Είσοδος*** (Input) αναφερόμαστε στη ροή δεδομένων προς την Κ.Μ.Ε. ενώ με τον όρο ***Έξοδος*** (Output) αναφερόμαστε στη ροή δεδομένων από την Κ.Μ.Ε. προς τις περιφερειακές συσκευές.

Το τμήμα του Λ.Σ. το οποίο ασχολείται με τις ***διαδικασίες εισόδου/εξόδου*** μεταξύ του κεντρικού μέρους και των εξωτερικών προς αυτό συσκευών εξασφαλίζει τη διασύνδεση των συσκευών με το κεντρικό

σύστημα και την ομαλή επικοινωνία. Επίσης, αναλαμβάνει να διαχειριστεί τις εντολές που εκτελούνται και τα σφάλματα που παρουσιάζονται.

Στις αρμοδιότητες του συγκεκριμένου μέρους του Λ.Σ. είναι η αποδοτική διαχείριση των ***περιφερειακών μονάδων*** και ο ορισμός της σειράς ικανοποίησης των διαφόρων δραστηριοτήτων, όπως των αιτημάτων εγγραφής ή ανάγνωσης. Έτσι, οι συσκευές διακρίνονται σε ***διαμοιραζόμενες*** (shared) και ***αποκλειστικές*** (dedicated). Τις διαμοιραζόμενες συσκευές μπορούν να τις χρησιμοποιούν πολλοί χρήστες ταυτόχρονα, όπως οι δίσκοι, ενώ τις αποκλειστικές μπορεί να τις χρησιμοποιεί ένας χρήστης κάθε στιγμή, όπως οι εκτυπωτές οι οποίοι κατά τη διάρκεια της χρήσης τους δεν επιτρέπεται να χρησιμοποιούνται από άλλο πρόγραμμα.



Εικόνα 3.4. Το Λ.Σ. ***Linux*** είναι ΕΛ/ΛΑΚ και μπορεί να τρέξει μέσα από οπτικό δίσκο (LiveCD) χωρίς να απαιτείται μόνιμη εγκατάσταση.

3.1.5 Γνωστά Λειτουργικά Συστήματα

Έχουν αναπτυχθεί διάφορα Λ.Σ. τα οποία αξιοποιούν κατάλληλα την αρχιτεκτονική των επεξεργαστών. Στη συνέχεια παρατίθενται τα πιο γνωστά από αυτά.



Εικόνα 3.5. Λ.Σ. ***OS X*** της Apple.

Το **Unix** αναπτύχθηκε στα εργαστήρια Bell Labs της AT&T το 1969. Το μεγαλύτερο μέρος του έχει αναπτυχθεί σε γλώσσα προγραμματισμού C. Το Unix αποτελεί ένα Λ.Σ. πολυδιεργασίας, καταμερισμού χρόνου (timesharing), πολλών χρηστών (multiuser), και με *φορητότητα* (portability) - δηλαδή εύκολη εγκατάσταση σε διαφορετικού τύπου συστήματα.

Το **MS-DOS** (Microsoft Disk Operating System) είναι Λ.Σ. στο οποίο οι εντολές του χρήστη πληκτρολογούνται σε μία γραμμή. Αναπτύχθηκε από την εταιρεία Microsoft το 1981 για λογαριασμό της εταιρείας IBM (Εικόνα 3.2).



Εικόνα 3.6. Λ.Σ. ***Android*** της Google

Τα **MS Windows** της Microsoft διαθέτουν γραφικό περιβάλλον διεπαφής χρήστη παραθυρικού τύπου (Εικόνα 3.3). Πρόκειται για Λ.Σ. πολλαπλών διεργασιών και πολλαπλών χρηστών. Από το 1985 μέχρι σήμερα έχουν αναπτυχθεί διάφορες εκδόσεις των Windows (π.χ. 95, 98, XP, 7, 8.1 κ.α.).

Το **Linux** είναι ένα λειτουργικό σύστημα τύπου Unix, το οποίο δημιουργήθηκε από τον Linus Torvalds το 1991. Ο πηγαίος κώδικάς του είναι «ανοικτός», με αποτέλεσμα να αναπτύσσεται συνεχώς από μια μεγάλη κοινότητα χρηστών (Εικόνα 3.4).

Το **Mac OS X** έχει ως βάση το Λ.Σ. Unix και διανέμεται αποκλειστικά για χρήση με τους υπολογιστές Mac της εταιρείας Apple. Το Mac OS X είναι η εξέλιξη του Mac OS το οποίο ήταν το αρχικό Λ.Σ. της Apple (1984-1999). Τα OS X διακρίνονται για τα πρωτότυπα γραφικά περιβάλλοντα διεπαφής τους (Εικόνα 3.5).

Το **Android** χρησιμοποιείται κυρίως σε συσκευές κινητής τηλεφωνίας και tablets και τρέχει τον πυρήνα του Λ.Σ. Linux. Αρχικά αναπτύχθη-

κε από την Google (2007) και αργότερα από την Open Handset Alliance (Εικόνα 3.6).

Ανακεφαλαίωση

Ένα Λ.Σ αποτελείται από ένα σύνολο προγραμμάτων τα οποία ελέγχουν ένα υπολογιστικό σύστημα, και λειτουργεί ως ενδιάμεσος ανάμεσα στο σύστημα και τον χρήστη. Τα χαρακτηριστικά των συγχρόνων Λ.Σ. είναι ο πολυπρογραμματισμός και η πολυδιεργασία. Έχουν αναπτυχθεί διάφορα Λ.Σ. τα οποία εξελίσσονται παράλληλα με την εξέλιξη της αρχιτεκτονικής των σύγχρονων επεξεργαστών.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Ποιο ρόλο επιτελεί το Λειτουργικό Σύστημα σε έναν υπολογιστή;
2. Να αναφέρετε ονομαστικά τις μεθόδους διαχείρισης των περιφερειακών συσκευών από το Λ.Σ.
3. Δείτε ποια Λ.Σ. διαθέτει το εργαστήριο υπολογιστών του σχολείου σας και εντοπίστε ομοιότητες και διαφορές.
4. Αν στην τάξη σας βρίσκονται μαθητές με διαφορετική καταγωγή, μελετήστε τις ασυμβατότητες των Λ.Σ. μεταξύ διαφορετικών γλωσσών.
5. Αναζητήστε το LiveCD του Linux και δοκιμάστε το σε κάποιους υπολογιστές του σχολικού εργαστηρίου.
6. Αναζητήστε πληροφορίες για Λ.Σ. τα οποία είναι φιλικά στη χρήση τους από Άτομα με Αναπηρία (ΑμεΑ).
7. Ποια από τα παρακάτω αποτελούν εργασίες ενός Λ.Σ.;
   Α. Εκτύπωση
   Β. Διαχείριση μνήμης
   Γ. Επεξεργασία κειμένου
   Δ. Διαχείριση Κ.Μ.Ε.



Χρήσιμοι Υπερσύνδεσμοι

http://www.android.com
Διαδικτυακός τόπος για το Λ.Σ. Android.

http://www.linux.gr/news
Διαδικτυακός τόπος της ελληνικής κοινότητας Linux.



Λέξεις κλειδιά
Διεπαφή, Σύστημα Αρχείων, Διαχείριση Κ.Μ.Ε., Είσοδος/Έξοδος, Πυρήνας.

Πληροφοριακά Συστήματα

Στόχοι του κεφαλαίου είναι οι μαθητές:

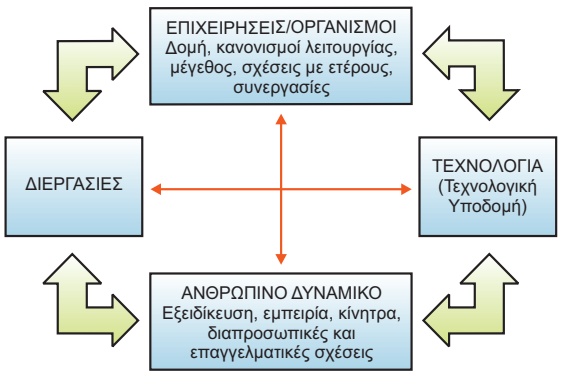
- ✓ Να εντάξουν τις γνώσεις τους για θέματα σχετικά με τη διαχείριση δεδομένων, τη δημιουργία, την αποθήκευση και την ανάκτηση πληροφοριών στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.
- ✓ Να αιτιολογούν ότι τα δεδομένα αποθηκεύονται σε οργανωμένες δομές και ανακτώνται μέσω συγκεκριμένων συστημάτων και μεθοδολογιών.

3.2.1 Τι είναι τα Πληροφοριακά Συστήματα

Τα τελευταία χρόνια έχουν διαμορφωθεί νέες απαιτήσεις για θέματα που αφορούν τη διαχείριση των δεδομένων, τη δημιουργία, αποθήκευση, ανάκτηση και χρήση της πληροφορίας. Οι απαιτήσεις αυτές οδηγούν στην ανάπτυξη βάσεων δεδομένων και νέων πληροφοριακών αρχιτεκτονικών. Τα Πληροφοριακά Συστήματα και οι Βάσεις Δεδομένων εντάσσονται στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.

Πληροφοριακό σύστημα (Π.Σ.) ονομάζεται ένα σύνολο αλληλοσυσχετιζόμενων και αλληλεπιδρώντων οντοτήτων που συλλέγουν, επεξεργάζονται, αποθηκεύουν, ανακτούν και διανέμουν πληροφορίες για την υποστήριξη των αποφάσεων και ελέγχου σε μια επιχείρηση ή σε έναν οργανισμό.

Ένα Π.Σ. λειτουργεί μέσα σε ένα περιβάλλον, επηρεάζεται από αυτό και το επηρεάζει. Μέσα σ' αυτό λειτουργεί και το ανθρώπινο δυναμικό με την τεχνολογική υποδομή που διαθέτει (Εικόνα 3.7).



Εικόνα 3.7. Το μοντέλο αλληλεπίδρασης ενός Π. Σ.



Προερωτήσεις

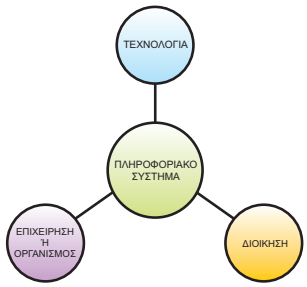
- Ποια είναι τα στάδια υλοποίησης μιας παραγγελίας μέσω διαδικτύου;
- Πώς επιδρά η εξάπλωση του Διαδικτύου και των δικτύων στη λειτουργία ενός οργανισμού ή μιας επιχείρησης;
- Πώς εκτελούνται οι λειτουργίες παραγγελιών, παραγωγής και μεταφοράς προϊόντων σε συνθήκες παγκοσμιοποίησης;



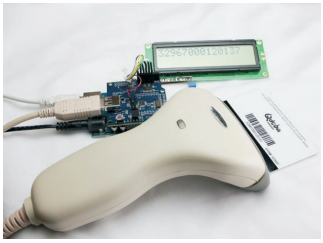
Η οντότητα (**entity**) είναι μία αυτόνομη μονάδα του φυσικού κόσμου, για παράδειγμα ένα αντικείμενο, ένα πρόσωπο, μία κατάσταση και γενικά οτιδήποτε μπορεί να προσδιοριστεί ως ανεξάρτητη ύπαρξη. Για παράδειγμα, σε ένα σχολείο, οντότητες μπορεί να είναι τα ονόματα ή τα στοιχεία των μαθητών, των καθηγητών, οι αίθουσες διδασκαλίας, τα βιβλία κάθε τάξης, οι βαθμολογίες κ.ά.



Για μια επιχείρηση που διανέμει διάφορα είδη προϊόντων ο προγραμματισμός των εργασιών της, η παράδοση των αγαθών και τα έξοδα διανομής καθορίζουν σε μεγάλο βαθμό την κερδοφορία της. Έτσι, η εφοδιαστική αλυσίδα (**logistics**) παίζει σπουδαίο ρόλο στις δραστηριότητές της. Η αξιοποίηση ενός Π.Σ. γύρω από αυτά τα ζητήματα είναι καθοριστικής σημασίας για την επιχείρηση.



Εικόνα 3.8. Τα συστατικά μέρη ενός Π.Σ.



Εικόνα 3.9. Συσκευή γραμμοκώδικα με χρήση πλακέτας Arduino.

Οι σύγχρονοι εξυπηρετητές διαθέτουν πλατφόρμα δεδομένων άμεσης πρόσβασης για διαχείριση και δόμηση επιχειρησιακών λύσεων βασισμένες στο σύννεφο.

Σήμερα, κάθε λογαριασμός μπορεί να περιέχει έως και 100TB από Blobs (Binary Large OBject). Ένα Blob μπορεί να έχει μέγεθος εκατοντάδων gigabyte.



Εικόνα 3.10. Μία Β.Δ. με τρεις συλλογές δεδομένων.

Κάθε Π.Σ. αποτελείται από ***Δεδομένα ή Διαδικασίες που αφορούν την επιχείρηση ή τον οργανισμό, Τεχνολογίες Πληροφορικής (υλικό και λογισμικό) και Ανθρώπους (ανθρώπινο δυναμικό και διοίκηση)*** (Εικόνα 3.8).

Σε ένα Π.Σ. συντελείται: ***Συλλογή Δεδομένων, Αποθήκευση Δεδομένων, Επεξεργασία Δεδομένων και Παρουσίαση της Πληροφορίας.*** Η διαχείριση των δεδομένων ενός Π.Σ. είναι σημαντική για έναν οργανισμό ή μια επιχείρηση. Για το λόγο αυτό απαιτείται η δημιουργία ειδικής υπηρεσίας που θα καθορίζει α) τις ανάγκες και β) την δυνατότητα πρόσβασης σε δεδομένα και πληροφορίες.

3.2.2 Αρχιτεκτονικές Αποθήκευσης

Κάθε επιχείρηση ακολουθεί διαφορετικούς τρόπους αποθήκευσης των δεδομένων σύμφωνα με τις ανάγκες της και τις τεχνολογίες που μπορεί να υποστηρίξει. Οι αρχιτεκτονικές αποθήκευσης κατηγοριοποιούνται ανάλογα με τις μεθόδους που χρησιμοποιούν. Στις ***αρχιτεκτονικές αποθήκευσης άμεσης πρόσβασης*** είναι δυνατή η άμεση πρόσβαση στα δεδομένα ή τα εμπορεύματα μέσω συσκευών εισόδου. Οι καταχωρήσεις στο σύστημα αποθήκευσης μπορούν να γίνονται με συσκευές αναγνώρισης (Εικόνα 3.9). Στις ***Αρχιτεκτονικές αποθήκευσης βασισμένες σε δίκτυο*** η πρόσβαση στα δεδομένα γίνεται μέσω ενός δικτύου υπολογιστών, όπου τα δεδομένα αποθηκεύονται είτε τοπικά είτε σε κάποια άλλη τοποθεσία της επιχείρησης. Έτσι δίνεται η δυνατότητα απομακρυσμένης πρόσβασης σε όλα τα δεδομένα. Οι ***Αρχιτεκτονικές αποθήκευσης βασισμένες στο σύννεφο*** στηρίζονται στο Διαδίκτυο και χρησιμοποιούν τις υποδομές ενός παρόχου υπηρεσιών σύννεφου (***Cloud Service Provider, CSP***). Περιλαμβάνουν λειτουργικό σύστημα για υπηρεσίες σύννεφου και τις απαραίτητες διαδικτυακές υπηρεσίες που εξασφαλίζουν τη διασυνδεσιμότητα και τη διαλειτουργικότητα. Για τη συγκεκριμένη αρχιτεκτονική υπάρχει η δυνατότητα αύξησης σε πραγματικό χρόνο του αποθηκευτικού χώρου και διατίθενται υπηρεσίες που επιτρέπουν την αποθήκευση μεγάλων ποσοτήτων ***μη δομημένης πληροφορίας***, η οποία είναι προσβάσιμη μέσω του διαδικτύου, χωρίς την ανάγκη τοπικής αποθήκευσης και επεξεργασίας.

3.2.3 Βάσεις Δεδομένων

Η αποτελεσματική και αποδοτική διαχείριση των δεδομένων ενός Π.Σ. απαιτεί την οργάνωσή τους με δομημένο τρόπο. Για το σκοπό αυτό χρησιμοποιούνται ***Βάσεις Δεδομένων*** (***Data Bases, D.B.***) που αποτελούν το λειτουργικό πυρήνα του Π.Σ.. Σε μία Β.Δ. καταγράφονται ολοκληρωμένες συλλογές δεδομένων που συσχετίζονται (Εικόνα 3.10).

Βάση Δεδομένων (Β.Δ.) είναι μία οργανωμένη συλλογή από συσχετιζόμενα δεδομένα, επεξεργασμένα και αποθηκευμένα με τέτοιο τρόπο, ώστε να μπορούν να χρησιμοποιούνται σε όλες τις εφαρμογές ενός οργανισμού ή μιας επιχείρησης.

Η πιο συνηθισμένη μορφή Β.Δ. είναι οι ***Σχεσιακές Βάσεις Δεδομένων (Σ.Β.Δ.)***, όπου τα δεδομένα οργανώνονται σε συσχετιζόμενους πίνακες με βάση το Σχεσιακό Μοντέλο Δεδομένων. Για παράδειγμα, στην εικόνα 3.11 το κοινό στοιχείο των 2 πινάκων είναι ο κωδικός του μαθήματος.

Στιγμιότυπο Πίνακα Μαθημάτων

Κωδικός_Μαθήματος	**Όνομα_Μαθήματος**	**Διδακτικές_Ώρες**
20	Εφαρμογές Πληροφορικής	2
30	Εισαγωγή στην Επιστήμη των Η/Υ	1
…	…	…

Στιγμιότυπο Πίνακα Καθηγητών

Αριθμός_Μητρώου	**Επώνυμο**	**Όνομα**	**…**	**Κωδικός_Μαθήματος**
7329	Επώνυμο1	Όνομα1		20
7499	Επώνυμο2	Όνομα2		30
7384	Επώνυμο3	Όνομα3		20
….	….	….		….

Εικόνα 3.11. Σχεσιακό Μοντέλο Δεδομένων με ένα κοινό στοιχείο.

Η διαχείριση των συσχετιζόμενων δεδομένων μίας Β.Δ. γίνεται με ένα σύνολο κατάλληλων προγραμμάτων το οποίο καλείται ***Σύστημα Διαχείρισης της Βάσης Δεδομένων - ΣΔΒΔ (Database Management System, DBMS)*** (Εικόνα 3.12).



Εικόνα 3.12. Δεδομένα και Σ.Δ.Β.Δ.

Τα συστήματα διαχείρισης βάσεων δεδομένων αλληλεπιδρούν με τον χρήστη μέσα από γλώσσες ερωταποκρίσεων.

3.2.4 Γλώσσες Ερωτοαποκρίσεων (SQL, XML)

Μία από τις δημοφιλέστερες γλώσσες ερωτοαποκρίσεων είναι η ***SQL (Structured Query Language, Δομημένη Γλώσσα Ερωταποκρίσεων),*** η οποία αναπτύχθηκε το 1971 από την IBM.

Άλλη μια σημαντική γλώσσα ερωτοαποκρίσεων είναι η ***XML, (Extensible Markup Language, Επεκτάσιμη Γλώσσα Σήμανσης).*** Η ***XML*** χρησιμοποιώντας ένα σύνολο κανόνων καθιστά εφικτή τη δημιουργία οποιασδήποτε ετικέτας απαιτείται για την περιγραφή των δεδομένων και της δομής τους (Εικόνα 3.13).

Η δομή των δεδομένων όπως αυτά αναπαρίστανται από την XML, οπτικά έχει τη μορφή ***ενός ανάποδου δέντρου*** το οποίο ξεκινάει από μια ρίζα και εκτείνεται προς τα κάτω με πολλαπλά κλαδιά. Η XML περιλαμβάνει τη γλώσσα ερωταποκρίσεων "Xpath", η οποία επιτρέπει την διατύπωση ερωτημάτων σε μορφή ***έκφρασης διαδρομής*** ή μονο-

Σχεσιακό Μοντέλο Δεδομένων (Relation Data Model, RDM) είναι το είδος του λογικού μοντέλου βάσεων δεδομένων που μπορεί να συνδυάζει τα δεδομένα ενός πίνακα με τα δεδομένα ενός άλλου, αρκεί οι δυο πίνακες να έχουν ένα κοινό στοιχείο δεδομένων.

```
SELECT ENAME, JOB, SAL
FROM EMPLOYES WHERE
DEPTNO = 20
AND SAL > 1000;
```

Με την ερώτηση αυτή σε SQL εκτελείται αναζήτηση στη βάση δεδομένων EMPLOYES και επιστρέφει το όνομα, τη θέση και τον μισθό των υπαλλήλων της διεύθυνσης 20 που κερδίζουν πάνω από 1000 ευρώ.

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
 <book>
  <title>Networks</title>
  <copies>15</copies>
 </book>
 <book>
  <title>Algorithms</title>
  <copies>8</copies>
 </book>
</library>
```

Εικόνα 3.13. Τμήμα XML κειμένου για μία δανειστική βιβλιοθήκη.

```
/library/book[copies>10]/title
```

Με αυτή την έκφραση διαδρομής επιλέγονται και επιστρέφονται οι τίτλοι των βιβλίων της βιβλιοθήκης με περισσότερα από 10 αντίτυπα. Από την παραπάνω βιβλιοθήκη θα επιστραφεί ο τίτλος Networks.



Χρήσιμοι Υπερσύνδεσμοι
http://www.w3.org/standards/xml/core
Διαδικτυακός τόπος για τις προδιαγραφές **xml.**

http://www.w3schools.com/XPath/
Διαδικτυακός τόπος των προδιαγραφών **Xpath.**



Λέξεις κλειδιά
Αρχιτεκτονικές Αποθήκευσης, Γλώσσες Ερωταποκρίσεων, Συστήματα αποθήκευσης πληροφοριών, Γλώσσες Σήμανσης, Δικτυακές Σχεσιακές Βάσεις Δεδομένων.

πατιού (path expression) και αποκρίνεται επιστρέφοντας τα δεδομένα τα οποία πληρούν τα κριτήρια που έχει θέσει ο χρήστης.

Ανακεφαλαίωση

Στα Π.Σ. η οργάνωση και η διαχείριση δεδομένων σε επιχειρησιακή βάση έχει εμπλουτιστεί με τη χρήση βάσεων δεδομένων και τεχνικές βασισμένες στο δίκτυο. Η αξιοποίηση των δυνατοτήτων του σύννεφου (cloud) επιτρέπει την ανάπτυξη ενός οργανισμού ή μιας επιχείρησης χωρίς επενδύσεις σε υπολογιστική υποδομή. Οι γλώσσες ερωτοαποκρίσεων ενός συστήματος διαχείρισης βάσεων δεδομένων όπως η SQL και η XML αξιοποιούνται σε ένα ευρύ φάσμα επιχειρησιακών εφαρμογών.

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Οργανώστε δεδομένα και πληροφορίες που αφορούν την τάξη σας ή το σχολείο σας (πόσοι μαθητές, πόσες μαθήτριες, πόσες τάξεις) και διαμορφώστε ένα μικρό αληθινό Π.Σ. που να αφορά τη ζωή στο σχολείο.
2. Ενημερωθείτε από συμμαθητές σας για τα συστήματα οργάνωσης των δημοσίων οργανισμών άλλων χωρών.
3. Ποια τα πλεονεκτήματα της χρήσης και της αξιοποίησης της τεχνικής αποθήκευσης δεδομένων στο σύννεφο;
4. Με ποιες γλώσσες προγραμματίζονται τα σύγχρονα περιβάλλοντα των Πληροφοριακών Συστημάτων;
5. Επισκεφτείτε μια επιχείρηση της περιοχής σας και ενημερώστε τους συμμαθητές σας για τη διαδικασία οργάνωσής της. Πώς οργανώνονται οι προμήθειές της, πώς γίνεται η διακίνηση των εμπορευμάτων ή των υπηρεσιών που παρέχει και πώς έχει οργανωθεί το τμήμα του λογιστηρίου**.**
6. Να χαρακτηρίσετε με Σωστό ή Λάθος τις παρακάτω προτάσεις:
   - **Α.** Ένα Π.Σ. λειτουργεί μέσα σε ένα περιβάλλον που το επηρεάζει και από το οποίο επηρεάζεται.
   - **Β.** Στις αρχιτεκτονικές αποθήκευσης βασισμένες στο σύννεφο είναι αναγκαία η τοπική αποθήκευση των δεδομένων.
   - **Γ.** Η διαχείριση των συσχετιζόμενων δεδομένων μίας Β.Δ. γίνεται μέσω του συστήματος διαχείρισης της Β.Δ..
   - **Δ.** Μέσω του σχεσιακού μοντέλου δεδομένων συνδυάζονται τα δεδομένα ενός πίνακα με τα δεδομένα ενός άλλου, αρκεί να έχουν ένα κοινό στοιχείο δεδομένων.
   - **Ε.** Τα Π.Σ. αναπτύσσονται με σκοπό την υποστήριξη των αποφάσεων και τον έλεγχο μιας επιχείρησης.



Δίκτυα

Στόχοι του κεφαλαίου είναι:

- ✓ Οι μαθητές να εντάξουν τις γνώσεις τους για θέματα επικοινωνίας και δικτύωσης συστημάτων στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.
- ✓ Οι μαθητές να οργανώσουν σε νοητικό μοντέλο τα βασικά θέματα που αφορούν τα δίκτυα επικοινωνίας.



Προερωτήσεις

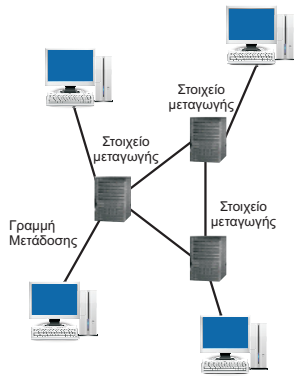
- Πώς επιτυγχάνεται η επικοινωνία των υπολογιστικών συστημάτων;
- Ποια υποδομή απαιτείται για να επιτευχθεί η επικοινωνία μεταξύ των διαφόρων τμημάτων σε ένα Πληροφοριακό Σύστημα;
- Ποιες συσκευές χρησιμοποιούνται στη δόμηση ενός δικτύου;
- Ποιες είναι οι σύγχρονες υπηρεσίες των δικτύων υπολογιστών;

3.3.1 Τι είναι ένα Δίκτυο Υπολογιστών

Η ραγδαία εξέλιξη καθώς και η σύγκλιση των υπολογιστικών με τα τηλεπικοινωνιακά συστήματα έχει επιφέρει σημαντικές αλλαγές στον τρόπο με τον οποίο επικοινωνούν και συναλλάσσονται οι άνθρωποι. Η απανταχού πρόσβαση σε άμεσα διαθέσιμους υπολογιστικούς πόρους και ασύρματες συσκευές, καθώς και η ανάπτυξη συνδέσεων μεγάλης ταχύτητας διευκολύνει τη συνεργασία και τις κοινωνικές αλληλεπιδράσεις. Τα δίκτυα υπολογιστών, με κύριο εκπρόσωπο το Διαδίκτυο (Internet) παρέχουν την απαραίτητη υποδομή για την επίτευξη της Κοινωνίας της Πληροφορίας.

Δίκτυο Υπολογιστών ή Δίκτυο Επικοινωνιών είναι ένα σύνολο συνδεδεμένων μεταξύ τους συσκευών με φυσικές συνδέσεις (κανάλια επικοινωνίας), οι οποίες μπορούν να παράγουν, να στέλνουν, να προωθούν και να λαμβάνουν πληροφορίες (απλά δεδομένα, ήχο, εικόνα και βίντεο).

Οι υπολογιστές και οι συσκευές (εκτυπωτές, τερματικά, δρομολογητές και δορυφόροι) που συνδέονται σε ένα δίκτυο ονομάζονται ***σταθμοί*** ή ***τερματικές συσκευές***. Χαρακτηριστικές εφαρμογές δικτύων υπολογιστών αποτελούν οι τηλεπικοινωνίες (σταθερή/κινητή τηλεφωνία), η καλωδιακή τηλεόραση, το ηλεκτρονικό ταχυδρομείο, η τηλεγραφία, η τηλεδιάσκεψη), οι οικονομικές υπηρεσίες (π.χ. ηλεκτρονική συναλλαγή με την εφορία), οι πωλήσεις, το μάρκετινγκ και η διαφήμιση.



Εικόνα 3.14. Στοιχεία Μεταγωγής. Σε αυτή την κατηγορία ανήκουν οι Γέφυρες (Bridges), οι Μεταγωγείς (Switches), οι Δρομολογητές (Routers) και οι Πύλες (Gateways).



Το λογισμικό το οποίο πραγματοποιεί τις διάφορες λειτουργίες ενός δικτύου (όπως δρομολόγηση, έλεγχο σφαλμάτων, τμηματοποίηση πληροφορίας, κρυπτογράφηση πληροφορίας) είναι οργανωμένο με βάση κανόνες, οι οποίοι ονομάζονται ***πρωτόκολλα***.

3.3.2 Στοιχεία δικτύων

Οι διάφορες συσκευές ενός δικτύου συνδέονται μεταξύ τους με γραμμές/ζεύξεις μετάδοσης (links).

Σε ένα δίκτυο υπάρχουν ***υπολογιστές υποδοχής***, ***γραμμές μετάδοσης*** και ***στοιχεία μεταγωγής***.

Οι **Υπολογιστές Υποδοχής** (hosts) παίζουν το ρόλο του πομπού ή του δέκτη και μπορεί να είναι κινητές συσκευές, προσωπικοί υπολογιστές, ή κεντρικά ισχυρά υπολογιστικά συστήματα (Mainframes).

Οι **Γραμμές Μετάδοσης** (Transmission Lines, Links) αποτελούν τα φυσικά μονοπάτια επικοινωνίας (κανάλια ή δίαυλους) διαμέσου των οποίων μεταφέρονται τα δεδομένα από τη μία συσκευή στην άλλη. Η επικοινωνία γίνεται με ασύρματο και ενσύρματο τρόπο.

Τα **Στοιχεία Μεταγωγής** (Switching Elements) είναι οι ενδιάμεσες συσκευές που συνδέουν τις γραμμές μετάδοσης και επιφορτίζονται με το έργο μεταφοράς των δεδομένων από την μία πλευρά στην άλλη μέσω διαδικασιών γνωστών ως δρομολόγηση και/ή μεταγωγή (Εικόνα 3.14).

3.3.3 Κατηγοριοποίηση δικτύων

Ένα δίκτυο μπορεί να ανήκει σε διάφορες κατηγορίες ανάλογα με τη γεωγραφική του έκταση, το μέγεθός του και την τεχνολογία μετάδοσης και προώθησης που χρησιμοποιεί.

3.3.3.1 Είδη δικτύων ανάλογα με την τεχνολογία μετάδοσης

Στα **Δίκτυα σημείου προς σημείο** (point to point) δημιουργούνται συνδέσεις μεταξύ δύο κόμβων, οπότε ένα μήνυμα διατρέχοντας πολλές τέτοιες συνδέσεις, φθάνει στον προορισμό του. Στα ***δίκτυα εκπομπής***, όλα τα μέλη του δικτύου μοιράζονται έναν κοινό δίαυλο. Στην ***πολυεκπομπή*** (multicasting) το μήνυμα λαμβάνεται από συγκεκριμένους παραλήπτες ενώ στην ***καθολική εκπομπή*** (broadcasting) από όλους.

3.3.3.2 Είδη δικτύων ανάλογα με την τεχνολογία προώθησης της πληροφορίας

Στα **δίκτυα μεταγωγής κυκλώματος** (circuit switching networks) η μετάδοση επιτυγχάνεται μέσα από μια αποκλειστικά χρησιμοποιούμενη ***φυσική σύνδεση (κύκλωμα)***, στην οποία τα δεδομένα δεν υποβάλλονται σε καμία επεξεργασία κατά τη διέλευσή τους από το δίκτυο. Η τεχνολογία μεταγωγής κυκλώματος χρησιμοποιείται κυρίως στο τηλεφωνικό δίκτυο.

Στα **δίκτυα μεταγωγής πακέτου** (packet switching networks) τα δεδομένα χωρίζονται σε μικρότερα τμήματα, τα ***πακέτα***. Κάθε πακέτο πέραν

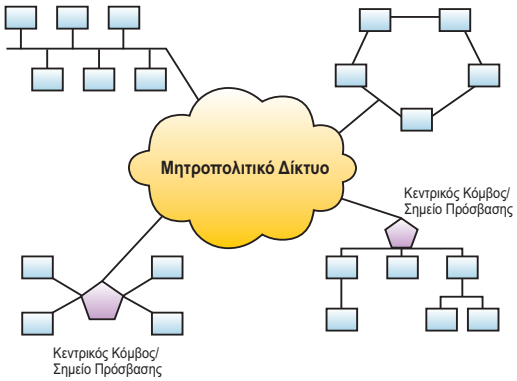
του τμήματος προς αποστολή, περιέχει και πληροφορίες ελέγχου οι οποίες διασφαλίζουν τη σωστή δρομολόγηση του πακέτου μέσα στο δίκτυο. Στη συνέχεια αυτά μέσω ενδιάμεσων ***κόμβων*** ή στοιχείων μεταγωγής, φθάνουν στον τελικό παραλήπτη που τα συναρμολογεί και δημιουργεί το αρχικό μήνυμα. Τα πακέτα μπορούν να ακολουθούν την ίδια διαδρομή ή κάθε πακέτο μπορεί να ακολουθεί τη δική του διαδρομή.

3.3.3.3 Είδη δικτύων βάσει περιοχής που καλύπτουν

Τα **Τοπικά Δίκτυα** (LAN – Local Area Networks) καλύπτουν μία μικρή έκταση, συνδέοντας συσκευές που βρίσκονται σε ένα δωμάτιο ή σε ένα κτίριο ή συγκρότημα κτιρίων.

Τα **Μητροπολιτικά Δίκτυα** (MAN – Metropolitan Area Networks) εκτείνονται στο περιβάλλον μιας ολόκληρης πόλης και χρησιμοποιούνται για την διασύνδεση δικτύων LAN ή σαν δίκτυα κορμού. Παραδείγματα τέτοιων δικτύων είναι ένα τηλεπικοινωνιακό δίκτυο υποδομής και ένα δίκτυο καλωδιακής τηλεόρασης (Εικόνα 3.15).

Τα δίκτυα **Ευρείας Περιοχής** (WAN – Wide Area Networks) επεκτείνονται σε μεγάλες γεωγραφικές περιοχές που αποτελούνται από διάφορες χώρες ή ακόμα και ηπείρους. Χρησιμοποιούν πλήθος ενδιάμεσων συσκευών, όπως π.χ. δορυφόρους.



Εικόνα 3.15. Παράδειγμα ενός Μητροπολιτικού δικτύου που αποτελείται από 4 δίκτυα.

3.3.4 Τοπολογίες δικτύων

Ο τρόπος με τον οποίο συνδέονται οι σταθμοί σε ένα δίκτυο ονομάζεται ***τοπολογία δικτύου***.

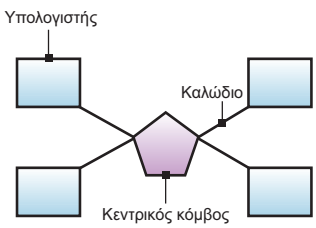
Υπάρχουν τρεις βασικές τοπολογίες δικτύων, οι οποίες διαφέρουν ως προς την αξιοπιστία, το κόστος, την ανοχή τους σε σφάλματα και την ταχύτητα ανάνηψης μετά από κατάρρευση.

Τοπολογία Αστέρα: Υπάρχει ένας κεντρικός κόμβος για τον έλεγχο της κυκλοφορίας και όλες οι συσκευές συνδέονται με αυτόν με μία φυσική σύνδεση (Εικόνα 3.16).

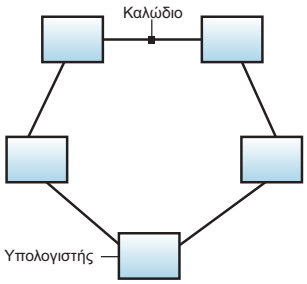


Διαδίκτυο (Internet ή internetwork) είναι ένα σύνολο από δύο ή περισσότερα δίκτυα (LAN, MAN, WAN κλπ.) που συνδέονται μεταξύ τους με κατάλληλες συσκευές. Σήμερα με τον όρο Διαδίκτυο αναφερόμαστε στην τεχνολογία που στηρίζεται στα πρωτόκολλα TCP και IP και αποτελεί τη βάση για τις εφαρμογές στον Παγκόσμιο Ιστό (World Wide Web, WWW).

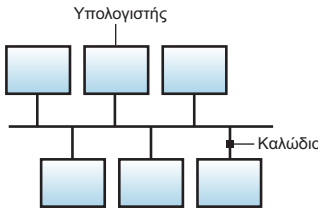
Το ***TCP/IP*** (Transmission Control Protocol/Internet Protocol=Πρωτόκολλο Ελέγχου Μετάδοσης / Πρωτόκολλο Διαδικτύου) είναι μια συλλογή πρωτοκόλλων επικοινωνίας στα οποία βασίζεται το Διαδίκτυο.



Εικόνα 3.16. Τοπολογία Αστέρα.



Εικόνα 3.17. Τοπολογία Δακτυλίου.



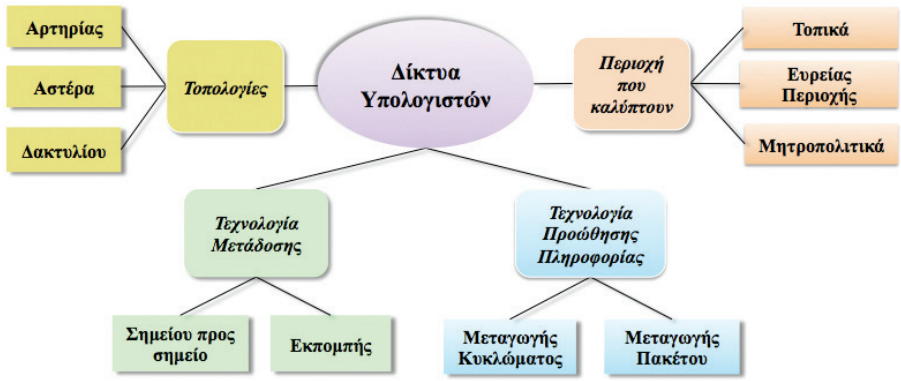
Εικόνα 3.18. Τοπολογία Αρτηρίας.

Τοπολογία Δακτυλίου: Σε αυτην την τοπολογία, η κάθε συσκευή συνδέεται με μια γραμμή (point-to-point link) με τις δύο διπλανές συσκευές, δημιουργώντας ένα δακτύλιο. Ένα μήνυμα μεταφέρεται από τον κάθε κόμβο στον διπλανό του προς την ίδια κατεύθυνση μέχρι να φθάσει στον προορισμό του (εικόνα 3.17). Η τοπολογία αυτή χρησιμοποιείται τόσο σε τοπικά όσο και ευρείας περιοχής δίκτυα.

Τοπολογία Αρτηρίας: Στην τοπολογία αυτή, υπάρχει μια γραμμή (καλώδιο) που αποτελεί τη ραχοκοκαλιά του δικτύου και όλες οι συσκευές είναι συνδεμένες σε αυτήν (Εικόνα 3.20). Η τοπολογία αυτή συναντάται κυρίως σε τοπικά δίκτυα.

Εκτός από τις τρεις αυτές βασικές τοπολογίες υπάρχουν και άλλες οι οποίες αποτελούν παραλλαγές/συνδυασμούς των τριών βασικών, όπως οι τοπολογίες πλέγματος και τοπολογίες δένδρου.

Όλες οι κατηγοριοποιήσεις που έχουν αναφερθεί μέχρι τώρα συνοψίζονται στον παρακάτω νοητικό χάρτη (Εικόνα 3.19):



Εικόνα 3.19. Νοητικός χάρτης κατηγοριοποίησης των Δικτύων Υπολογιστών.



Εικόνα 3.20. Sir Tim Berners Lee. Ο δημιουργός του Παγκόσμιου Ιστού.

3.3.5 Σύγχρονες υπηρεσίες δικτύων

Οι σύγχρονες υπηρεσίες των δικτύων υπολογιστών στηρίζονται κυρίως στην αλματώδη ανάπτυξη του Διαδικτύου, το οποίο από ένα ρόλο παρουσίασης πληροφοριών μεταλλάχθηκε σε ένα εργαλείο κοινωνικής δικτύωσης, αναμορφώνοντας τον τρόπο επικοινωνίας και συνεργασίας μεταξύ ανθρώπων.

Ο **Παγκόσμιος Ιστός** (World Wide Web ή απλά Web), δίνει στο χρήστη τη δυνατότητα της πρόσβασης σε μεγάλη ποικιλία από πληροφορίες που είναι αποθηκευμένες στο διαδίκτυο υπό μορφή ιστοσελίδων.

Η τηλεφωνία μέσω διαδικτύου (**Voice over IP** ή **VoIP**) προσφέρει φωνητική συνομιλία σε πραγματικό χρόνο με καλή ποιότητα και με μηδενικό κόστος.

Οι **τεχνολογίες DSL** (Digital Subscriber Line) παρέχουν πρόσβαση υψηλών ταχυτήτων στο διαδίκτυο και με χρήση των υπαρχουσών χάλκινων τηλεφωνικών γραμμών.

Το **Υπολογιστικό Νέφος** ή **σύννεφο** (Cloud Computing) παρέχει υπολογιστικούς πόρους (όπως διάφορες εφαρμογές, βάσεις δεδομένων, υπηρεσίες αρχείων, ηλεκτρονικό ταχυδρομείο κ.α.) μέσω ενός δικτύου υπολογιστών. Πρόκειται για μία παγκόσμια τεχνολογική υποδομή (global technological infrastructure), στην οποία ο χρήστης ενός υπολογιστή έχει πρόσβαση και χρησιμοποιεί λογισμικό και δεδομένα τα οποία είναι εγκατεστημένα ή βρίσκονται εκτός του προσωπικού του υπολογιστικού συστήματος.

Ανακεφαλαίωση

Δίκτυα υπολογιστών δημιουργούνται με τη διασύνδεση υπολογιστών και περιφερειακών συσκευών με διάφορους τρόπους. Οι υπηρεσίες των δικτύων έχουν εφαρμογή στους περισσότερους τομείς της καθημερινότητάς μας. Αναδύονται καινούριες εφαρμογές σε πολύ συχνή βάση.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Τι είναι ένα δίκτυο υπολογιστών;
2. Αναζητήστε στο διαδίκτυο πληροφορίες για τη δομή του ***δικτύου της κινητής τηλεφωνίας***. Αν στην τάξη σας βρίσκονται μαθητές με διαφορετική καταγωγή, συγκρίνετε τα σχετικά δίκτυα μεταξύ τους από πλευράς δομής τους (ομαδική εργασία).
3. Αναζητήστε πληροφορίες στο διαδίκτυο για τα κύρια χαρακτηριστικά δύο ενσύρματων και ***2 ασυρμάτων καναλιών επικοινωνίας***.
4. Αναζητήστε στο διαδίκτυο πληροφορίες για υπηρεσίες ***υπολογιστικού νέφους***.
5. Αναζητήστε πληροφορίες για την συμβολή των δικτύων υπολογιστών στην υποστήριξη ΑμεΑ.
6. Ποιές από τις παρακάτω προτάσεις είναι σωστές;
   - **Α.** Οι Υπολογιστές Υποδοχής μπορεί να είναι κινητές συσκευές, προσωπικοί υπολογιστές.
   - **Β.** Στα Δίκτυα Μεταγωγής Κυκλώματος τα δεδομένα υποβάλλονται σε επεξεργασία κατά την διέλευσή τους απο το δίκτυο.
   - **Γ.** Τα Μητροπολιτικά Δίκτυα είναι μεγαλύτερα από τα Δίκτυα Ευρείας Περιοχής.
   - **Δ.** Η Τοπολογία Δακτυλίου χρησιμοποιείται τόσο σε τοπικά όσο και ευρείας περιοχής δίκτυα.



Πολλά Πληροφοριακά Συστήματα χρησιμοποιούν Αρχιτεκτονικές αποθήκευσης βασισμένες στο δίκτυο και στο σύννεφο.



Χρήσιμοι Υπερσύνδεσμοι

http://www.w3.org
Ο οργανισμός World Wide Web Consortium.

http://www.eett.gr/opencms/opencms/EETT/index.html
Εθνική Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων.

http://www.yme.gr
Υπουργείο Υποδομών Μεταφορών και Δικτύων



Λέξεις κλειδιά
Δίκτυο, μεταγωγή, μετάδοση, τοπολογίες, διαδίκτυο, LAN, WAN, υπολογιστικό νέφος.

Τεχνητή Νοημοσύνη

Στόχοι του κεφαλαίου είναι οι μαθητές:

- ✓ Να εντάξουν την Τεχνητή Νοημοσύνη στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών
- ✓ Να γνωρίσουν τις επιστημονικές περιοχές εφαρμογής της Τεχνητής Νοημοσύνης
- ✓ Να μπορούν να αναφέρουν τομείς στους οποίους έχει εφαρμογή η Τεχνητή Νοημοσύνη.

3.4.1 Τι είναι η Τεχνητή Νοημοσύνη

Τα τελευταία χρόνια η Τεχνητή Νοημοσύνη έχει ενταχθεί στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών λόγω των θεμάτων που πραγματεύεται και των δεξιοτήτων που αναπτύσσονται στους χώρους εφαρμογής της.

Τεχνητή Νοημοσύνη (Τ.Ν.) είναι ο τομέας της επιστήμης των υπολογιστών, που ασχολείται με τη σχεδίαση ευφυών υπολογιστικών συστημάτων, δηλαδή συστημάτων ικανών για λειτουργίες που αποδίδονται σε ανθρώπινη νοημοσύνη.

Η Τ.Ν. εμφανίζεται σε διάφορα πεδία, όπως:

- στον προγραμματισμό ηλεκτρονικών παιχνιδιών
- στα έμπειρα συστήματα τα οποία χρησιμοποιούνται για την λήψη αποφάσεων γύρω από ζητήματα πραγματικής ζωής, διάγνωση και θεραπεία ασθενειών
- στην επεξεργασία φυσικών γλωσσών που βοηθά στην κατανόηση των νοημάτων μίας γλώσσας με τη χρήση μηχανών
- στον προγραμματισμό νοήμονος συμπεριφοράς σε μηχανές με τη χρήση νευρωνικών δικτύων τα οποία προσομοιώνουν τις φυσικές συνδέσεις ενός ανθρώπινου εγκεφάλου
- στη ρομποτική όπου προγραμματίζονται υπολογιστικά συστήματα ώστε να «βλέπουν», να «ακούν» και να «αντιδρούν» με βάση τα ανθρώπινα πρότυπα συμπεριφοράς (Εικόνα 3.21).



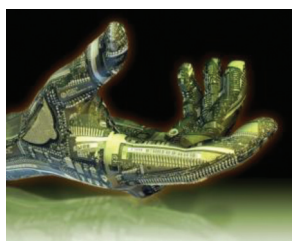
Προερωτήσεις

• Σε έναν αγώνα σκάκι μεταξύ ανθρώπου και υπολογιστή πώς σχεδιάζονται οι πιθανές κινήσεις της μηχανής;
• Πώς λειτουργεί ένα σύστημα διόρθωσης κειμενογράφου;
• Πώς γίνεται η αναζήτηση πληροφοριών στο διαδίκτυο;
• Με ποιον τρόπο θα δίνατε οδηγίες σε ένα αυτοκινούμενο όχημα ώστε να διανύσει μια διαδρομή με εμπόδια;



1. Το ρομπότ δεν θα κάνει κακό σε άνθρωπο, ούτε με την αδράνειά του θα επιτρέψει να βλαφτεί ανθρώπινο όν.
2. Το ρομπότ πρέπει να υπακούει τις διαταγές που του δίνουν οι άνθρωποι, εκτός αν αυτές οι διαταγές έρχονται σε αντίθεση με τον πρώτο νόμο.
3. Το ρομπότ οφείλει να προστατεύει την ύπαρξή του, εφόσον αυτό δεν συγκρούεται με τον πρώτο και τον δεύτερο νόμο.

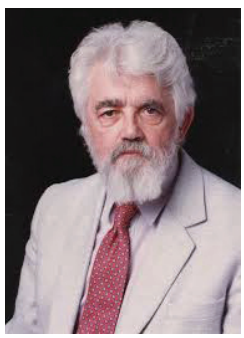
(Ισαάκ Ασίμωφ, Isaac Asimov, 1942)



Εικόνα 3.21. Χέρι ρομπότ.

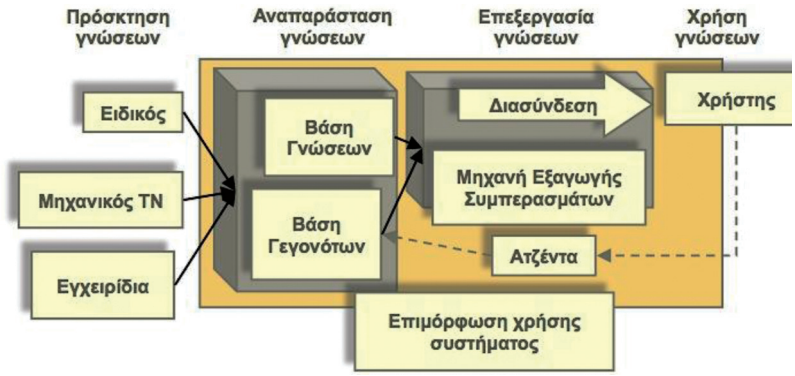
Κάθε μορφή γνώσης μπορεί να διακριθεί σε τρία στάδια:
- Πρόσκτηση γνώσεων
- Επεξεργασία γνώσεων
- Χρήση γνώσεων

Τεστ Turing
Η ιδέα της T.N. προήλθε από ένα «παιχνίδι μίμησης» που σήμερα είναι γνωστό ως Τεστ Turing. Ήταν ένα παιχνίδι ανάμεσα σε έναν ερωτώντα, μία γυναίκα, και έναν ψηφιακό υπολογιστή. Μεταξύ τους επικοινωνούσαν μέσω ηλεκτρονικού ταχυδρομείου (τηλε-εκτυπωτής Turing). Ο ερωτώντας μπορούσε να θέσει οποιαδήποτε ερώτηση είτε στην γυναίκα είτε στον υπολογιστή με τελικό στόχο να ανακαλύψει τη σωστή ταυτότητα του συνομιλητή του. Οι ερωτήσεις αποτελούσαν τα εισερχόμενα, οι απαντήσεις της γυναίκας και του υπολογιστή τα εξερχόμενα του συστήματος και το τεστ στόχευε να δείξει τη δυνατότητα ύπαρξης μηχανών με νοημοσύνη.



Εικόνα 3.23. Ο Τζων ΜακΚάρθυ, (John McCarthy) έλαβε το 1971 το Βραβείο Turing για τη συνεισφορά του στον τομέα της T.N.

Κάθε πρόβλημα στην T.N. διακρίνεται σε δύο τμήματα. Το πρώτο τμήμα αφορά στην αναπαράσταση των υποθέσεων και των συμπερασμάτων, ενώ το δεύτερο αφορά στον ορισμό και την παραγωγή των λογικών σχέσεων μεταξύ των υποθέσεων και των συμπερασμάτων. Η T.N. χρησιμοποιεί ***λογική*** για να αναπαραστήσει τη γνώση και το ***μηχανισμό των αποδείξεων και των συμπερασμάτων*** για να την επεξεργαστεί. Ένα τυπικό σύστημα T.N. αποτελείται από τρία μέρη: το στάδιο εισερχομένων, το στάδιο επεξεργασίας και το στάδιο εξερχομένων. Με παρόμοιο τρόπο στην T.N. κάθε μορφή γνώσης μπορεί να αναπαρασταθεί σε τρία αντίστοιχα στάδια (Εικόνα 3.22). Το πρώτο αφορά στον τρόπο με τον οποίο η γνώση γίνεται μεταδόσιμη (Πρόσκτηση γνώσεων) μέσω ειδικών, εγχειριδίων ή μηχανικών της T.N. οι οποίοι τροφοδοτούν μια βάση γνώσης.



Εικόνα 3.22. Αναπαράσταση Γνώσεων από την μεριά της Τεχνητής Νοημοσύνης.

Το δεύτερο στάδιο (Επεξεργασία γνώσεων) αφορά στην επεξεργασία αυτών που έχουν εισαχθεί και στο τρίτο στάδιο (Χρήση γνώσεων), καταγράφεται το πώς αξιοποιούνται τα τελικά συμπεράσματα. Τα τρία αυτά στάδια μπορεί να βρίσκονται σε κατάσταση αλληλεπίδρασης.

3.4.2 Εξέλιξη της Τεχνητής Νοημοσύνης

Το 1956 σε ένα ερευνητικό πρόγραμμα T.N. ο ***Τζων ΜακΚάρθυ*** (Εικόνα 3.25) ανέπτυξε μια αλγεβρική γλώσσα επεξεργασίας λιστών που έδωσαν ώθηση στην έρευνα των ***Γενετικών Αλγορίθμων*** (Genetic Algorithms). Οι γενετικοί αλγόριθμοι είναι αλγόριθμοι οι οποίοι αναζητούν μέσα από ένα χώρο υποψηφίων λύσεων, την πιο κατάλληλη με βάση κάποιο συγκεκριμένο κριτήριο. Σημαντικά προβλήματα που αντιμετωπίζονται με αυτόν τον τρόπο είναι η εύρεση μεγίστου συνάρτησης μιας μεταβλητής, το πρόβλημα εύρεσης συντομότερης διαδρομής, η σχεδίαση κυκλωμάτων VLSI και ο σχεδιασμός ενός ωρολογίου προγράμματος. Την δεκαετία του ’70 εμφανίζονται τα πρώτα ***έμπειρα συστήματα*** (Expert Systems) ως εμπορικές εφαρμογές και την δεκαετία του ’80 γίνεται προσπάθεια για ενοποίηση των διαλέκτων της γλώσσας συναρτησιακού προγραμματισμού (Functional Programming) ***LISP*** κάτω από το πρότυπο της ***Common LISP.*** Η ***LISP*** βασίζεται στην αποτί-

μηση συναρτήσεων αντί για την εκτέλεση εντολών. Προγράμματα βασισμένα στο συναρτησιακό προγραμματισμό είναι γραμμένα με βάση συναρτήσεις που αποτιμούν βασικές τιμές.

Παράλληλα υλοποιήθηκαν τα πρώτα ***Τεχνητά Νευρωνικά Δίκτυα*** (Artificial Neural Nets). Το αντικείμενο των τεχνητών νευρωνικών δικτύων αποτελεί ένα ευρύ και αυτόνομο επιστημονικό πεδίο, που σχετίζεται με το γενικότερο πλαίσιο των ευφυών συστημάτων και δικτύων. Μπορεί να σκεφτεί κανείς μια σειρά επιτυχημένων εφαρμογών όπως τα έξυπνα φίλτρα ηλεκτρονικού ταχυδρομείου, οι υψηλών ταχυτήτων και αποτελεσματικοί δρομολογητές, και η αυτόματη αναγνώριση πληροφοριών που έχουν σημασία και συνάφεια.

Ένα ακόμα βήμα στην εξέλιξη της Τ.Ν. ήταν η σύνδεσή της με την θεωρία της ***Ασαφούς Λογικής*** (Fuzzy Logic). Η θεωρία της Ασαφούς Λογικής μέσω κατάλληλων αλγορίθμων έχει προσαρμοσθεί σε συστήματα ώστε, μέσω της υπάρχουσας γνώσης μιας ασαφούς κατάστασης να προκύπτουν διαχειρίσιμα και συγκεκριμένα συμπεράσματα. Βασικός εισηγητής της είναι ο καθηγητής Λότφι Ζάντεχ (Εικόνα 3.26). Σχετικές εφαρμογές υπάρχουν σε συστήματα ταξινόμησης και σε αυτόματους διορθωτές κειμένων.

Τη δεκαετία του ’90 με την ανάπτυξη και τη διάδοση του διαδικτύου εμφανίζονται οι πρώτοι ***ευφυείς πράκτορες*** (intelligent agents), ανεξάρτητα προγράμματα τα οποία λαμβάνουν αποφάσεις και αλληλεπιδρούν με τα διάφορα συστήματα, και τα διαδικτυακά ρομπότ. Ακολούθησαν ο ***Εξελικτικός Υπολογισμός*** (Evolutionary Computation), η ***Νοημοσύνη Σμηνών*** (Swarm Intelligence), η αξιοποίηση της ***Μηχανικής Μάθησης*** (Machine Learning) και η ***Ανακάλυψη Γνώσης σε Βάσεις Δεδομένων*** (Knowledge Discovery in Databases) ως κατηγορίες της ***Υπολογιστικής Νοημοσύνης***.

3.4.3 Τομείς εφαρμογών της Τεχνητής Νοημοσύνης

Η Τ.Ν. επωφελήθηκε από την πρόοδο της Ηλεκτρονικής, της μικροηλεκτρονικής και της νανοτεχνολογίας (Εικόνα 3.25). Η ανάπτυξη και η εξέλιξη της Τ.Ν. στηρίζεται στην αξιοποίηση γνώσεων από περιοχές που εκτείνονται από τη μηχανική έως τη θεωρητική πληροφορική.

Η Τ.Ν. παρέχει την δυνατότητα επεξεργασίας σε χώρους που δεν μπορεί να έχει πρόσβαση ο άνθρωπος, όπως το διάστημα (Εικόνα 3.26), τα βάθη των ωκεανών, οι μολυσμένες περιοχές.

Η Τ.Ν. εφαρμόζεται σε ένα πλήθος πεδίων, όπως η συμπερασματολογία, ο αυτοματισμός, η ρομποτική, η αναγνώριση φωνής, η ευφυής αναζήτηση στο διαδίκτυο και τα συστήματα ελέγχου. Ο ρόλος της Τ.Ν. είναι σημαντικός στην επίλυση σύνθετων προβλημάτων, στο σχεδιασμό της συμπεριφοράς μηχανών, τη χρήση βιομηχανικών ρομπότ και ειδικότερα όπου χρειάζεται να γίνει ανάλυση μέσων και σκοπών. Σημαντικά πεδία εφαρμογής επίσης είναι η ***οπτική - ακουστική αντίληψη*** και η ***αναγνώριση προτύπων*** (pattern recognition).



Εικόνα 3.24. Λότφι Ζάντεχ (Lotfi Zadeh). Βασικός εισηγητής της Ασαφούς Λογικής το 1965.

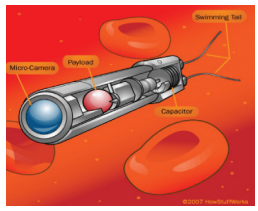


Η **Μηχανική Μάθηση** πραγματοποιείται όταν ένα υπολογιστικό σύστημα δημιουργεί νέα μοντέλα ή πρότυπα μέσα από ένα σύνολο δεδομένων.



Ο **Εξελικτικός Υπολογισμός** και η **Νοημοσύνη Σμηνών** είναι μέθοδοι διαδικασιών βελτιστοποίησης υπολογισμών που προέκυψαν παρατηρώντας πρότυπα που υπάρχουν στη φύση. Σε αυτά γίνεται χρήση μαθηματικών εργαλείων από την Θεωρία Πιθανοτήτων και τα Δυναμικά Συστήματα.

Η **Ανακάλυψη Γνώσης σε Βάσεις Δεδομένων** είναι μία σύνθετη διαδικασία η οποία προσδιορίζει έγκυρες, νέες, χρήσιμες και κατανοητές σχέσεις - πρότυπα που υπάρχουν σε μία βάση δεδομένων.



Εικόνα 3.25. Ιατρικά μικρορομπότ



Εικόνα 3.26. Ο «ρομποναύτης» της NASA κατασκευάστηκε για την συντήρηση του διαστημικού τηλεσκόπιου HUBLLE.

```
man(giannis).
man(giorgos).
woman(aggeliki).
```

Εικόνα 3.27. Πρόγραμμα σε Prolog.

Αφού γραφεί το πρόγραμμα και εκτελεστεί, θα εμφανιστεί:
?-
όπου περιμένει ερωτήσεις. Π.χ.
?- man(giorgos).
και απαντά
Yes
?- woman(maria).
και απαντά
No



Χρήσιμοι Υπερσύνδεσμοι

http://www.eccai.org
Ευρωπαϊκή Συντονιστική Επιτροπή για Τ.Ν.

http://www.eetn.gr/
Ελληνική Εταιρεία Τεχνητής Νοημοσύνης



Λέξεις κλειδιά
Υπολογιστικά μοντέλα ανθρώπινης γνώσης, Νευρωνικά Δίκτυα, Έμπειρα Συστήματα, Ασαφής Λογική, Γενετικοί Αλγόριθμοι.

Για παράδειγμα, στην περίπτωση ενός πυραύλου «αναγνώριση προτύπων» σημαίνει «εξατομίκευση στόχου» και «όραση υπολογιστή», σημαίνει δηλαδή την κατασκευή ενός συστήματος που «βλέπει» και «αντιδρά» εξίσου καλά με τον άνθρωπο. Άλλο πεδίο επιστημονικής εφαρμογής αποτελεί η έμπειρη ανάλυση και η παροχή συμβουλών στην ***ιατρική διάγνωση***. Στην ιατρική έχουν αναπτυχθεί συστήματα που αναλύουν τα συμπτώματα μιας ασθένειας, το ιατρικό ιστορικό, τα αποτελέσματα των εργαστηριακών εξετάσεων ενός ασθενή και στη συνέχεια προτείνουν στο θεράποντα ιατρό μία ποικιλία δυνατών διαγνώσεων.

3.4.4 Γλώσσες προγραμματισμού που χρησιμοποιούνται στην Τ.Ν.

Η ***Prolog*** είναι μια γλώσσα λογικού προγραμματισμού, η οποία κυριαρχεί στην περιοχή των εφαρμογών της Τ.Ν.. Ένα πρόγραμμα της ***Prolog*** είναι μία συλλογή από γεγονότα και κανόνες που χρησιμοποιούνται για να αποδειχθούν κάποιες προτάσεις. Αρχικά ο χρήστης εισάγει τα δεδομένα ενός προβλήματος στο περιβάλλον της ***Prolog***. Κατόπιν θέτει ερωτήματα γύρω από αυτά ή τις σχέσεις τους και λαμβάνει απαντήσεις. Οι απαντήσεις που δίνονται σε κάθε ερώτημα αξιοποιούν αποκλειστικά τα γεγονότα και τους κανόνες που έχουν εισαχθεί στο εκάστοτε πρόγραμμα (εικόνα 3.27). Μέσα από συσχετισμούς και συλλογισμούς, το περιβάλλον της ***Prolog*** καταλήγει να αποδείξει το αληθές μιας πρότασης ή την προβολή του αποτελέσματος που θα προκύψει.

Ανακεφαλαίωση

Η Τ.Ν. είναι ο τομέας της επιστήμης των υπολογιστών, που ασχολείται, εκτός των άλλων, με τη σχεδίαση ευφυών (νοημόνων) υπολογιστικών συστημάτων, δηλαδή συστημάτων που επιδεικνύουν χαρακτηριστικά που σχετίζονται με τη νοημοσύνη στην ανθρώπινη συμπεριφορά. Τα προβλήματα που επιλύονται αφορούν την αντίληψη μέσω της όρασης, την μηχανική μάθηση, την εξαγωγή συμπερασμάτων από βάσεις δεδομένων κλπ. Η βασική γλώσσα προγραμματισμού της είναι η Prolog.

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Σχολιάστε και προτείνετε ταινίες επιστημονικής φαντασίας με κύρια αναφορά στην Τ.Ν. Εξηγήστε γιατί σας έκαναν εντύπωση.
2. Να διερευνήσετε τι σημαίνει για τον άνθρωπο, τους ηθικούς του κώδικες και την εξέλιξή του η αποδοχή βοήθειας ή συμβουλών από μία μηχανή ή ένα υπολογιστικό σύστημα;
3. Αναζητήστε αναφορές στην Τ.Ν. από την αρχαία ελληνική σκέψη, τη μυθολογία, την ιστορία ή τη λογοτεχνία των νεότερων χρόνων.

Βιβλιογραφία

Comer, D. E. (2014). *Δίκτυα και Διαδίκτυα Υπολογιστών*. Αθήνα: Κλειδάριθμος.

Floridi, L. (2008). *Εισαγωγή στη φιλοσοφία της Πληροφορικής*. Αθήνα: Νήσος.

Forouzan, B. A. (2006). *Πρωτόκολλο TCP/IP*. Αθήνα: Γκιούρδας.

Laudon, K. C., & Laudon, J. P. (2009). *Πληροφοριακά Συστήματα Διοίκησης*. Αθήνα: Κλειδάριθμος.

Russell, S., & Norvig, P. (2004). *Τεχνητή νοημοσύνη: Μια σύγχρονη προσέγγιση*. Αθήνα: Κλειδάριθμος.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2013). *Λειτουργικά Συστήματα*. Αθήνα: Γκιούρδας.

Tanenbaum, A. S. (2009). *Σύγχρονα Λειτουργικά Συστήματα*. Αθήνα: Κλειδάριθμος.

Tanenbaum, A. S. (2012). *Δίκτυα Υπολογιστών*. Αθήνα: Κλειδάριθμος.

Webber, A. B. (2009). *Σύγχρονες γλώσσες προγραμματισμού*. Πανεπιστημιακές Εκδόσεις Κρήτης.

Αλεξανδρής, Ν., Μπελεσιώτης, Β. Σ., & Παναγιωτόπουλος, Θ. (2002). *Εισαγωγή στην Επιστήμη των Υπολογιστών*. Αθήνα: Εκδόσεις Βαρβαρήγου.

Βακάλη, Α., Γιαννόπουλος, Η., Ιωαννίδης, Ν., Κοίλιας, Χ., Μάλαμας, Κ. Μανωλόπουλος, Ι., & Πολίτης, Π. (1999). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*. Αθήνα: ΥΠΕΠΘ.

Βλαχάβας, Ι., Κεφαλάς, Π., Βασιλειάδης, Ν., Κόκκορας, Φ., & Σακελλαρίου, Η. (2011). *Τεχνητή Νοημοσύνη*. Εκδόσεις Πανεπιστημίου Μακεδονίας.

Βογιατζής, Ι., Ιωαννίδης, Ν., Κοίλιας, Χ., Μελετίου, Γ., & Μόρμορης, Μ. (2010). *Εισαγωγή στην Αλγοριθμική*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Γεωργόπουλος, Ν., Κοπανάκη, Ε., Πανταζή, Μ., Νικολαράκος, Χ., & Βαγγελάτος, Ι. (2013). *Ηλεκτρονικό Επιχειρείν*. Αθήνα: Εκδόσεις Σταμούλης.

Δημητριάδης, Α. (2007). *Διοίκηση-Διαχείριση Πληροφοριακών Συστημάτων*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Δουληγέρης, Χ. (2014). *Σύγχρονα Τηλεπικοινωνιακά και Διαδικτυακά Πρωτόκολλα*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Καλαφατούδης, Σ., Δροσίτης, Ι., & Κοίλιας, Χ. (2012). *Εισαγωγή στις Τεχνολογίες Πληροφορίας και Επικοινωνίας*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Κοίλιας, Χ. (2004). *Δομές Δεδομένων και Οργανώσεις Αρχείων*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Κοίλιας, Χ., & Παναγιωτάκος, Δ. (1994). *Ερμηνευτικό Λεξικό Όρων Πληροφορικής*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Μανωλόπουλος, Ι., & Παπαδόπουλος, Α. (2006). *Συστήματα Βάσεων Δεδομένων*. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Γλωσσάρι

AND operation = πράξη ΚΑΙ, τομή, σύζευξη, λογικός πολλαπλασιασμός. Η πράξη άλγεβρας Boole της οποίας το αποτέλεσμα έχει την τιμή 1, αν και μόνο εάν κάθε τελεστέος έχει την τιμή 1 (ΕΛΟΤ).

Ada. Γενικού σκοπού υψηλού επιπέδου διαδικασιακή γλώσσα προγραμματισμού, που αναπτύχθηκε αρχικά υπό την αιγίδα του αμερικανικού Υπουργείου Άμυνας για την υλοποίηση πολύ μεγάλων έργων λογισμικού.

access = προσπέλαση. Πρόσβαση σε δεδομένα με σκοπό την ανάγνωση, εγγραφή ή μετακίνηση δεδομένων ή εντολών.

access method = μέθοδος προσπέλασης. 1) Στα τοπικά δίκτυα πρωτόκολλο που καθορίζει ποια συσκευή θα έχει προσπέλαση στο μέσο μεταφοράς κάθε στιγμή. 2) Η τεχνική ή και τα προγράμματα για τη μετακίνηση δεδομένων μεταξύ συσκευών εισόδου/εξόδου και κύριας μνήμης.

access right = δικαίωμα προσπέλασης. Δικαίωμα που παραχωρείται σε ένα χρήστη να έχει πρόσβαση σε ορισμένα δεδομένα ή προγράμματα και να τα χρησιμοποιεί κατά ένα συγκεκριμένο τρόπο (ISO).

access time = χρόνος προσπέλασης. Ο χρόνος που χρειάζεται για την ανεύρεση και ανάκτηση καταχωρημένης πληροφορίας σε κάποιο είδος μνήμης. Συνήθως αναφέρεται στο μέγιστο απαιτούμενο χρόνο.

algorithm = αλγόριθμος. Πεπερασμένο σύνολο σαφώς καθορισμένων κανόνων που βοηθούν στην επίλυση ενός προβλήματος μέσω ενός πεπερασμένου αριθμού βημάτων (ΕΛΟΤ).

algorithmic languages = αλγοριθμικές γλώσσες. Κατηγορία γλωσσών προγραμματισμού κατάλληλων για την περιγραφή προβλημάτων που μπορούν να λυθούν αλγοριθμικά.

alphabetic data = αλφαβητικά δεδομένα. Δεδομένα που αποτελούνται από γράμματα μόνο ή από γράμματα και διάκενα.

alphanumeric = αλφαριθμητικό. Σύνολο χαρακτήρων που μπορεί να εμπεριέχει γράμματα, ψηφία και ειδικά σύμβολα όπως π.χ. σημεία στίξης.

analysis = ανάλυση. 1) Η φάση της ανάπτυξης ενός συστήματος πληροφορικής κατά την οποία αναλύεται η λειτουργία ενός οργανισμού και γίνεται ο καθορισμός των απαιτήσεων και προδιαγραφών των προγραμμάτων. 2) Η μεθοδική μελέτη ενός προβλήματος και η διαδικασία της διάσπασής του σε μικρότερες μονάδες για περαιτέρω έρευνα σε λεπτομέρεια.

analyst = αναλυτής. Πρόσωπο με ειδικότητα τον καθορισμό των απαιτήσεων και τη σύνταξη των προδιαγραφών των προγραμμάτων για μια συγκεκριμένη εφαρμογή.

append = προσάρτηση. Προσθήκη στοιχείων στο τέλος αρχείου.

application = εφαρμογή. Προγράμματα που γράφονται για την κάλυψη συγκεκριμένης ανάγκης π.χ. μιας επιχείρησης σε επεξεργασία δεδομένων.

applications software = λογισμικό εφαρμογών. Λογισμικό που γράφεται για να καλύψει ανάγκες εφαρμογών.

arithmetic = αριθμητικός. Τύπος δεδομένων που υποστηρίζουν όλες οι σύγχρονες γλώσσες προγραμματισμού. Με στοιχεία του αριθμητικού τύπου δεδομένων γίνονται αριθμητικές πράξεις.

arithmetic expression = αριθμητική έκφραση. Μία έκφραση που περιλαμβάνει αριθμητικές πράξεις και τελεστέους και η οποία μπορεί να μετατραπεί σε μια απλή αριθμητική τιμή.

arithmetic operator = αριθμητικός τελεστής. Σύμβολο μιας γλώσσας προγραμματισμού, που καθορίζει την αριθμητική πράξη που πρέπει να γίνει μεταξύ δύο τελεστέων, π.χ. +, -, *, /, ^.

array = πίνακας. Διάταξη δεδομένων μιας ή περισσοτέρων διαστάσεων.

array element = στοιχείο πίνακα. Ένα απλό προσπελάσιμο στοιχείο δεδομένου σε έναν πίνακα.

array index = δείκτης πίνακα. Σύνολο μιας ή περισσοτέρων τιμών που χρησιμοποιούνται για να προσπελαστεί ένα στοιχείο πίνακα.

artificial intelligence (AI) = τεχνητή νοημοσύνη. Ο τομέας της επιστήμης των υπολογιστών, που ασχολείται με τη σχεδίαση ευφυών (νοημόνων) υπολογιστικών συστημάτων, ώστε να καταστεί ο υπολογιστής ικανός για λειτουργίες που αποδίδονται σε ανθρώπινη νοημοσύνη.

artificial neural nets = τεχνητά νευρωνικά δίκτυα που έχουν δυνατότητα μάθησης μετασχηματίζοντας την εσωτερική τους δομή.

artificial intelligence languages = γλώσσες τεχνητής νοημοσύνης. Γλώσσες με δομή κατάλληλη για ανάπτυξη προγραμμάτων τεχνητής νοημοσύνης.

assembler = συμβολομεταφραστής. Πρόγραμμα που μεταφράζει συμβολική γλώσσα σε γλώσσα μηχανής του δεδομένου υπολογιστή.

assembly language = συμβολική γλώσσα. Γλώσσα χαμηλού επιπέδου εξαρτώμενη από το υλικό και η οποία έχει άμεση αντιστοιχία με τη γλώσσα μηχανής. Αποτελεί συμβολική αναπαράσταση του δυαδικού κώδικα της γλώσσας μηχανής και χρειάζεται συμβολομετάφραση.

assignement = εκχώρηση. Μηχανισμός τιμοδότησης μιας μεταβλητής (ISO).

BASIC (Beginners All-purpose Symbolic Instruction Code). Δημοφιλής γλώσσα προγραμματισμού. Τυπικά ασχολείται με αυτήν όποιος βρίσκεται -τουλάχιστον- στα πρώτα του βήματα στον προγραμματισμό.

backup = εφεδρεία. Δημιουργία αντιγράφων ασφαλείας που μπορούν να χρησιμοποιηθούν σε περίπτωση απώλειας των αρχικών.

backup copy = αντίγραφο εφεδρείας. Ο όρος χρησιμοποιείται για το αντίγραφο ενός αρχείου ή προγράμματος που τηρείται στην περίπτωση καταστροφής του αρχικού.

binary search = δυαδική αναζήτηση. Μέθοδος αναζήτησης κατά την οποία τα δεδομένα διαιρούνται διαδοχικά σε δύο ίσα τμήματα. Ένα από τα δύο τμήματα απαλλάσσεται από περαιτέρω έρευνα, γιατί είναι γνωστό ότι δεν περιέχει το προς αναζήτηση δεδομένο.

boolean (data type) = λογικός τύπος δεδομένου. Τύπος δεδομένου με δύο τιμές: αληθής (true) - ψευδής (false).

bridge = γέφυρα. Μια ηλεκτρονική συσκευή ή λογισμικό που χρησιμοποιείται για τη σύνδεση ενός τύπου ή πρωτοκόλλου δικτύου με ένα άλλο.

bubble sort = ταξινόμηση φυσαλίδας. Απλός αλγόριθμος ταξινόμησης κατά τον οποίο διαδοχικά στοιχεία συγκρίνονται μεταξύ τους και εάν δεν είναι τοποθετημένα σωστά, αντιμετατίθενται. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου καμία άλλη αντιμετάθεση δεν μπορεί να γίνει.

business applications = επιχειρησιακές εφαρμογές. Κατηγορία εφαρμογών προσανατολισμένων στις ανάγκες επιχειρήσεων. Συχνά αποκαλούνται και εμπορικές εφαρμογές.

C Language = γλώσσα C. Γλώσσα υψηλού επιπέδου ιδιαίτερα κατάλληλη για τη δημιουργία λογισμικού συστήματος.

C++ Language = γλώσσα C++. Αντικειμενοστρεφής έκδοση της C.

COBOL (COmmon Business Oriented Language). Γλώσσα προγραμματισμού υψηλού επιπέδου, βασισμένη στην Αγγλική, που χρησιμοποιείται κυρίως για διαχειριστικές εφαρμογές.

call = κλήση. Η ενεργοποίηση ενός προγράμματος υπολογιστή, μιας ρουτίνας ή μιας υπορουτίνας, συνήθως με καθορισμό των συνθηκών εισόδου και εκτέλεση άλματος προς ένα σημείο εισόδου (ISO).

cloud computing = Υπολογιστικό Νέφος ή σύννεφο. Παρέχει υπολογιστικούς πόρους (όπως διάφορες εφαρμογές, βάσεις δεδομένων, υπηρεσίες αρχείων, ηλεκτρονικό ταχυδρομείο κ.α.,) μέσω ενός δικτύου υπολογιστών. Πρόκειται για μία παγκόσμια τεχνολογική υποδομή στην οποία ο χρήστης ενός υπολογιστή έχει πρόσβαση και χρησιμοποιεί λογισμικό και δεδομένα, τα οποία είναι εγκατεστημένα ή βρίσκονται εκτός του προσωπικού του υπολογιστικού συστήματος.

Cloud Service Provider (CSP) = Πάροχος υπηρεσιών σύννεφου.

code = κώδικας. 1) Σύνολο μη διφορούμενων κανόνων που καθορίζουν τον τρόπο με τον οποίο δεδομένα μπορούν να παρασταθούν με διάκριτη μορφή (ISO). 2) Ένα ή περισσότερα προγράμματα ή τμήμα προγράμματος.

combination = συνδυασμός. Δεδομένο πλήθος διαφορετικών στοιχείων επιλεγμένων από ένα σύνολο χωρίς να λαμβάνεται υπόψη η διάταξη στην οποία βρίσκονται τα στοιχεία αυτά (ΕΛΟΤ).

command line = γραμμή εντολών. Μια γραμμή οθόνης, συνήθως η τελευταία, από την οποία μπορούν να δοθούν εντολές προς το λειτουργικό σύστημα.

comment = σχόλιο. Γλωσσικό δόμημα που επιτρέπει την εισαγωγή κειμένου σε ένα πρόγραμμα, χωρίς να υπάρχει αντίκτυπος στην εκτέλεση του προγράμματος (ISO).

communication = επικοινωνία. Μια επικοινωνία περιλαμβάνει τέσσερα βασικά στοιχεία: μια πηγή, ένα κανάλι επικοινωνίας, έναν προορισμό και ένα μήνυμα.

communications network = δίκτυο επικοινωνιών. Μια συλλογή διασυνδεμένων μονάδων που παρέχουν υπηρεσίες επικοινωνίας δεδομένων στους σταθμούς που είναι συνδεδεμένοι με το δίκτυο.

communications nodes = επικοινωνιακοί κόμβοι. Οι υπολογιστές που συνδέονται σε ένα δίκτυο.

compile = μεταγλωττίζω. Μεταφράζω σε γλώσσα χαμηλού επιπέδου πρόγραμμα υπολογιστή εκφρασμένο σε γλώσσα προσανατολισμένη στο πρόβλημα (ISO).

compiler = μεταγλωττιστής. Πρόγραμμα υπολογιστή που χρησιμοποιείται για να μεταγλωττίζει (ISO).

computational complexity = υπολογιστική πολυπλοκότητα.

computational theory = θεωρία υπολογισμού.

computer program = πρόγραμμα υπολογιστή. Ακολουθία εντολών κατάλληλων για επεξεργασία. Η επεξεργασία μπορεί να περιλαμβάνει τη χρήση ενός συμβολομεταφραστή, ενός μεταγλωττιστή, ενός διερμηνευτή ή άλλου μεταφραστή, για να προετοιμάσει το πρόγραμμα για εκτέλεση καθώς και την ίδια την εκτέλεση του προγράμματος (ΕΛΟΤ).

computer science = επιστήμη των υπολογιστών, πληροφορική. Ο κλάδος της επιστήμης και τεχνολογίας που ασχολείται με μεθόδους και τεχνικές αναφερόμενες στην επεξεργασία δεδομένων που εκτελούνται με αυτόματα μέσα (ΕΛΟΤ).

condition = συνθήκη. 1) Μια έκφραση σε πρόγραμμα ή διαδικασία που μπορεί να εκτιμηθεί είτε ως αληθής είτε ως ψευδής, όταν εκτελείται το πρόγραμμα ή η διαδικασία. 2) Μια τιμή από ένα σύνολο συγκεκριμένων τιμών που μπορεί να λάβει ένα στοιχείο δεδομένου.

cryptography = κρυπτογραφία. Σύνολο τεχνικών μετατροπής ενός μηνύματος σε κωδικοποιημένη μορφή ώστε να μην γίνεται αντιληπτό από τρίτους.

DBMS – (Database Management System) = Σύστημα Διαχείρισης Βάσεων Δεδομένων – ΣΔΒΔ. Είναι αυτοτελής συλλογή από τμήματα λογισμικού (προγράμματα) για τη δημιουργία, επεξεργασία και συντήρηση των βάσεων δεδομένων.

data = δεδομένα. Παράσταση γεγονότων, εννοιών ή εντολών σε τυποποιημένη μορφή που είναι κατάλληλη για επικοινωνία, ερμηνεία ή επεξεργασία από άνθρωπο ή αυτόματα μέσα (ΕΛΟΤ).

data base = βάση δεδομένων. 1) Συλλογή δεδομένων με συγκεκριμένη δομή για αποδοχή, αποθήκευση και παροχή δεδομένων για πολλούς χρήστες σε κάθε ζήτηση. 2) Συλλογή διασυνδεόμενων δεδομένων οργανωμένων σύμφωνα προς ένα σχήμα βάσης δεδομένων για την εξυπηρέτηση μιας ή περισσοτέρων εφαρμογών.

data dictionary = λεξικό δεδομένων. Ένα λεξικό των στοιχείων δεδομένων που χρησιμοποιούνται σε μια βάση δεδομένων ή σε κάποιο έργο λογισμικού μαζί με τις συσχετίσεις τους με άλλα δεδομένα και προγράμματα που τα χρησιμοποιούν.

data entry = εισαγωγή δεδομένων. Διαδικασία εισαγωγής από ειδικό προσωπικό των δεδομένων στον υπολογιστή για επεξεργασία.

data processing = επεξεργασία δεδομένων. Η συστηματική εκτέλεση πράξεων σε δεδομένα. Παραδείγματα: χειρισμός, συγχώνευση, ταξινόμηση, συμβολομετάφραση, μεταγλώττιση (ΕΛΟΤ).

data structure = δομή δεδομένων. Ένα σύνολο δεδομένων μαζί με τις επιτρεπτές λειτουργίες σε αυτά.

data transmission = μετάδοση δεδομένων. Η κάθε είδους μεταφορά δεδομένων σε απόσταση.

data type = τύπος δεδομένου. Στις γλώσσες προγραμματισμού, ένα σύνολο τιμών μαζί με ένα σύνολο επιτρεπομένων πράξεων (ISO).

data validation = επικύρωση δεδομένων. Διεργασία που χρησιμοποιείται για να διαπιστώνεται η ορθότητα, η πληρότητα ή η λογικότητα των δεδομένων (ISO).

data value = τιμή δεδομένου.

debug = αίρω σφάλματα (σε προγραμματισμό). Ανιχνεύω, εντοπίζω και εξαλείφω σφάλματα σε προγράμματα υπολογιστή ή σε άλλο λογισμικό (ISO).

debugger = αποσφαλματωτής. Βοηθητικό πρόγραμμα -που συνήθως παρέχεται με τις σύγχρονες γλώσσες προγραμματισμού- και καθιστά αποδοτική την έρευνα για τον εντοπισμό σφαλμάτων σε ένα πρόγραμμα.

debugging = αποσφαλμάτωση, εκσφαλμάτωση. Έλεγχος της λογικής ενός προγράμματος για τον εντοπισμό και απομάκρυνση σφαλμάτων.

dimension = διάσταση. Ο αριθμός των δεικτών σε έναν πίνακα. Γενικότερα ο μέγιστος αριθμός και τάξη μιας σειράς από συσχετιζόμενα στοιχεία.

distributed data processing = κατανεμημένη επεξεργασία δεδομένων. Επεξεργασία δεδομένων στην οποία μερικές ή όλες από τις λειτουργίες επεξεργασίας, αποθήκευσης και ελέγχου, εκτός από τις λειτουργίες εισόδου/εξόδου, είναι διασπαρμένες μεταξύ των σταθμών επεξεργασίας δεδομένων (ISO).

documentation = τεκμηρίωση. 1) Διαχείριση τεκμηρίων. Η διαχείριση τεκμηρίων που μπορεί να περιλαμβάνει τις ενέργειες προσδιορισμού, απόκτησης, επεξεργασίας, αποθήκευσης και διανομής τους. 2) Συλλογή τεκμηρίων. Συλλογή τεκμηρίων σ' ένα δεδομένο θέμα (ISO).

editing = σύνταξη (προγραμμάτων, δεδομένων). 1) Μετασχηματισμός τιμών σε παραστάσεις που προδιαγράφονται από δεδομένο μορφότυπο. 2) Διαδικασία εγγραφής-διόρθωσης ενός προγράμματος ή προετοιμασίας δεδομένων με χρήση κατάλληλου προγράμματος.

editor = συντάκτης. Βοηθητικό πρόγραμμα υπολογιστή σχεδιασμένο για να εκτελεί λειτουργίες όπως εισαγωγή, επαναδιευθέτηση, τροποποίηση και διαγραφή δεδομένων σύμφωνα με προδιαγραμμένους κανόνες.

entity = οντότητα. Οποιοδήποτε συγκεκριμένο ή αφηρημένο αντικείμενο ενδιαφέροντος, συμπεριλαμβανομένων και των συσχετίσεων μεταξύ αντικειμένων. Για παράδειγμα ένα πρόσωπο, ένα αντικείμενο, ένα συμβάν ή μια διεργασία ενδιαφέροντος και τα σχετικά δεδομένα που πρέπει να αποθηκεύονται σε μια βάση δεδομένων.

evolutionary computation = εξελικτικός υπολογισμός. Είναι μια διαδικασία που ακολουθεί κάποιο καλά ορισμένο μοντέλο, το οποίο είναι κατανοητό και μπορεί να εκφραστεί με έναν αλγόριθμο,ένα πρωτόκολλο ή μια τοπολογία δικτύου.

executable = εκτελέσιμος. Κώδικας συχνά υπό μορφή αρχείου δίσκου τον οποίο μπορεί να εκτελέσει το εν χρήσει λειτουργικό σύστημα.

executable code = εκτελέσιμος κώδικας. Ένα σύνολο εντολών γλώσσας μηχανής για δεδομένο υπολογιστή ή μικροεπεξεργαστή, που μπορεί να εκτελεστεί απ' ευθείας χωρίς να χρειάζεταιμεταγλώττιση.

expert system = έμπειρο σύστημα. Πρόγραμμα που επιδεικνύει νοήμονα συμπεριφορά σε συγκεκριμένους τομείς και διαδικασίες, ανάλογη ενός ανθρώπου εμπειρογνώμονα με ειδικότητα στον ίδιο τομέα.

expression = έκφραση. Γλωσσικό δόμημα για τον υπολογισμό μιας τιμής από έναν ή περισσότερους τελεστέους. Τελεστέοι μπορεί να είναι κυριολεκτικές σταθερές, αναγνωριστικά (ταυτότητας), αναφορές σε πίνακα, κλήσεις συναρτήσεων κλπ (ISO).

FIFO (First In-First Out) = πρώτος μέσα-πρώτος έξω. Τρόπος λειτουργίας μιας ουράς, κατά τον οποίο το πρώτο στοιχείο που εισάγεται είναι και το πρώτο που μπορεί να εξαχθεί.

FORTRAN (FORmula TRANslation). Η πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου προσανατολισμένη σε επιστημονικά προβλήματα.

FTP (File Transfer Protocol) = πρωτόκολλο μεταφοράς αρχείου. Ένα πρωτόκολλο επικοινωνίας του Internet για τη διεκπεραίωση λειτουργιών μεταφοράς αρχείων.

factorial = παραγοντικό. Το γινόμενο των φυσικών αριθμών 1,2,3,... μέχρι ένα δεδομένο φυσικό ακέραιο (που συμπεριλαμβάνεται) (ΕΛΟΤ).

false = ψευδής. Λογική τιμή που μπορεί να αποδοθεί σε μεταβλητές λογικού τύπου.

field = πεδίο. Προκαθορισμένο τμήμα μιας εγγραφής το οποίο χρησιμοποιείται για κάποια ιδιαίτερη κατηγορία δεδομένων (ISO).

file = αρχείο. Σύνολο από συναφείς εγγραφές που υφίστανται επεξεργασία ως μια μονάδα. Βλ. και record.

function = συνάρτηση, λειτουργία. 1) Μαθηματική οντότητα της οποία η τιμή εξαρτάται από τις τιμές μιας ή περισσοτέρων ανεξαρτήτων μεταβλητών κατά τρόπον που μια μοναδική τιμή της εξηρτημένης μεταβλητής, δηλαδή της ίδιας της συνάρτησης, αντιστοιχεί σε κάθε επιτρεπόμενο συνδυασμό των τιμών καθορισμένων περιοχών των ανεξάρτητων μεταβλητών (ISO). 2) Σε γλώσσες προγραμματισμού, διαδικασία η οποία όταν εκτελείται παρέχει μια τιμή και η κλήση της οποίας μπορεί να χρησιμοποιηθεί ως τελεστέος σε μια έκφραση. Παράδειγμα: η συνάρτηση SIN παρέχει την τιμή του ημx, όταν καλείται με SIN(X) (ISO).

functional programming = συναρτησιακός προγραμματισμός. Προγραμματισμός ο οποίος έχει ως βασικές του παραμέτρους συναρτήσεις

fuzzy logic = ασαφής λογική. Λογική η οποία εξάγεται με βάση τις εμπειρίες και τα συμπεράσματά μας.

gateway = πύλη (δικτύων). Λειτουργική μονάδα που διασυνδέει δύο δίκτυα υπολογιστών με διαφορετικές αρχιτεκτονικές δικτύου (ISO).

GIGO (Garbage In-Garbage Out) = άχρηστα στην είσοδο-άχρηστα στην έξοδο. Όρος που χρησιμοποιείται για να δηλώσει το γεγονός, ότι αν τα δεδομένα εισόδου είναι ακατάλληλα, τότε και τα αποτελέσματα θα είναι επίσης ακατάλληλα.

general-purpose language = γλώσσα γενικής χρήσης. Γλώσσα προγραμματισμού κατάλληλη για μια ευρεία περιοχή εφαρμογών.

genetic algorithm = γενετικός αλγόριθμος. Αλγόριθμος που χρησιμοποιεί την ιδέα της εξέλιξης μέσα από φυσικές επιλογές και διασταυρώσεις, ώστε να βρει λύσεις σε προβλήματα που μπορούν να περιγραφούν με αυτόν τον τρόπο.

graph = γράφος, γράφημα. Ένα σύνολο σημείων, κορυφών ή κόμβων και ένα σύνολο ακμών, τόξων ή γραμμών που ενώνουν μερικά ή όλα τα σημεία του.

hacker = (πληροφορικός) πειρατής. Ενθουσιώδης της πληροφορικής που χρησιμοποιεί τη γνώση του και τα μέσα για να αποκτήσει ανεξουσιοδότητη πρόσβαση σε προστατευμένους πόρους.

host computer = υπολογιστής υποδοχής. Σε δίκτυο υπολογιστών, κάθε υπολογιστής που παρέχει στους τελικούς χρήστες υπηρεσίες, όπως υπολογισμούς και πρόσβαση σε βάσεις δεδομένων και που μπορεί να επιτελεί λειτουργίες ελέγχου του δικτύου (ISO).

high-level language = γλώσσα υψηλού επιπέδου. Γλώσσα προγραμματισμού που προσομοιάζει της φυσικής γλώσσας και που χρειάζεται ένα μεταγλωττιστή ή ένα διερμηνευτή π.χ. Pascal, COBOL, FORTRAN, PROLOG, BASIC κλπ.

index = δείκτης, ευρετήριο. 1) Κατάλογος περιεχομένων ενός αρχείου ή ενός εγγράφου μαζί με τα κλειδιά ή τις παραπομπές για εντοπισμό των περιεχομένων. 2) Στον προγραμματισμό, ένας ακέραιος που αναγνωρίζει τη θέση ενός στοιχείου δεδομένων σε μια ακολουθία στοιχείων δεδομένων. 3) Σύμβολο ή αριθμός που χρησιμοποιείται για να αναγνωρίζει συγκεκριμένη ποσότητα σε πίνακα ομοίων ποσοτήτων. 4) Πίνακας που χρησιμοποιείται για να εντοπίζονται εγγραφές σε ένα σύνολο σειριακών δεδομένων με δείκτες ή σε ένα αρχείο με δείκτες.

informatics = πληροφορική.

information = πληροφορία (ες). Στην επεξεργασία πληροφοριών, γνώση που αφορά πράγματα όπως πράξεις, έννοιες, αντικείμενα, γεγονότα, ιδέες και διεργασίες, που μέσα σε συγκεκριμένο κείμενο έχουν μια ιδιαίτερη σημασία.

information processing = επεξεργασία πληροφοριών. Η συστηματική εκτέλεση πράξεων σε δεδομένα. Παραδείγματα: χειρισμός, συγχώνευση, ταξινόμηση, υπολογισμός, συμβολομετάφραση, μεταγλώττιση (ΕΛΟΤ).

information system = πληροφοριακό σύστημα. Σύστημα υπολογιστή που επεξεργάζεται δεδομένα, ώστε να προκύψουν πληροφορίες που μπορούν να χρησιμοποιηθούν στη λήψη αποφάσεων.

instruction = εντολή. Σε μια γλώσσα προγραμματισμού μια έκφραση που έχει νόημα και η οποία καθορίζει μια πράξη και προσδιορίζει τους τελεστέους της, αν υπάρχουν (ISO).

instruction set = σύνολο εντολών. Το σύνολο των εντολών μηχανής ενός συγκεκριμένου υπολογιστή.

integer = ακέραιος. Ένας από τους αριθμούς 0, +1, -1, +2, -2, ...

interpreter = διερμηνευτής (σε προγραμματισμό). 1) Πρόγραμμα υπολογιστή που χρησιμοποιείται για να διερμηνεύει (ISO). 2) Πρόγραμμα που μεταφράζει και εκτελεί κάθε εντολή μιας γλώσσας υψηλού επιπέδου πριν τη μετάφραση και εκτέλεση της επόμενης εντολής.

iterative = επαναληπτικός.

kernel = πυρήνας. 1) Το μέρος ενός λειτουργικού συστήματος που εκτελεί βασικές λειτουργίες, όπως εκχώρηση πόρων υλικού. Συν. του nucleus. 2) Πρόγραμμα που εκτελείται σε διαφορετικά περιβάλλοντα λειτουργικού συστήματος. 3) Μέρος προγράμματος που πρέπει να είναι στην κύρια μνήμη για να φορτώνει άλλα μέρη του προγράμματος.

knowledge-based system = σύστημα βασισμένο στη γνώση. Σύστημα που εξάγει λογικά συμπεράσματα από ένα σύνολο γνώσεων.

LIFO (Last-In-First-Out) = τελευταίος μέσα-πρώτος έξω. Τρόπος λειτουργίας σε μια δομή δεδομένων, κατά την οποία το στοιχείο που εισήχθη τελευταίο είναι και το πρώτο που αποσύρεται.

LISP (LISt Processing) = γλώσσα LISP. Γλώσσα προγραμματισμού σχεδιασμένη για επεξεργασία λιστών που χρησιμοποιείται εκτενώς για προβλήματα τεχνητής νοημοσύνης.

LOGO. Γλώσσα προγραμματισμού υψηλού επιπέδου κατάλληλη για μικρές ηλικίες.

language = γλώσσα. Σύνολο χαρακτήρων, συμβατικών παραδοχών και κανόνων που χρησιμοποιούνται για να διαβάζουν πληροφορίες (ISO).

language processor = επεξεργαστής γλώσσας. Πρόγραμμα υπολογιστή που εκτελεί λειτουργίες όπως μετάφραση, διερμηνεία και άλλες εργασίες που απαι-

τούνται για την επεξεργασία μιας ορισμένης γλώσσας προγραμματισμού (ISO).

library = βιβλιοθήκη. 1) Αρχείο ή σύνολο συσχετιζομένων αρχείων. 2) Συλλογή συναρτήσεων, κλήσεων, υπορουτινών ή άλλων δεδομένων. 3) Αρχείο δεδομένων το οποίο περιέχει αρχεία και πληροφορίες ελέγχου που τους επιτρέπεται να προσπελαστούν μεμονωμένα.

line editor = συντάκτης γραμμής. Τύπος συντάκτη στον οποίο παρέχεται η δυνατότητα διόρθωσης μόνο σε επίπεδο γραμμής.

linkage editor = συνδέτης. Πρόγραμμα υπολογιστή που χρησιμοποιείται για να δημιουργεί μια φορτώσιμη ενότητα από μια ή περισσότερες αντικειμενικές ενότητες, που έχουν μεταφραστεί ανεξάρτητα ή από φορτώσιμες ενότητες με συμπλήρωση των αντιστοιχιών μεταξύ των κοινών αναφορών που χρησιμοποιούνται από διάφορες αντικειμενικές ενότητες και ενδεχόμενα με μεταθέσεις στοιχείων στη μνήμη (ISO).

linked list = συνδεσμική λίστα. Μια λίστα στην οποία η μετάβαση από έναν κόμβο στον επόμενο γίνεται με τη χρήση ενός δείκτη (pointer), που αποτελεί μέρος του κόμβου.

linker = συνδέτης. Βλ. linkage editor.

linking loader = συνδετικός φορτωτής. Βοηθητικός επεξεργαστής που εκτελεί σύνδεση και φόρτωση ενός μεταφρασμένου προγράμματος με σκοπό την εκτέλεσή του.

list = λίστα. Διαταγμένο σύνολο στοιχείων (ISO).

loader = φορτωτής. Πρόγραμμα υπολογιστή που μεταφέρει δεδομένα στην κύρια μνήμη.

local area network (LAN) = τοπικό δίκτυο. Δίκτυο υπολογιστών τοποθετημένο στο χώρο ενός χρήστη σε περιορισμένη γεωγραφική περιοχή (ISO).

local variable = τοπική μεταβλητή. Μεταβλητή της οποίας το όνομα και η τιμή μπορούν να χρησιμοποιηθούν μόνο μέσα στην ενότητα του προγράμματος όπου ορίζεται.

logic programming = λογικός προγραμματισμός. Μέθοδος κατασκευής προγραμμάτων ως σύνολα λογικών κανόνων με προκαθορισμένους αλγόριθμους για την επεξεργασία δεδομένων εισόδου.

logical constant = λογική σταθερά. Σταθερά με τιμή αληθής ή ψευδής.

logical expression = λογική έκφραση. Έκφραση που περιέχει λογικούς τελεστές και τελεστέους και που μπορεί να αντικατασταθεί με την τιμή αληθής ή ψευδής.

loop = βρόχος. Σύνολο εντολών που μπορεί να εκτελεστεί επανειλημμένα, όσο ισχύει μια ορισμένη συνθήκη (ISO).

machine language = γλώσσα μηχανής. Γλώσσα χαμηλού επιπέδου που οι εντολές της αποτελούνται μόνο από εντολές μηχανής (ISO).

machine learning = μηχανική μάθηση. Είναι η μάθηση σε ένα γνωστικό σύστημα όπως γίνεται αντιληπτή στην καθημερινή ζωή και συνδέεται με δύο βασικές ιδιότητες α) την ικανότητα στην πρόσκτηση γνώσης κατά την αλληλεπίδραση με το περιβάλλον και β) την ικανότητα να βελτιώνει με την επανάληψη τον τρόπο εκτέλεσης μία ενέργειας.

menu (list of options) = κατάλογος επιλογών. Κατάλογος δυνατών εργασιών που παρέχει σε μια συγκεκριμένη φάση εργασίας ένα προϊόν λογισμικού.

multiprocessing = πολυεπεξεργασία. Τρόπος εκμετάλλευσης που επιτρέπει παράλληλη επεξεργασία από δύο ή περισσότερους επεξεργαστές ενός πολυεπεξεργαστή (ISO).

multiprogramming = πολυπρογραμματισμός. Τρόπος εκμετάλλευσης που επιτρέπει διεμπλεκόμενη εκτέλεση δύο ή περισσοτέρων προγραμμάτων υπολογιστή από έναν και μόνο επεξεργαστή (ISO).

multitasking = πολυδιεργασία. Τρόπος εκμετάλλευσης που επιτρέπει τη συντρέχουσα διεξαγωγή ή τη διεμπλεκόμενη εκτέλεση δύο ή περισσοτέρων στοιχειωδών εργασιών (ISO).

multiuser system = σύστημα πολλών χρηστών. Σύστημα με δυνατότητα πρακτικά ταυτόχρονης υποστήριξης πολλών χρηστών.

NOT operation = πράξη ΟΧΙ, άρνηση, λογική αντιστροφή. Η μοναδιαία πράξη άλγεβρας Boole της οποίας το αποτέλεσμα έχει τιμή αντίθετη από αυτήν του τελεστέου (ISO).

network = δίκτυο. Ενα σύνολο κόμβων μαζί με τους κλάδους διασύνδεσής τους (ISO).

network topology = τοπολογία δικτύου. Η διάταξη των συνδέσεων και κόμβων σε ένα δίκτυο.

networking = δικτύωση. Η σύνδεση γεωγραφικά χωρισμένων υπολογιστών μέσω γραμμών επικοινωνίας.

node = κόμβος. 1) Σε δίκτυο δεδομένων, σημείο

όπου μία ή περισσότερες λειτουργικές μονάδες διασυνδέουν κανάλια ή κυκλώματα δεδομένων (ISO). 2) Η παράσταση μιας κατάστασης ή ενός γεγονότος με τη χρήση ενός σημείου σε ένα διάγραμμα. 3) Τα στοιχεία μιας δομής δένδρου ή συνδεδεμένης λίστας.

nucleus = πυρήνας. Το τμήμα προγράμματος ελέγχου που διαμένει στην κύρια μνήμη (ISO).

numeric character = αριθμητικός χαρακτήρας. Χαρακτήρας που ανήκει στο σύνολο των ψηφίων 0 έως 9.

OSI (Open Systems Interconnection) = διασύνδεση ανοικτών συστημάτων. Η διασύνδεση συστημάτων υπολογιστών σύμφωνα με τα πρότυπα ISO και τις συστάσεις CCITT για την ανταλλαγή δεδομένων (ISO).

object oriented language = αντικειμενοστρεφής γλώσσα. Γλώσσα που αντανακλά τις έννοιες του αντικειμενοστρεφούς προγραμματισμού.

object oriented programming = αντικειμενοστρεφής προγραμματισμός. Μέθοδος για δόμηση προγραμμάτων ως ιεραρχικά οργανωμένες τάξεις, που περιγράφουν τα δεδομένα και τις λειτουργίες αντικειμένων, που μπορούν να αλληλεπιδρούν με άλλα αντικείμενα.

object program = πρόγραμμα αντικειμένου, τελικό πρόγραμμα. Πρόγραμμα υπολογιστή σε μια τελική γλώσσα που έχει μεταφραστεί από μια γλώσσα πηγής (ISO).

operand = τελεστέος. Μια οντότητα στην οποία εφαρμόζεται μια πράξη (ΕΛΟΤ).

operating system = λειτουργικό σύστημα. Λογισμικό που ελέγχει την εκτέλεση των προγραμμάτων και που μπορεί να παρέχει υπηρεσίες όπως εκχώρηση πόρων, χρονοπρογραμματισμό, έλεγχο εισόδου/εξόδου και διαχείριση δεδομένων. Αν και τα λειτουργικά συστήματα είναι κατά το μέγιστο μέρος λογισμικό, είναι δυνατές μερικές ή ολικές υλοποιήσεις με υλικό (ΕΛΟΤ).

operation = πράξη. Μια σαφώς ορισμένη ενέργεια, η οποία όταν εφαρμόζεται σε οποιοδήποτε επιτρεπτό συνδυασμό γνωστών οντοτήτων, παράγει μια νέα οντότητα. Για παράδειγμα, η διεργασία της πρόσθεσης στην αριθμητική, όταν προστίθονται το ένα και το δύο με αποτέλεσμα τρία, οι αριθμοί ένα και δύο είναι οι τελεστέοι, ο αριθμός τρία είναι το αποτέλεσμα και το σημείο συν είναι ο τελεστής που δείχνει ότι η πράξη που εκτελείται είναι η πρόσθεση (ΕΛΟΤ).

operator = τελεστής, χειριστής. 1) Σύμβολο που παριστάνει τη φύση μιας πράξης που πρόκειται να εκτελεστεί (ΕΛΟΤ). 2) Πρόσωπο που χειρίζεται μια συσκευή.

parallel programming = παράλληλος προγραμματισμός. Η ανάπτυξη εφαρμογών που εκμεταλλεύονται ταυτόχρονα την ύπαρξη πολλαπλών επεξεργαστών σε ένα υπολογιστικό σύστημα με σκοπό την αύξηση των υπολογιστικών επιδόσεων και τη μείωση του χρόνου εκτέλεσης της εφαρμογής.

Pascal. Δημοφιλής γλώσσα τρίτης γενιάς που εισήχθη το 1971.

pattern recognition = αναγνώριση προτύπων. Η αναγνώριση ταυτότητας σχημάτων, μορφών ή διατάξεων με αυτόματα μέσα.

pointer = δείκτης. 1) Στοιχείο δεδομένου που περιέχει τη διεύθυνση ενός άλλου στοιχείου δεδομένου. 2) Οπτικά παρουσιαζόμενο σύμβολο στην οθόνη το οποίο ένας χρήστης μετακινεί με μια συσκευή στίξης, όπως ένα ποντίκι.

procedural = διαδικασιακός, διαδικαστικός.

procedural language = διαδικασιακή γλώσσα. Γλώσσα προσανατολισμένη στο πρόβλημα που διευκολύνει την έκφραση μιας διαδικασίας με τη μορφή ρητά εκφρασμένου αλγόριθμου (ISO).

procedure = διαδικασία. Στις γλώσσες προγραμματισμού, μια ομάδα εντολών με ή χωρίς τυπικές παραμέτρους, η εκτέλεση της οποίας προκαλείται μέσω μιας κλήσης διαδικασίας.

processing = επεξεργασία. Η εκτέλεση λογικών πράξεων και υπολογισμών σε δεδομένα συμπεριλαμβανομένης και της προσωρινής διατήρησης των δεδομένων στη μνήμη του επεξεργαστή.

program = πρόγραμμα (υπολογιστή). Ακολουθία εντολών κατάλληλων για επεξεργασία. Η επεξεργασία μπορεί να περιλαμβάνει τη χρήση ενός συμβολομεταφραστή, ενός μεταγλωττιστή, ενός διερμηνευτή ή άλλου μεταφραστή, για να προετοιμάσει το πρόγραμμα για εκτέλεση καθώς και την ίδια την εκτέλεση του προγράμματος (ΕΛΟΤ).

program documentation = τεκμηρίωση προγράμματος. Έγγραφα ή άλλα μέσα που παρέχουν την περιγραφή του προγράμματος. Η τεκμηρίωση προγράμματος κανονικά περιλαμβάνει ένα αντίγραφο λίστας προγράμματος, αποτελέσματα δοκιμών, ιστορικό τροποιήσεων στο πρόγραμμα και άλλες πληροφορίες.

program specification = προδιαγραφή προγράμματος. Τεκμήριο που περιγράφει τη δομή και λειτουργίες ενός προγράμματος με επαρκή λεπτομέρεια, ώστε να επιτρέπει τον προγραμματισμό και να διευκολύνει τη συντήρηση (ISO).

program testing = δοκιμή προγράμματος. Ενα βήμα της ανάπτυξης προγράμματος όπου ένα πλήρες πρόγραμμα δοκιμάζεται για σφάλματα. Μια δοκιμή προγράμματος εμπεριέχει την αποσφαλμάτωση του προγράμματος.

program testing and debugging = δοκιμή και αποσφαλμάτωση προγράμματος. Διαδικασίες εντοπισμού και διόρθωσης τυχόν σφαλμάτων ενός προγράμματος.

program verification = επαλήθευση προγράμματος. Αποδεικνύει ότι ένα πρόγραμμα συμπεριφέρεται σύμφωνα προς τις προδιαγραφές του.

programmer = προγραμματιστής. Πρόσωπο υπεύθυνο για το σχεδιασμό, εγγραφή, έλεγχο, διόρθωση, συντήρηση και τεκμηρίωση ενός προγράμματος.

programming = προγραμματισμός. Η διαδικασία δημιουργίας προγραμμάτων υπολογιστή.

programming language = γλώσσα προγραμματισμού. Τεχνητή γλώσσα σχεδιασμένη για να δημιουργεί ή να εκφράζει προγράμματα (ISO).

protocol = πρωτόκολλο. Σύνολο κανόνων της σημασιολογίας και του συντακτικού που καθορίζει τη συμπεριφορά των λειτουργικών μονάδων στην επίτευξη επικοινωνίας (ISO).

pseudocode = ψευδοκώδικας. 1) Κώδικας που απαιτεί μετάφραση πριν από την εκτέλεση (ISO). 2) Τρόπος αποτύπωσης αλγορίθμων με χρήση προκαθορισμένων λέξεων κλειδιών.

push = ωθώ, σπρώχνω. Εισάγω ένα στοιχείο δεδομένων στην κορυφή μιας στοίβας.

query = ερώτηση. Μια αίτηση για δεδομένα από βάση δεδομένων που βασίζεται σε ειδικές συνθήκες. Παράδειγμα, η ερώτηση για διαθεσιμότητα μιας θέσης σε ένα σύστημα δέσμευσης πτήσης.

query language = γλώσσα ερωταποκρίσεων. Μια γλώσσα χειρισμού δεδομένων για τελικούς χρήστες προκειμένου να ανακαλούν ή να τροποποιούν δεδομένα σε μια βάση δεδομένων

queue = ουρά. Δομή δεδομένων με δύο άκρα στην οποία νέα στοιχεία εισάγονται από το ένα άκρο και υπάρχοντα στοιχεία εξέρχονται από το άλλο άκρο. Βλ. και FIFO.

quicksort = γρήγορη ταξινόμηση. Αλγόριθμος ταξινόμησης που βασίζεται στο διαμερισμό του συνόλου των στοιχείων σε δύο μέρη, στην ανταλλαγή των πιο απομακρυσμένων στοιχείων και στην αναδρομική κλήση του για καθένα από τα δύο μέρη. Η μέθοδος αυτή είναι η καλύτερη γνωστή μέχρι σήμερα για εσωτερική ταξινόμηση.

relation data model (RDM) = σχεσιακό μοντέλο δεδομένων. Είναι το είδος του λογικού μοντέλου βάσεων δεδομένων που μπορεί να συνδυάζει τα δεδομένα ενός πίνακα με τα δεδομένα ενός άλλου, αρκεί οι δυο πίνακες να έχουν ένα κοινό στοιχείο δεδομένων.

record = εγγραφή. Στις γλώσσες προγραμματισμού, συνάθροισμα που αποτελείται από αντικείμενα σχετικά με δεδομένα με πιθανόν διαφορετικά ιδιοχαρακτηριστικά, στα οποία αντικείμενα έχουν συνήθως προσαρτηθεί αναγνωριστικά ταυτότητας (ISO).

recursion = αναδρομή. Κατάσταση κατά την οποία μια διαδικασία ή συνάρτηση καλεί τον εαυτό της.

recursive function = αναδρομική συνάρτηση. Μια συνάρτηση της οποίας οι τιμές είναι φυσικοί αριθμοί που παράγονται από φυσικούς αριθμούς με τύπους αντικατάστασης, στους οποίους η ίδια η συνάρτηση είναι ένας τελεστέος (ΕΛΟΤ).

repeater = επαναλήπτης. Σε κόμβο τοπικού δικτύου, μια συσκευή που αναγεννά σήματα για να επεκτείνει την εμβέλεια μετάδοσης μεταξύ σταθμών δεδομένων ή για να διασυνδέει δύο κλάδους.

reserved word = δεσμευμένη λέξη. Λέξη γλώσσας πηγής η σημασία της οποίας καθορίζεται από τους κανόνες αυτής της γλώσσας και δεν μπορεί να αλλάξει για την ευκολία οποιουδήποτε προγράμματος εκφρασμένου σε αυτή τη γλώσσα. Μάλιστα στις περισσότερες περιπτώσεις προγραμμάτων εκφρασμένων στη γλώσσα πηγής, μπορεί να απαγορεύεται η χρήση τέτοιων λέξεων σε άλλα συμφραζόμενα του προγράμματος. Παραδείγματα: 1. Η λέξη SIN μπορεί να είναι μια δεσμευμένη λέξη για την κλήση μιας υπορουτίνας υπολογισμού του ημιτόνου. 2. Λέξεις της COBOL όπως OCCURS, MOVE, COMPUTE κ.α.

retrieval = ανάκτηση. Διαδικασία αναζήτησης ενός στοιχείου με σκοπό τη λήψη (ανάγνωση) του περιεχομένου του.

robot = ρομπότ. Μια μηχανική συσκευή η οποία μπορεί να προγραμματιστεί για να εκτελεί κάποια

εργασία χειρισμού ή μετακίνηση από ένα τόπο σε άλλο υπό συνθήκες αυτομάτου ελέγχου.

robotics = ρομποτική. Οι τεχνικές που χρησιμοποιούνται για σχεδιασμό, κατασκευή και χρήση ρομπότ.

router = δρομολογητής. Υπολογιστής που προσδιορίζει τη διαδρομή κυκλοφορίας δικτύου. Η επιλογή διαδρομής γίνεται από αρκετές διαδρομές που βασίζονται σε πληροφορίες, οι οποίες λαμβάνονται από ειδικά πρωτόκολλα και αλγορίθμους, που προσπαθούν να αναγνωρίζουν τη συντομότερη ή καλύτερη διαδρομή και άλλα κριτήρια.

routine = ρουτίνα. Ένα πρόγραμμα καλούμενο από άλλο πρόγραμμα, που μπορεί να έχει γενική ή συχνή χρήση (ΕΛΟΤ).

running time = χρόνος εκτέλεσης. Ο χρόνος που απαιτείται για την εκτέλεση ενός προγράμματος.

SQL (Structured Query Language) = δομημένη γλώσσα ερωταπαντήσεων. Γλώσσα ερωταπαντήσεων πολύ υψηλού επιπέδου που χρησιμοποιείται για αναζητήσεις σε βάσεις δεδομένων.

search = αναζητώ, ψάχνω. Εξετάζω σύνολο στοιχείων δεδομένων για να εντοπίσω ένα ή περισσότερα στοιχεία που έχουν μια δεδομένη ιδιότητα (ISO).

searching = αναζήτηση, ψάξιμο.

searching techniques = τεχνικές αναζήτησης. Σύνολο τεχνικών με βάση τις οποίες επιτυγχάνεται η λειτουργία της αναζήτησης.

selection sort = ταξινόμηση με επιλογή. Τεχνική ταξινόμησης κατά την οποία βρίσκεται το ελάχιστο στοιχείο μιας ομάδας και αντιμετατίθεται με το πρώτο, στη συνέχεια βρίσκεται το ελάχιστο στοιχείο των υπολοίπων και αντιμετατίθεται με το δεύτερο κ.ο.κ.

sequential = ακολουθιακός. Τρόπος επεξεργασίας κατά τον οποίο δύο ή περισσότερες πράξεις εκτελούνται η μία μετά την άλλη (ΕΛΟΤ).

sequential search = ακολουθιακή ή σειριακή αναζήτηση.

simplex line = ημιαμφίδρομη γραμμή, μονόδρομη γραμμή. Μια γραμμή επικοινωνίας που χρησιμοποιείται για τη μετάδοση δεδομένων κατά μία μόνο κατεύθυνση.

software = λογισμικό. Πνευματική δημιουργία που περιλαμβάνει τα προγράμματα, τις διαδικασίες, τους κανόνες και οποιαδήποτε σχετική τεκμηρίωση που αναφέρεται στη λειτουργία ενός συστήματος επεξεργασίας δεδομένων (ΕΛΟΤ).

software development environment = περιβάλλον ανάπτυξης λογισμικού. Σύνολο μεταφραστικών προγραμμάτων και άλλων εργαλείων ανάπτυξης λογισμικού που χρησιμοποιούνται στη δημιουργία προγραμμάτων εφαρμογών.

software development tool = εργαλείο ανάπτυξης λογισμικού. Ένα σύνολο προγραμμάτων που βοηθά στην ανάπτυξη ορισμένων τύπων λογισμικού εφαρμογών.

software engineer = μηχανικός λογισμικού. Πρόσωπο που σχεδιάζει και γράφει προγράμματα υπολογιστών σύμφωνα με τις αρχές της τεχνολογίας λογισμικού.

software life cycle = κύκλος ζωής λογισμικού. Ο κύκλος σχεδίασης, ανάπτυξης, εγκατάστασης και συντήρησης λογισμικού.

software package = πακέτο λογισμικού. Πλήρες και τεκμηριωμένο σύνολο προγραμμάτων που παρέχεται σε έναν αριθμό χρηστών για μια εφαρμογή (ISO).

software piracy = πειρατεία λογισμικού. Παράνομη αναπαραγωγή προϊόντων λογισμικού.

software reusability = δυνατότητα επαναχρήσης λογισμικού, επαναχρηστικότητα λογισμικού. Επιδιωκόμενη ιδιότητα ενός προϊόντος λογισμικού σύμφωνα με την οποία μπορεί το εν λόγω προϊόν να χρησιμοποιηθεί και για τη σύσταση ενός άλλου.

software testing = έλεγχος λογισμικού. Φάση της ανάπτυξης ενός προϊόντος λογισμικού κατά την οποία το προϊόν ελέγχεται ως προς την αναμενόμενη λειτουργία του.

software tools = εργαλεία λογισμικού. Ειδικά προγράμματα που διευκολύνουν συγκεκριμένες εργασίες της ανάπτυξης λογισμικού.

sorting = ταξινόμηση. Η διαδικασία τοποθέτησης των στοιχείων δεδομένων σε μια δομή δεδομένων με αύξουσα ή φθίνουσα σειρά.

source code = πηγαίος κώδικας. Η είσοδος σε μεταγλωττιστή ή συμβολομεταφραστή που έχει γραφεί σε μια γλώσσα πηγής.

source program = πηγαίο πρόγραμμα, αρχικό πρόγραμμα. Πρόγραμμα υπολογιστή εκφρασμένο σε μια γλώσσα πηγής (ISO).

specification = προδιαγραφή. Λεπτομερής διατύπωση υπό μορφή τεκμηρίου, που παρέχει οριστική περιγραφή ενός συστήματος με σκοπό την ανάπτυξη ή επικύρωση του συστήματος (ISO).

square root = τετραγωνική ρίζα.

stack = στοίβα. Δομή δεδομένων με ένα μόνο άκρο, στην οποία το τελευταίο στοιχείο που μπήκε είναι και το πρώτο που μπορεί να εξαχθεί. Βλ. και LIFO.

statement = εντολή, πρόταση (προγράμματος). Γλωσσικό δόμημα που παριστάνει ένα βήμα σε μια ακολουθία ενεργειών ή σε ένα σύνολο δηλώσεων (ISO).

structured programming = δομημένος προγραμματισμός. Μέθοδος για την κατασκευή προγραμμάτων που χρησιμοποιεί μόνο ιεραρχικά εμπερικλειόμενα κατασκευάσματα, καθένα από τα οποία έχει ένα απλό σημείο εισόδου και ένα εξόδου. Τρεις τύποι ελέγχου χρησιμοποιούνται στο δομημένο προγραμματισμό: ακολουθία, επιλογή και επανάληψη.

swarm intelligence = νοημοσύνη σμήνους. Μέθοδος διαδικασιών βελτιστοποίησης υπολογισμών που προέκυψαν παρατηρώντας πρότυπα που υπάρχουν στη φύση.

subprogram = υποπρόγραμμα. Ενα πρόγραμμα καλούμενο από άλλο πρόγραμμα, σε αντίθεση με ένα κύριο πρόγραμμα.

subroutine = υπορουτίνα. Ακολουθιακό σύνολο εντολών ή προτάσεων που μπορούν να χρησιμοποιηθούν σε ένα ή περισσότερα σημεία ενός προγράμματος υπολογιστή.

switching elements = στοιχεία μεταγωγής. Πρόκειται για τις ενδιάμεσες συσκευές που συνδέουν τις γραμμές μετάδοσης και επιφορτίζονται με το έργο της δρομολόγησης των δεδομένων από τη μια γραμμή στην άλλη ή από το ένα δίκτυο στο άλλο.

symbolic language = συμβολική γλώσσα.

symbolic logic = συμβολική λογική. Ο επιστημονικός κλάδος στον οποίο οι συλλογισμοί και οι πράξεις αντιμετωπίζονται με τη χρήση μιας τεχνητής γλώσσας σχεδιασμένης έτσι, ώστε να αποφεύγονται τα διφορούμενα και η λογική ανεπάρκεια των φυσικών γλωσσών (ΕΛΟΤ).

system analysis = ανάλυση συστήματος. Συστηματική έρευνα ενός πραγματικού ή σχεδιασμένου συστήματος για να προσδιοριστούν οι απαιτήσεις πληροφοριών και οι διεργασίες του συστήματος και πώς αυτά συνδέονται μεταξύ τους και με οποιοδήποτε άλλο σύστημα (ISO).

system software = λογισμικό συστήματος. Λογισμικό ανεξάρτητο της εφαρμογής, που υποστηρίζει την εκτέλεση του λογισμικού εφαρμογής (ISO).

table = πίνακας. 1) Παράθεση δεδομένων καθένα από τα οποία μπορεί να προσδιοριστεί μονοσήμαντα μέσω μιας ή περισσοτέρων μεταβλητών (ISO). 2) Ένα αντικείμενο βάσης δεδομένων που αποτελείται από μια ομάδα γραμμών (εγγραφές), οι οποίες χωρίζονται σε στήλες (πεδία) που περιέχουν δεδομένα.

telecommunication = τηλεπικοινωνία. 1) Η μετάδοση σημάτων ελέγχου και πληροφοριών μεταξύ δύο ή περισσοτέρων θέσεων, όπως από τηλέγραφο, ραδιόφωνο ή τηλεόραση. 2) Η μετάδοση δεδομένων μεταξύ συστημάτων υπολογιστών από γραμμές τηλεπικοινωνίας και μεταξύ ενός συστήματος υπολογιστή και απομακρυσμένων συσκευών.

test = δοκιμή. Η λειτουργία μιας μονάδας και η σύγκριση του επιτευχθέντος αποτελέσματος με το προβλεπόμενο, προκειμένου να γίνει αποδεκτή ή μη η μονάδα. Π.χ. δοκιμή συσκευής, δοκιμή προγράμματος.

test data = δεδομένα δοκιμής. Τα δεδομένα που χρησιμοποιούνται για ένα ελεγκτικό πρόβλημα (ISO).

text editor = συντάκτης κειμένου. Πρόγραμμα υπολογιστή που δίνει στο χρήστη τη δυνατότητα να δημιουργεί και να αναθεωρεί κείμενα. Ο συντάκτης κειμένου, αν και επιτελεί το ίδιο έργο με τον επεξεργαστή κειμένου, συνήθως έχει μικρότερες δυνατότητες από τον τελευταίο.

third generation language = γλώσσα τρίτης γενεάς. Η γενιά των γλωσσών προγραμματισμού που εμφανίστηκαν τη δεκαετία του 60 και ακολούθησαν τις γλώσσες μηχανής και τις συμβολικές γλώσσες. Είναι γλώσσες υψηλού επιπέδου που χρησιμοποιούν κάποιο μεταφραστικό πρόγραμμα.

topology = τοπολογία. Ο χάρτης ή το σχέδιο ενός δικτύου. Η φυσική τοπολογία περιγράφει τη διάταξη των καλωδίων και η λογική ή ηλεκτρική τοπολογία περιγράφει τη ροή των μηνυμάτων.

translate = μεταφράζω, μεταφέρω. 1) Σε γλώσσες προγραμματισμού, ο μετασχηματισμός όλου ή μέρους ενός προγράμματος που εκφράζεται σε μια γλώσσα προγραμματισμού, από τη γλώσσα αυτή σε μια άλλη γλώσσα ή σε μια γλώσσα μηχανής κατάλληλη για εκτέλεση. 2) Σε γραφικά υπολογιστή, η μετατόπιση μιας εικόνας σε χώρο οπτικής παρουσίασης από μια θέση στην άλλη χωρίς την περιστροφή της.

translator = μεταφραστής. Πρόγραμμα υπολογιστή που μεταφράζει από μια γλώσσα σε άλλη γλώσσα και ιδιαίτερα από μια γλώσσα προγραμματισμού σε άλλη γλώσσα προγραμματισμού (ISO).

transmission = εκπομπή. Η αποστολή δεδομένων από ένα μέρος για λήψη οπουδήποτε.

transmission medium = μέσο εκπομπής. Το φυσικό μέσο που μεταφέρει σήματα μεταξύ σταθμών δεδομένων π.χ. σύρμα συνεστραμένου ζεύγους, οπτική ίνα, ομοαξονικό καλώδιο.

transmit = εκπέμπω.

transmitter = πομπός.

tree = δένδρο. Δομή δεδομένων που αποτελείται από κόμβους, οι οποίο συνδέονται με ακμές. Σε κάθε κόμβο καταλήγει μία μόνο ακμή, αλλά μπορεί να ξεκινούν καμία, μία ή περισσότερες ακμές. Σε ένα μόνο κόμβο που αποκαλείται ρίζα, δεν καταλήγει καμία ακμή.

UNIX operating system = λειτουργικό σύστημα UNIX. Λειτουργικό σύστημα που αναπτύχθηκε από τα εργαστήρια BELL και το χαρακτηρίζει ο πολυπρογραμματισμός σε περιβάλλον πολλαπλών χρηστών. Το UNIX αναπτύχθηκε αρχικά για χρήση σε μινι-υπολογιστές αλλά προσαρμόστηκε για μεγάλους υπολογιστές καθώς και για μικρο-υπολογιστές.

unary operation = μοναδιαία πράξη, μονοτελής πράξη. Μια πράξη με έναν και μόνον έναν τελεστέο. Παράδειγμα: η άρνηση (ΕΛΟΤ).

user = χρήστης. Πρόσωπο που χρησιμοποιεί το σύστημα του υπολογιστή.

user friendly system = σύστημα φιλικό προς το χρήστη. Χαρακτηριστικό ενός συστήματος ή εφαρμογής που δηλώνει ευκολία στη χρήση του.

user interface = διεπαφή χρήστη. Υλικό, λογισμικό ή και τα δύο, με τα οποία επιτρέπεται να αλληλεπιδρά ο χρήστης και να εκτελεί λειτουργίες στο σύστημα, στο πρόγραμμα ή στη συσκευή.

user manual = εγχειρίδιο χρήστη

validation = επικύρωση. Διεργασία που χρησιμοποιείται για να διαπιστώνεται η ορθότητα, η πληρότητα ή λογικότητα των δεδομένων (ISO).

validity check = έλεγχος εγκυρότητας. Ελεγχος για να προσδιορίζει, αν μια ομάδα κώδικα είναι πραγματικά χαρακτήρες του δεδομένου κώδικα σε χρήση.

variable = μεταβλητή. 1) Οντότητα της οποίας η τιμή μπορεί να είναι απροσδιόριστη -ή απροσδιόριστη μέσα σε γνωστά όρια- μέχρι να αποδοθεί μια πραγματική τιμή σε αυτή σε δεδομένη εφαρμογή (ΕΛΟΤ). 2) Στις γλώσσες προγραμματισμού, ένα γλωσσικό αντικείμενο που μπορεί να λάβει διάφορες τιμές, μία κάθε φορά. Οι τιμές μιας μεταβλητής περιορίζονται συνήθως σε έναν τύπο δεδομένων. 3) Ένα όνομα που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου, του οποίου η τιμή μπορεί ν' αλλάζει κατά τη διάρκεια λειτουργίας του προγράμματος.

verification = επαλήθευση, επιβεβαίωση. Η ενέργεια του προσδιορισμού αν μια λειτουργία έχει επιτελεσθεί ορθά.

version = εκδοχή, παραλλαγή, έκδοση. Όρος που χρησιμοποιείται για την αναγνώριση ενός συγκεκριμένου προϊόντος λογισμικού. Όταν το προϊόν τίθεται σε κυκλοφορία ή πωλείται για πρώτη φορά, αποκαλείται πρώτη έκδοση. Καθώς το προϊόν τροποποιείται ή βελτιώνεται, γίνονται διαθέσιμες μεταγενέστερες εκδόσεις.

virus = ιός. Ένα αυτοαναπαραγόμενο πρόγραμμα το οποίο «μολύνει» και μπορεί να καταστρέψει άλλα προγράμματα και δεδομένα.

WYSIWYG (What You See Is What You Get) = ό,τι βλέπεις είναι αυτό που παίρνεις. Όρος που χρησιμοποιείται για να περιγράψει συστήματα, στα οποία υπάρχει η δυνατότητα να αποτυπωθεί στον εκτυπωτή ό,τι ακριβώς φαίνεται στην οθόνη (προφέρεται γουίσγουινγκ).

wide area network (WAN) = δίκτυο ευρείας περιοχής. Δίκτυο που παρέχει υπηρεσίες επικοινωνιών σε γεωγραφική περιοχή ευρύτερη από αυτή που εξυπηρετείται από τοπικό δίκτυο ή το μητροπολιτικό δίκτυο και που μπορεί να χρησιμοποιεί ή να παρέχει δημόσιες ευκολίες επικοινωνίας.

window = παράθυρο. 1) Τμήμα μιας επιφάνειας οπτικής παρουσίασης στην οποία μπορούν να παρουσιαστούν εικόνες που ανήκουν σε ιδιαίτερη εφαρμογή. Διαφορετικές εφαρμογές μπορούν να παρουσιαστούν οπτικά ταυτόχρονα σε διαφορετικά παράθυρα. 2) Περιοχή της οθόνης με ορατά σύνορα μέσα στην οποία παρουσιάζονται οπτικά πληροφορίες. Ένα παράθυρο μπορεί να είναι μικρότερο ή του ιδίου μεγέθους με την οθόνη.

XML - (Extensible Markup Language). Γλώσσα η οποία χρησιμοποιώντας ένα σύνολο κανόνων επιτρέπει την αναπαράσταση δεδομένων με τρόπο τέτοιο που να γίνεται κατανοητός όχι μόνο από τους ανθρώπους αλλά και από τους υπολογιστές.

Σημ. ISO = Διεθνής Οργανισμός Τυποποίησης, ΕΛΟΤ = Ελληνικός Οργανισμός Τυποποίησης



ΙΕΠ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης