Περιφερειακή Διεύθυνση Πρωτοβάθμιας και Δευτεροβάθμιας Εκπαίδευσης Βορείου Αιγαίου

Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

Β' Λυκείου

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΠΕΡΙΦΕΡΕΙΑΚΗ ΔΙΕΥΘΥΝΣΗ ΠΡΩΤΟΒΑΘΜΙΑΣ & ΔΕΥΤΕΡΟΒΑΘΜΙΑΣ
ΕΚΠΑΙΔΕΥΣΗΣ ΒΟΡΕΙΟΥ ΑΙΓΑΙΟΥ

Η Πληροφορική αποτελεί αδιαμφισβήτητα έναν εξελισσόμενο τομέα της εκπαίδευσης και της καθημερινότητάς μας, με ανεξάντλητες δυνατότητες, συνεχείς επικαιροποιήσεις και ανεξερεύνητα μονοπάτια. Η συγγραφή, λοιπόν, ενός επικουρικού εγχειριδίου πάνω στο γνωστικό αυτό αντικείμενο προϋποθέτει βαθιά γνώση των ιδιαιτεροτήτων κάθε θεματικής ενότητας του σχολικού βιβλίου, διδακτική εμπειρία, αλλά, πάνω απ' όλα, αγάπη και ενδιαφέρον για το χρήστη (εκπαιδευτικό και μαθητή).

Το βιβλίο αυτό, με τίτλο «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ», είναι γραμμένο από λειτουργούς της εκπαίδευσης που δεν έχουν πάψει στιγμή να διδάσκονται και να διδάσκουν, με σεβασμό και προσήλωση στις αρχές της παιδαγωγικής και του γνωστικού τους αντικειμένου. Απευθύνεται σε όλους εσάς -τους μαθητές και τους μάχιμους εκπαιδευτικούς Πληροφορικής - που επιζητάτε περαιτέρω καθοδήγηση στο εξειδικευμένο αυτό κομμάτι της τεχνολογίας.

Προσωπικά εύχομαι η προσπάθεια αυτή να αποτελέσει το εφαλτήριο για έρευνα και επιστημονική αναζήτηση, τόσο για τα μέλη της συγγραφικής ομάδας, όσο και για τους εκπαιδευτικούς που θα το χρησιμοποιήσουν, καθώς και πηγή έμπνευσης και βελτίωσης για τους μαθητές μας.

Με ιδιαίτερη εκτίμηση,

Η ΠΕΡΙΦΕΡΕΙΑΚΗ Δ/ΝΤΡΙΑ
Π. & Δ. ΕΚΠ/ΣΗΣ ΒΟΡΕΙΟΥ ΑΙΓΑΙΟΥ

ΜΑΡΙΑ Χ. ΠΑΠΑΔΑΝΙΗΛ

Εισαγωγή στις αρχές της επιστήμης των Η/Υ. Περιφερειακή Διεύθυνση Πρωτοβάθμιας και Δευτεροβάθμιας Εκπαίδευσης Βορείου Αιγαίου. Μυτιλήνη, 2014

Εισαγωγή στις αρχές της επιστήμης των Η/Υ

Συγγραφική ομάδα:

Βασίλης Βασιλάκης, Εκπαιδευτικός Πληροφορικής Δ/θμιας εκπ/σης

Τιμολέων Θεοφανέλλης, Σχολικός Σύμβουλος Πληροφορικής Β. Αιγαίου

Αντώνης Νείρος, Εκπαιδευτικός Πληροφορικής Δ/θμιας εκπ/σης

Παναγιώτης Χατζηλάμπρου, Εκπαιδευτικός Πληροφορικής Δ/θμιας εκπ/σης

Γιώργος Χατζηνικολάκης, Εκπαιδευτικός Πληροφορικής Δ/θμιας εκπ/σης

Ευριπίδης Χατζηπαρασκευάς, Εκπαιδευτικός Πληροφορικής Δ/θμιας εκπ/σης

Φιλολογική επιμέλεια: Μαρία Ζαφειρίου, φιλόλογος Δ/θμιας εκπ/σης

Εισαγωγικό Σημείωμα

Όταν ενημερώθηκα για την προκήρυξη για την συγγραφή του σχολικού εγχειριδίου Πληροφορικής της Β' Λυκείου προβληματίστηκα ως προς τη δυσκολία του όλου εγχειρήματος. Πιστεύοντας όμως στη δυναμική μιας ομάδας έναντι της ατομικής δουλειάς και έχοντας ιδία εμπειρία των θετικών αποτελεσμάτων της πρώτης ανέλαβα την πρόκληση. Στάλθηκε λοιπόν ανοιχτή πρόσκληση για συμμετοχή σε όλους τους εκπαιδευτικούς πληροφορικής του Βορείου Αιγαίου. Ανταποκρίθηκαν πέντε, οι οποίοι αποτέλεσαν μαζί με μένα τη συγγραφική ομάδα. Πραγματοποιήθηκαν πολλές διαδικτυακές συναντήσεις, εκφράστηκαν πολλοί προβληματισμοί με την πίεση να είμαστε μέσα στους στόχους του προγράμματος σπουδών και στην απλή και κατανοητή παρουσίαση της ύλης του βιβλίου.

Στο μυαλό μας είχαμε την υλοποίηση του προγράμματος σπουδών με ένα βιβλίο που θα απευθύνεται σε όλους τους μαθητές του ΓΕΛ και του ΕΠΑΛ και θα μπορούν μόνοι τους να το κατανοήσουν χωρίς να χρειάζονται επεξηγήσεις από το διδάσκοντα. Στην προσπάθεια αυτή δεν έγιναν εκπτώσεις όσον αφορά την επιστημονικότητα των διδαχθέντων ενοτήτων ή εννοιών.

Το βιβλίο αυτό το προσφέρουμε ως βοήθημα και είμαστε πρόθυμοι να ακούσουμε και να λάβουμε υπόψη τα σχόλια των εκπαιδευτικών που θα το αξιοποιήσουν για τη βελτίωση του σε επόμενη έκδοση. Παρακαλούμε για τις παρατηρήσεις σας στο pe1920bgel@gmail.com.

Εκ μέρους της συγγραφικής ομάδας,

Τιμολέων Θεοφανέλλης

Σχολικός Σύμβουλος Πληροφορικής Β. Αιγαίου & Δωδεκανήσου

Πίνακας Περιεχομένων

ΕΝΟΤΗΤΑ 1: ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

1.1. ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ 8

1.2. ΘΕΩΡΗΤΙΚΗ & ΕΦΑΡΜΟΣΜΕΝΗ ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ..... 9

ΕΝΟΤΗΤΑ 2: ΘΕΜΑΤΑ ΘΕΩΡΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

2.1. ΠΡΟΒΛΗΜΑ 12

2.1.1. Η έννοια του προβλήματος 12

2.1.2. Κατηγορίες προβλημάτων 12

2.1.3. Υπολογιστικά προβλήματα 13

2.1.4. Διαδικασίες επίλυσης προβλήματος 13

2.2. ΑΛΓΟΡΙΘΜΟΙ 15

2.2.1. Ορισμός Αλγορίθμου 15

2.2.2. Χαρακτηριστικά αλγορίθμου 16

2.2.3. Ανάλυση Αλγορίθμων 16

2.2.4. Βασικοί τύποι αλγορίθμων 19

2.2.5. Τρόποι αναπαράστασης αλγορίθμων 22

2.2.6. Δεδομένα και η αναπαράστασή τους 26

2.2.7. Εντολές και δομές αλγορίθμου 37

2.2.8. Βασικές λειτουργίες δομών δεδομένων 48

2.2.9. Εκσφαλμάτωση λογικών λαθών 56

2.2.10. Τεκμηρίωση Αλγορίθμου 60

2.3. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ 61

2.3.1. Γλώσσες προγραμματισμού και “Προγραμματιστικά Υποδείγματα” 61

2.3.2. Σχεδίαση και συγγραφή κώδικα 63

2.3.3. Ο κύκλος ζωής εφαρμογής λογισμικού 67

ΕΝΟΤΗΤΑ 3: ΘΕΜΑΤΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

3.1. ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ..... 73

3.2. ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ 76

3.3. ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ 81

3.4. ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ 87

ΠΑΡΑΡΤΗΜΑ

Ελληνική βιβλιογραφία..... 89

Ξένη βιβλιογραφία..... 90

Διευθύνσεις διαδικτύου 91

Λεξικό Όρων..... 93

1. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Στόχοι:

Οι μαθητές θα πρέπει:

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

Όροι:

Θεωρητική επιστήμη υπολογιστών, εφαρμοσμένη επιστήμη υπολογιστών.

1.1. Επιστήμη Υπολογιστών

Η Επιστήμη των Υπολογιστών βρίσκεται σήμερα στο επίκεντρο της παγκόσμιας οικονομίας και επηρεάζει σε μεγάλο βαθμό τον τρόπο ζωής των πολιτών. Μάλιστα, οι έννοιες που σχετίζονται με την Επιστήμη των Υπολογιστών έχουν τέτοιο βάθος και εύρος, που βρίσκουν αναρίθμητες εφαρμογές, τόσο στην καθημερινότητα, όσο και σε άλλες επιστήμες (αστρονομία και διαστημική, φυσική, χημεία, βιολογία και γενετική, κοινωνιολογία και ανθρωπολογία, αυτοκινητοβιομηχανία, επικοινωνία, κ.α.).

Η Επιστήμη των Υπολογιστών (Computer Science) και ο όρος Πληροφορική (Informatics) συχνά θεωρούνται ταυτόσημοι, αν και η επιστήμη υπολογιστών, με την πιο στενή της έννοια, εστιάζει στη θεωρητική πληροφορική και τα συναφή μαθηματικά της θεμέλια.

Η Επιστήμη των Υπολογιστών δεν συσχετίζεται μόνο με την κατασκευή υπολογιστών ή την ανάπτυξη λογισμικού, όπως ακριβώς η αστρονομία δεν ασχολείται μόνο με την κατασκευή τηλεσκοπίων και η βιολογία με μικροσκόπια. Οι υπολογιστές αποτελούν το βασικό εργαλείο υπολογισμών, με μεγαλύτερη έμφαση στις διαδικασίες και στις μεθόδους που χρησιμοποιούνται για την επίλυση των προβλημάτων, στο πως αξιοποιείται το κάθε εργαλείο και τι προκύπτει στα αποτελέσματα της χρήσης του. Μάλιστα, η επίλυση πολλών προβλημάτων με τα οποία ασχολείται η Επιστήμη των Υπολογιστών δεν απαιτεί καν τη χρήση Η/Υ, παρά μόνο μολύβι και χαρτί.

Έτσι, η επιστήμη των υπολογιστών εκτείνεται και καλύπτει ένα ευρύ φάσμα θεμάτων, από τη θεωρητική μελέτη των αλγορίθμων και των ορίων των υπολογισμών, μέχρι πρακτικά ζητήματα ανάπτυξης υπολογιστικών συστημάτων σε επίπεδο υλικού και λογισμικού. Οι τέσσερις θεμελιώδεις τομείς της είναι:

- Θεωρία υπολογισμών.
- Αλγόριθμοι και δομές δεδομένων.
- Μεθοδολογία προγραμματισμού και γλώσσες προγραμματισμού.
- Στοιχεία και αρχιτεκτονική υπολογιστών.

1.2. Θεωρητική & Εφαρμοσμένη Επιστήμη Υπολογιστών

Ένας τρόπος ομαδοποίησης του μεγάλου πλήθους πεδίων που συνιστούν την Επιστήμη Υπολογιστών είναι ο διαχωρισμός της σε δυο υποσύνολα, τα οποία και αλληλοσυμπληρώνονται:

α) **Θεωρητική Επιστήμη Υπολογιστών** (Theoritical Computer Science)

Το πεδίο της θεωρητικής Επιστήμης των Υπολογιστών περιλαμβάνει τόσο την κλασική θεωρία υπολογισμών, όσο και ένα ευρύ φάσμα άλλων θεμάτων που επικεντρώνονται στις πιο αφηρημένες, λογικές και μαθηματικές πτυχές της Πληροφορικής. Ενδεικτικά, ορισμένα θεωρητικά πεδία είναι τα ακόλουθα:

- Αλγόριθμοι
- Συμπίεση δεδομένων
- Κρυπτογραφία
- Διόρθωση και ανίχνευση σφαλμάτων.

β) **Εφαρμοσμένη Επιστήμη Υπολογιστών** (Applied Computer Science).

Το πεδίο της εφαρμοσμένης Επιστήμης των Υπολογιστών εστιάζει στην αξιοποίηση συγκεκριμένων εννοιών πληροφορικής για την άμεση και πρακτική επίλυση προβλημάτων του πραγματικού κόσμου. Ορισμένα χαρακτηριστικά πεδία εφαρμογής είναι τα ακόλουθα:

- Τεχνητή νοημοσύνη (λήψη αποφάσεων, ρομποτική, κ.α.)
- Γραφικά (επεξεργασία εικόνας, ειδικά εφέ, βιντεοπαιχνίδια, κ.α.)
- Δίκτυα υπολογιστών και επικοινωνίας
- Βάσεις Δεδομένων.

Προκειμένου να γίνει κατανοητός ο διαχωρισμός της Επιστήμης των Υπολογιστών στα δυο παραπάνω υποσύνολα, καθώς και η μεταξύ τους σύνδεση, ας αναφέρουμε ένα χαρακτηριστικό παράδειγμα:

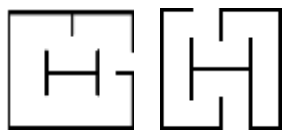
Εικόνα 1.2
Το μέγεθος κάθε προσώπου (*σημαντικότητα ιστοσελίδας*) είναι ανάλογο με το συνολικό μέγεθος των άλλων προσώπων που δείχνουν προς αυτό.

Η επιτυχία της μηχανής αναζήτησης της Google οφείλεται σε μεγάλο βαθμό στον τρόπο αξιολόγησης και κατάταξης των ιστοσελίδων που επιστρέφονται, όταν κάποιος κάνει μια αναζήτηση. Οι ιδρυτές της εταιρείας ανέπτυξαν έναν αλγόριθμο με την ονομασία PageRank, ο οποίος "βαθμολογεί" κάθε ιστοσελίδα, μετρώντας τον αριθμό και την ποιότητα των υπερσυνδέσμων προς αυτή για να εκτιμήσει πόσο "σημαντική" είναι. Στο θεωρητικό μέρος, για την ανάπτυξη του αλγορίθμου χρησιμοποιήθηκαν μαθηματικά, όπως θεωρία γράφων και πιθανότητες, ενώ στο εφαρμοσμένο μέρος, τα παραπάνω προσαρμόστηκαν και υλοποιήθηκαν σε κάποια γλώσσα προγραμματισμού για την αυτοματοποιημένη κατάταξη του τεράστιου όγκου ιστοσελίδων του Παγκόσμιου Ιστού.

Ερωτήσεις – Δραστηριότητες

1. Ένας συμμαθητής σας χειρίζεται με άνεση τον προσωπικό του υπολογιστή και ιδιαιτέρως τη σουίτα εφαρμογών γραφείου που διαθέτει, οπότε ισχυρίζεται ότι είναι πολύ καλός στην πληροφορική. Συμφωνείτε με την άποψή του; Αιτιολογήστε την απάντησή σας.

2. Το 1972, λέγεται ότι ένας δωδεκάχρονος τότε μαθητής, ο John Pledge εφηύρε μια σειρά από βήματα, τα οποία αν ακολουθήσουμε αυστηρά, μπορούμε να βγούμε από οποιονδήποτε λαβύρινθο με ορθές γωνίες. Αυτή η σειρά οδηγιών ονομάστηκε "Αλγόριθμος του Pledge".

Προκειμένου να μην παγιδευτούμε σε κύκλο, χρειαζόμαστε έναν μετρητή για να θυμόμαστε πόσες φορές έχουμε στρίψει. Συγκεκριμένα, για κάθε αριστερή στροφή προσθέτουμε 1, ενώ για κάθε δεξιά στροφή αφαιρούμε 1 από το μετρητή.	**Βήματα**: 1. Προχώρα ευθεία μέχρι να πέσεις σε τοίχο. 2. Μόλις πέσεις σε τοίχο, στρίψε δεξιά. 3. Ακολούθα τον τοίχο κατά μήκος, στρίβοντας αριστερά ή δεξιά και τροποποιώντας κατάλληλα την τιμή του μετρητή. 4. Επανάλαβε το βήμα γ, μέχρι ο μετρητής να γίνει μηδέν ή να φτάσεις στην έξοδο του λαβυρίνθου. 5. Εαν δεν έχεις φτάσει στην έξοδο, επανάλαβε τη διαδικασία από την αρχή (βήμα α).
Δοκιμάστε να εφαρμόσετε τον αλγόριθμο ξεκινώντας από τυχαίες θέσεις μέσα στους διπλανούς λαβύρινθους. Αναφέρετε πρακτικές εφαρμογές, όπου θα μπορούσε να αξιοποιηθεί.	

Χρήσιμες ιστοσελίδες

- ✓ ACM Computing Classification System: http://www.acm.org/about/class
- ✓ Εισαγωγή στην Επιστήμη των Υπολογιστών από το ΕΑΠ: http://www.cs.ucy.ac.cy/courses/EPL003/e-book.pdf

2. ΘΕΜΑΤΑ ΘΕΩΡΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Στο κεφάλαιο αυτό εξετάζεται η έννοια του προβλήματος και του αλγορίθμου, ώστε ο αναγνώστης να προσεγγίσει την λογική του προγραμματισμού.

Στόχοι:

Οι μαθητές μετά την ολοκλήρωση του κεφαλαίου:

- ✓ να μπορούν να κατατάσσουν ένα πρόβλημα στην κατηγορία που ανήκει.
- ✓ να μπορούν να διακρίνουν την ύπαρξη υπολογιστικών και μη προβλημάτων.
- ✓ να περιγράφουν τις φάσεις επίλυσης ενός υπολογιστικού προβλήματος.
- ✓ να περιγράφουν την έννοια του αλγορίθμου και την ύπαρξη συγκεκριμένων χαρακτηριστικών του.
- ✓ να αναφέρουν και να συσχετίζουν τις συγκεκριμένες έννοιες.
- ✓ να αιτιολογούν την ύπαρξη διαφόρων τύπων αλγορίθμων και να επεξηγούν τον τρόπο λειτουργίας τους.
- ✓ να αναγνωρίσουν τις διάφορες μορφές αναπαράστασης του αλγορίθμου και να μπορούν να επιλέγουν την πλέον κατάλληλη για συγκεκριμένο πρόβλημα.
- ✓ να αναφέρουν τους βασικούς τύπους και δομές δεδομένων που χρησιμοποιούνται σε αλγορίθμους.
- ✓ να διακρίνουν τις βασικές εντολές και δομές που χρησιμοποιούνται σε έναν αλγόριθμο.
- ✓ να προσδιορίζουν τον τρόπο λειτουργίας των δομών δεδομένων.
- ✓ να εντοπίζουν και να διορθώνουν τα λογικά λάθη ενός αλγορίθμου.
- ✓ να εξηγούν την ανάγκη δημιουργίας της κατάλληλης τεκμηρίωσης.
- ✓ να δημιουργήσουν ευκρινές γνωσιακό και οργανωμένο νοητικό σχήμα που να περιλαμβάνει τα είδη και τεχνικές προγραμματισμού, με βάση την εμπειρία προγραμματισμού από το Δημοτικό, το Γυμνάσιο και (πιθανόν) την Α΄ τάξη ΓΕΛ.
- ✓ να συνδυάζουν αλγοριθμικές δομές και δεδομένα/δομές δεδομένων για να δημιουργήσουν κώδικα/πρόγραμμα.
- ✓ να διαπιστώσουν ότι οι σημερινές εφαρμογές είναι αρκετά πολύπλοκες και η δημιουργία τους ακολουθεί συγκεκριμένα μοντέλα ανάπτυξης εφαρμογών λογισμικού που εξελίσσονται σε συγκεκριμένες φάσεις.

Όροι:

Αλγόριθμος, Τεκμηρίωση, Αναδρομή, Ψευδοκώδικας, Γλώσσα προγραμματισμού, Διάγραμμα ροής, Εκσφαλμάτωση, Λογισμικό Ανοικτού Κώδικα

2.1. Πρόβλημα

2.1.1. Η έννοια του προβλήματος

Στην καθημερινή μας ζωή καλούμαστε συνεχώς να αντιμετωπίσουμε ποικίλα προβλήματα, όπως προσωπικά, οικογενειακά, ή οικονομικά κ.α.. Επίσης τα προβλήματα μπορεί να είναι από πολύ ειδικά, έως πολύ γενικά. Για παράδειγμα πρόβλημα μπορεί να είναι η αντιμετώπιση της παχυσαρκίας για κάποιον, όπως επίσης πρόβλημα είναι η αντιμετώπιση των ναρκωτικών για ολόκληρη την κοινωνία.

*Ως **πρόβλημα** ορίζεται μία δύσκολη, περίπλοκη κατάσταση που πρέπει να αντιμετωπιστεί και επιζητεί λύση.*

2.1.2. Κατηγορίες προβλημάτων

Προβλήματα λοιπόν συναντώνται σε όλες τους τομείς της ατομικής ή προσωπικής και κοινωνικής ζωής. Προφανώς αντιμετώπιση ενός προβλήματος δεν συνεπάγεται και λύση πάντα. Σε μια προσπάθεια να ομαδοποιήσουμε τα προβλήματα, με βάση την δυνατότητα επίλυσής τους, διακρίνουμε τις εξής κατηγορίες:

- **Επιλύσιμα**. Είναι τα προβλήματα για τα οποία έχει βρεθεί η λύση τους ή εκείνα όπου η λύση δεν έχει βρεθεί ακόμα, αλλά υπάρχει συγκεκριμένη μέθοδος για την επίλυσή τους. Για παράδειγμα η μετακίνηση του ανθρώπου από μία πόλη σε μια γειτονική σε σύντομο χρονικό διάστημα είναι επιλύσιμο πρόβλημα. Η επίλυση μίας δευτεροβάθμιας εξίσωσης είναι επίσης επιλύσιμο πρόβλημα.
- **Ανοικτά**. Πρόκειται για τα προβλήματα για τα οποία δεν έχει βρεθεί κάποια λύση, αλλά παρόλα αυτά δεν έχει αποδειχτεί η αδυναμία λύσης τους. Τέτοιο πρόβλημα είναι, από τα μαθηματικά, η εικασία του Goldbach (διατυπώθηκε το 1742), που αναφέρει ότι κάθε άρτιος αριθμός ισούται με το άθροισμα δύο πρώτων αριθμών, αλλά ως τέτοιο μπορεί να χαρακτηριστεί και το κυκλοφοριακό πρόβλημα μιας μεγάλης πόλης.
- **Άλυτα**. Είναι τα προβλήματα για τα οποία δεν έχει βρεθεί λύση αλλά παράλληλα έχουμε διαπιστώσει την αδυναμία λύσης τους. Από τα μαθηματικά, τέτοιο πρόβλημα είναι ο τετραγωνισμός του κύκλου με κανόνα και διαβήτη. Το συγκεκριμένο πρόβλημα, μπορεί να λυθεί μόνο κατά προσέγγιση.

Πρώτος ονομάζεται ο αριθμός που διαιρείται μόνο με τον εαυτό του και την μονάδα.

2.1.3. Υπολογιστικά προβλήματα

Καλούνται τα προβλήματα τα οποία μπορούν να επιλυθούν και με υπολογιστή.

Μερικά παραδείγματα υπολογιστικών προβλημάτων είναι τα εξής:

- ο υπολογισμός του εμβαδού ενός τραπεζίου,
- η λύση μίας πρωτοβάθμιας ή δευτεροβάθμιας εξίσωσης,
- η ταξινόμηση αριθμών σε αύξουσα ή φθίνουσα σειρά.

2.1.4. Διαδικασίες επίλυσης προβλήματος

Εισαγωγή στην Υπολογιστική Σκέψη

Οι νοητικές διεργασίες που λαμβάνουν χώρα από τον άνθρωπο κατά τη φάση επίλυσης ενός (υπολογιστικού) προβλήματος συνιστούν σήμερα μια κρίσιμη δεξιότητα, σημαντική για όλους, όχι μόνο για όσους ασχολούνται με την Πληροφορική. Έτσι, η **υπολογιστική σκέψη** (Computational Thinking) περιλαμβάνει ένα σύνολο ικανοτήτων και τεχνικών επίλυσης προβλημάτων που βρίσκουν εφαρμογή παντού.

Οι δεξιότητες που απαιτούνται για την επίλυση ενός προβλήματος είναι:

- Ανάλυση του προβλήματος
- Αναγνώριση προτύπων
- Γενίκευση και αφαίρεση
- Σχεδιασμός αλγορίθμου

Φυσικά, προτού ξεκινήσουμε τη διαδικασία επίλυσης ενός προβλήματος, πρέπει να έχουμε κατανοήσει πλήρως τόσο το ίδιο το πρόβλημα, όσο και το χώρο του προβλήματος, δηλαδή το ιδιαίτερο πλαίσιο όπου εντάσσεται και τις συνθήκες υπό τις οποίες καλούμαστε να το αντιμετωπίσουμε, όπως για παράδειγμα τυχόν περιορισμούς και ιδιαιτερότητες που οφείλουμε να λάβουμε υπόψη.

Ας συνεχίσουμε με ένα παιχνίδι, μέσα από το οποίο θα φανεί η σημασία κάθε φάσης για την επίλυση ενός υπολογιστικού προβλήματος. Πρόκειται για μια απλοποιημένη εκδοχή ενός παλιού κινέζικου παιχνιδιού με την ονομασία “Nim Game”.

Κανόνες του παιχνιδιού

- Στο τραπέζι υπάρχει ένας αριθμός από ξυλάκια (ή οποιοδήποτε άλλο αντικείμενο).
- Καθένας από τους δυο παίκτες, εναλλάξ, παίρνει 1, 2 ή 3 ξυλάκια από το τραπέζι όταν έρχεται η σειρά του.
- Χάνει ο παίκτης που αναγκάζεται να πάρει το τελευταίο ξυλάκι από το τραπέζι.

Κατανόηση και ανάλυση του προβλήματος

Ένα πρόβλημα μπορεί εκ πρώτης όψεως να φαίνεται σύνθετο και δύσκολο, αλλά εφόσον το έχουμε κατανοήσει, είμαστε σε θέση να διευκολύνουμε την επίλυσή του είτε διασπώντας το σε μικρότερα, απλούστερα υποπροβλήματα τα οποία μπορούμε να διαχειριστούμε ευκολότερα είτε περιορίζοντας την έκτασή του είτε

επιλύοντας την απλούστερη αυτή εκδοχή και σταδιακά επεκτείνοντας τη λύση του, ώστε να ταιριάζει στο αρχικό πρόβλημα.

Στην προκειμένη περίπτωση, βασική προϋπόθεση είναι η σαφής διατύπωση του προβλήματος ώστε να μην αφήνει περιθώρια παρερμηνειών.

- Με βάση τους προαναφερόμενους κανόνες, επιχειρήστε να παίξετε το παιχνίδι σε ζευγάρια, με διαφορετικό αριθμό από ξυλάκια κάθε φορά και βεβαιωθείτε ότι κατανοήσατε τους κανόνες του.
- Μη διστάσετε να κάνετε ερωτήσεις, όποτε δεν είστε σίγουροι για το αν οι κινήσεις σας τηρούν τους κανόνες του παιχνιδιού.
- Υπάρχει ελάχιστος αριθμός για τα ξυλάκια στο τραπέζι, ώστε να μην έχει ξεκάθαρο πλεονέκτημα ο πρώτος ή ο δεύτερος παίκτης;
- Προσπαθήστε να "μειώσετε" την έκταση του προβλήματος, ξεκινώντας από έναν σχετικά μικρό αλλά ικανό αριθμό από ξυλάκια, π.χ. 10.

Αναγνώριση προτύπων

Αφού έχετε παίξει το παιχνίδι σε δυάδες, συζητήστε στην ολομέλεια αν φαίνεται να υπάρχει κάποια στρατηγική που να οδηγεί σε νίκη. Μην προδώσετε ακόμα τη στρατηγική σας, αλλά παίξτε εναντίον εκείνων που θεωρούν ότι έχουν βρει τέτοια στρατηγική, για να δείτε αν πράγματι ισχύει. Προσπαθήστε να σκεφτείτε αν παίζει ρόλο ποιος παίκτης ξεκινάει πρώτος.

Εάν ανακαλύψατε κάποιο σύνολο κανόνων που φαίνεται να σας δίνει πλεονέκτημα, συγχαρητήρια, είστε σε πολύ καλό δρόμο. Ίσως γενικεύοντας το πρότυπό σας να κερδίζετε πάντα ή τουλάχιστον να φέρνετε σε δύσκολη θέση τον αντίπαλό σας.

Γενίκευση και αφαίρεση

Αφού έχετε καταλήξει σε μια διαδικασία που λειτουργεί για τη λύση του προβλήματος των 10, 11, 12, κ.ο.κ. αντικειμένων, "αφαιρέστε" τις λεπτομέρειες και γενικεύστε τους "χρυσούς κανόνες", ώστε να ορίσετε ένα σαφές αλλά πιο γενικό πλαίσιο που οδηγεί στη νίκη για οποιοδήποτε αριθμό από αντικείμενα.

Σχεδίαση αλγορίθμου

Όταν έχετε καταλήξει στην ιδανική στρατηγική, επιχειρήστε να γράψετε σε ένα φύλλο χαρτί, με όποιο τρόπο θεωρείτε καλύτερο (με λόγια, με βήματα, με διάγραμμα ή άλλο) τις ενέργειες που επιλύουν το πρόβλημα, ώστε αν δώσετε το χαρτί σε κάποιον άσχετο με το παιχνίδι, αυτός ακολουθώντας μηχανικά τις οδηγίες σας, να κερδίζει.

Πρότυπη στρατηγική

Κάθε φορά που παίζουμε, φροντίζουμε τα σπίρτα που απομένουν στο τραπέζι να είναι 4*κ+1, π.χ. 17, 13, 9, 5. Με άλλα λόγια, σε όποιον απομένουν 4*κ + 1 σπίρτα όταν έρθει η σειρά του, βρίσκεται σε μειονεκτική θέση.

Παρατηρήσεις

- Αυτός που παίζει πρώτος, έχει τη δυνατότητα να κερδίζει πάντα, ακολουθώντας τη συγκεκριμένη στρατηγική, εφόσον ο αρχικός αριθμός των σπίρτων δεν είναι 4*κ+1.
- Αυτός που παίζει δεύτερος, κερδίζει, ακολουθώντας τη στρατηγική, μόνο όταν τα σπίρτα αρχικά είναι 4*κ+1.
- Βέβαια, εάν ο αντίπαλος δεν γνωρίζει την προαναφερόμενη στρατηγική, έχουμε σημαντικό πλεονέκτημα, ασχέτως της σειράς μας.

Ερωτήσεις – Δραστηριότητες

1. Να προτείνετε ένα πρόβλημα για κάθε μια από τις κατηγορίες - επιλύσιμα, ανοικτά, άλυτα.
2. Εργαζόμενοι ομαδοσυνεργατικά (χωριστείτε σε ομάδες των 3 μαθητών) και ερευνώντας στο διαδίκτυο, προσπαθήστε να διακρίνετε σε ποια κατηγορία (επιλύσιμα, ανοικτά, άλυτα) ανήκει το κάθε ένα πρόβλημα από τα παρακάτω:
   1. Ο διπλασιασμός του κύβου (ή Δήλιο πρόβλημα).
   2. Η εικασία του Riemann.
   3. Η τριχοτόμηση μίας γωνίας.
   4. Η εύρεση του εμβαδού ενός ορθογωνίου τριγώνου.
3. Γιατί το παιχνίδι με τα ξυλάκια της ενότητας 2.1.4 θεωρείται υπολογιστικό πρόβλημα; Πώς αλλάζει η στρατηγική στο παιχνίδι (αν αλλάζει), παραλλάσσοντας τους κανόνες του παιχνιδιού ώστε νικητής να είναι αυτός που παίρνει το τελευταίο/α σπίρτο/α;

Χρήσιμες ιστοσελίδες

- ✓ Πληροφορίες για το Nim Game και δυνατότητα online gaming: http://www.archimedes-lab.org/game_nim/nim.html

2.2. Αλγόριθμοι

2.2.1. Ορισμός Αλγορίθμου

Οποιαδήποτε εργασία θέλουμε να εκτελέσουμε, έχουμε στο μυαλό μας ένα πλάνο βημάτων που θα ακολουθήσουμε, ώστε να τη φέρουμε σε πέρας με επιτυχία. Για παράδειγμα για να ξεκινήσει ένα αυτοκίνητο, ο οδηγός πρέπει να κάνει κάποιες καθορισμένες ενέργειες με συγκεκριμένη σειρά:

1. Να βάλει μπροστά τη μηχανή
2. Να πατήσει τον συμπλέκτη
3. Να βάλει την πρώτη ταχύτητα
4. Να πατήσει τον επιταχυντή (γκάζι) και να αφήσει το συμπλέκτη.

Παρόμοια και ο υπολογιστής για να εκτελέσει μια εργασία πρέπει να λάβει τις κατάλληλες εντολές που θα τον καθοδηγούν στο τι θα κάνει. Στο κλασικό παράδειγμα του υπολογισμού του μέσου όρου τριών βαθμών, ο υπολογιστής πρέπει για τους, επί παραδείγματι, 25 μαθητές της τάξης:

1. να δεχθεί τρεις βαθμούς για τον καθένα,
2. να κάνει τον υπολογισμό του μέσου όρου
3. να εμφανίσει το αποτέλεσμα στην οθόνη

Η σαφής και ακριβής περιγραφή μια συγκεκριμένης σειράς από εντολές με σκοπό την επίλυση ενός προβλήματος σε πεπερασμένο χρόνο ονομάζεται ***Αλγόριθμος.***

Είναι σημαντικό οι εντολές του αλγορίθμου να βρίσκονται σε σωστή σειρά, να είναι απόλυτα καθορισμένες και να λύνουν το πρόβλημα σε εύλογο χρονικό διάστημα.

2.2.2. Χαρακτηριστικά αλγορίθμου

Κάθε αλγόριθμος έχει ορισμένα χαρακτηριστικά. Πρόκειται για κριτήρια που πρέπει να ικανοποιούνται, ώστε ο αλγόριθμος να επιλύει με επιτυχία το πρόβλημα.

- Είσοδος. Είναι τα δεδομένα που δέχεται ο αλγόριθμος για επεξεργασία. Είναι πιθανό ο αλγόριθμος να μην έχει είσοδο, αλλά τα δεδομένα να τα δημιουργεί ο ίδιος.
- Έξοδος. Κάθε αλγόριθμος πρέπει οπωσδήποτε να έχει τουλάχιστον μια έξοδο, δηλαδή να παράγει κάποιο αποτέλεσμα.
- Περατότητα. Ο αλγόριθμος πρέπει να τερματίζεται μετά από ένα πεπερασμένο αριθμό βημάτων. Αλγόριθμος που πληρεί όλα τα κριτήρια εκτός αυτού της περατότητας είναι υπολογιστική διαδικασία.
- Καθοριστικότητα. Κάθε εντολή του αλγορίθμου πρέπει να είναι ορισμένη με ακρίβεια και σαφήνεια. Ο αλγόριθμος πρέπει να μπορεί ν' αντιμετωπίσει την κάθε περίπτωση που μπορεί να προκύψει. Το κριτήριο της καθοριστικότητας παραβιάζεται συνήθως στις διαιρέσεις, όταν δεν λαμβάνεται υπόψη η περίπτωση να είναι ο διαιρέτης ίσος με μηδέν.
- Αποτελεσματικότητα. Κάθε εντολή του αλγορίθμου πρέπει να είναι τόσο απλή, ώστε αυτός που την εκτελεί, είτε είναι άνθρωπος είτε είναι υπολογιστής να μπορεί να την εκτελέσει μηχανικά σε πεπερασμένο χρόνο, χωρίς να χρειάζεται να χρησιμοποιήσει την ευφυία του. Η αποτελεσματικότητα έχει να κάνει με την εκτελεσιμότητα. Η κάθε εντολή του αλγορίθμου δεν αρκεί να είναι σωστά καθορισμένη, αλλά και εκτελέσιμη. Στην αντίθετη περίπτωση ο αλγόριθμος είναι μη αποτελεσματικός, αφού δεν παράγει αποτέλεσμα.

2.2.3. Ανάλυση Αλγορίθμων

Στην πράξη δεν θέλουμε αλγορίθμους που απλά να λύνουν το πρόβλημα, θέλουμε και αποδοτικούς αλγορίθμους. Σε πολλές περιπτώσεις υπάρχουν πολλοί

αλγόριθμοι για το ίδιο πρόβλημα και πρέπει να αποφασίσουμε ποιος είναι ο καλύτερος.

Η αποδοτικότητα ενός αλγορίθμου εξαρτάται από δύο κυρίως παράγοντες:

- το χρόνο εκτέλεσής του
- τον αποθηκευτικό χώρο που χρησιμοποιεί

Ο χρόνος εκτέλεσης θεωρείται ως ο πιο σημαντικός παράγοντας εκτίμησης της αποδοτικότητας ενός αλγορίθμου. Εδώ πρέπει να τονιστεί ότι ενδιαφέρει η σχετική αποδοτικότητα των αλγορίθμων, δηλαδή πόσο γρηγορότερος είναι ένας αλγόριθμος σε σχέση με κάποιον άλλο και όχι ο απόλυτος χρόνος εκτέλεσής του, που εξαρτάται από πολλούς παράγοντες όπως το υλικό του υπολογιστή και η γλώσσα προγραμματισμού.

Όλα αυτά οδήγησαν στην ανάπτυξη ενός τομέα της Πληροφορικής που ασχολείται με την αξιολόγηση της αποδοτικότητας των αλγορίθμων και ονομάζεται ***Ανάλυση Αλγορίθμων.***

Μία παράμετρος της ανάλυσης ενός αλγορίθμου είναι ο προσδιορισμός της **πολυπλοκότητάς**[1] **του.** Η πολυπλοκότητα ενός αλγορίθμου εκτιμάται με τον υπολογισμό της συνάρτησης f(N), που συσχετίζει το χρόνο εκτέλεσης f, με το πλήθος των δεδομένων εισόδου N. Πιο αποδοτικός θεωρείται ο αλγόριθμος που έχει το μικρότερο χρόνο εκτέλεσης για το ίδιο πλήθος δεδομένων εισόδου.

Ας συγκρίνουμε τις γραφικές παραστάσεις των συναρτήσεων f1[2] (κόκκινη γραμμή) και f2[3] (πράσινη γραμμή) που φαίνονται στο παρακάτω σχήμα, με σκοπό να αποφασίσουμε ποιος αλγόριθμος είναι πιο αποδοτικός, αυτός που αντιστοιχεί στην f1 ή αυτός της f2.

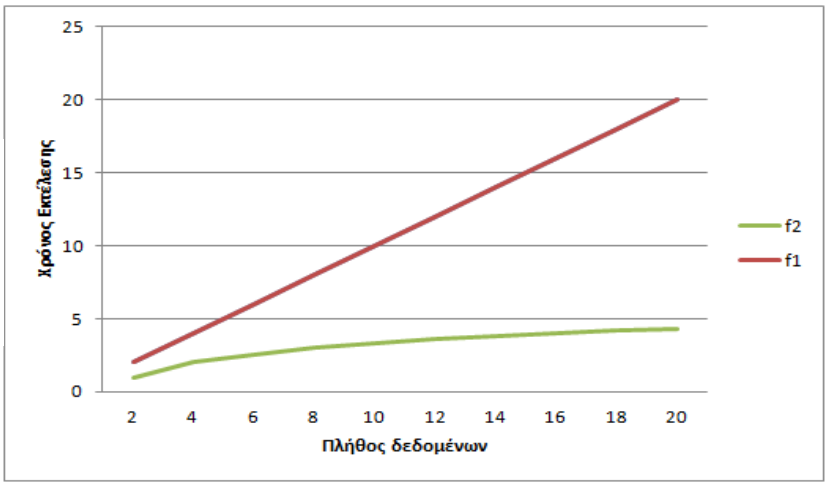
---

[1] Συναντάται και ο όρος υπολογιστική πολυπλοκότητα.

[2] Η συνάρτηση f1 αντιστοιχεί στον αλγόριθμο σειριακής αναζήτησης σε πίνακα που περιγράφεται αναλυτικά στην ενότητα 2.2.8.

[3] Η συνάρτηση f2 αντιστοιχεί στον αλγόριθμο δυαδικής αναζήτησης σε πίνακα που περιγράφεται αναλυτικά επίσης στην ενότητα 2.2.8.

Διαπιστώνουμε ότι η συνάρτηση f1 έχει μεγαλύτερες τιμές από την f2 για κάθε πλήθος δεδομένων εισόδου. Η διαφορά των τιμών αυξάνεται όσο αυξάνει το πλήθος των δεδομένων εισόδου. Είναι προφανές ότι η συνάρτηση f2 είναι πολύ πιο αποδοτική από την f1 ιδιαίτερα αν πρόκειται να διαχειριστούμε μεγάλο πλήθος δεδομένων εισόδου.



Σχήμα 2.2.1. Γραφική Παράσταση Χρόνου Εκτέλεσης – Πλήθος Δεδομένων

Η ανάλυση αλγορίθμων ασχολείται και με τον προσδιορισμό της **υπολογισιμότητας των αλγορίθμων** (computability of algorithms) όπου εξετάζεται η ικανότητα των αλγορίθμων να επιλύουν προβλήματα. Η θεωρία της υπολογισιμότητας αποδεικνύει ότι υπάρχει ένας αριθμός προβλημάτων που δεν μπορούν να επιλυθούν από κάποιον αλγόριθμο. Ο προσδιορισμός αυτών των προβλημάτων είναι αρκετά περίπλοκος και ξεφεύγει από τους σκοπούς του παρόντος βιβλίου.

Ερωτήσεις - Δραστηριότητες

1. Να διατυπώσετε τον ορισμό του αλγορίθμου. Ποια τα χαρακτηριστικά ενός αλγορίθμου;
2. Αναφέρετε τους δύο παράγοντες που καθορίζουν την αποδοτικότητα ενός αλγορίθμου. Ποιος από τους δύο χρησιμοποιείται περισσότερο;
3. Δίνεται ο αλγόριθμος εύρεσης του Μέγιστου Κοινού Διαιρέτη δύο θετικών αριθμών.(Αλγόριθμος του Ευκλείδη)

   Δίνονται δύο θετικοί αριθμοί, ο α και ο β.

   Διαίρεσε τον α με το β και έστω υ το υπόλοιπο της διαίρεσης.

   Αν το υ=0 τότε Μ.Κ.Δ.=β.

   Αν το υ ≠ 0 τότε θέσε α=β και β=υ

   Πήγαινε στο βήμα 2 και συνέχισε την εκτέλεση από εκεί.

   - Για τον παραπάνω αλγόριθμο αναγνωρίστε την είσοδο και την έξοδο του.

- Ικανοποιείται το κριτήριο της περατότητας; Πότε τερματίζεται ο αλγόριθμος;

- Για να ελέγξετε αν πληρούνται τα κριτήρια της καθοριστικότητας και της αποτελεσματικότητας εκτελέστε τον αλγόριθμο και βρείτε το ΜΚΔ των αριθμών α=120 και β=300.
- Πόσες φορές εκτελείται η δεύτερη εντολή για τις παραπάνω τιμές των α και β;

4. Προσπαθήστε να υπολογίσετε το ΜΚΔ με την μέθοδο που μάθατε στο Γυμνάσιο. Κάθε αριθμός με διαδοχικές διαιρέσεις αναλύεται σε γινόμενο πρώτων παραγόντων. Ο ΜΚΔ ισούται με τους κοινούς παράγοντες των δύο αριθμών στη μικρότερη δύναμη.

πχ. για τους αριθμούς 12 και 20.

$12 = 2^2*3$ και $20 = 2^2*5$ οπότε $ΜΚΔ(12,20) = 2^2 = 4$.

1. Υπολογίστε το ΜΚΔ (120, 300) με ανάλυση πρώτων παραγόντων.
2. Ποιος αλγόριθμος υπολογισμού του ΜΚΔ είναι πιο αποδοτικός, ο αλγόριθμος Ευκλείδη ή αυτός της ανάλυσης σε πρώτους παράγοντες; Δικαιολογήστε την απάντησή σας.

2.2.4. Βασικοί τύποι αλγορίθμων

Οι αλγόριθμοι αποτελούν βασικό στοιχείο της Επιστήμης των Υπολογιστών. Για το λόγο αυτό, έχουν αναπτυχθεί πολλά είδη και κατηγορίες αλγορίθμων. Οι αλγόριθμοι μπορούν να κατηγοριοποιηθούν ως *σειριακής* ή *παράλληλης* επεξεργασίας, ανάλογα με τον τρόπο εκτέλεσης των εντολών του αλγορίθμου, είτε ως επαναληπτικοί ή αναδρομικοί. Παρακάτω, περιγράφουμε αναλυτικά τον τρόπο λειτουργίας των βασικών τύπων αλγορίθμων.

Σειριακή και Παράλληλη Επεξεργασία

Η ακολουθία αποτελεί μια από τις θεμελιώδεις πλευρές της ανθρώπινης και φυσικής δραστηριότητας. Πολλά πράγματα στην καθημερινότητά μας ακολουθούν μια σειρά. Για παράδειγμα για να φτιάξουμε ένα φαγητό ακολουθούμε μια σειρά βημάτων. Η ανθρώπινη ομιλία επίσης είναι σειριακή και μεταδίδεται με μία σειρά από λέξεις προφορικά ή γραπτά. Αυτός ίσως να είναι ο λόγος που οι πρώτοι αλγόριθμοι και τα πρώτα προγράμματα υλοποιήθηκαν με βάση την έννοια της ακολουθίας. Με τον τρόπο αυτό το πρότυπο ενός αλγορίθμου καθοριζόταν ως μία πεπερασμένη ακολουθία-σειρά πράξεων.

Ωστόσο οι ανθρώπινες και φυσικές δραστηριότητες δεν είναι μόνο σειριακές αλλά μπορεί να είναι και παράλληλες. Μια δράση δεν αναπτύσσεται μόνο σειριακά αλλά συχνά μπορεί να συμβαίνει ταυτόχρονα σε πολλά διαφορετικά σημεία. Η παραλληλία είναι εξίσου θεμελιώδης και σημαντική όσο και η ακολουθία. Κάθε άνθρωπος που εργάζεται σε ένα οργανισμό δρα σειριακά, ωστόσο ως κομμάτι του οργανισμού που εργάζεται μπορεί να περιέχει πολλούς ανθρώπους που όλοι λει-

τουργούν παράλληλα. Η συντονισμένη δραστηριότητα πολλών μεμονωμένων ανθρώπων που εργάζονται ταυτόχρονα (παράλληλα) ως μία ομάδα δίνει πολύ περισσότερες δυνατότητες στην ομάδα σε σχέση με το άτομο.

Συνεπώς μια πλήρης εικόνα της ανθρώπινης δραστηριότητας αποτελείται από δυο βασικές έννοιες: την **σειριακή** και την **παράλληλη.**

Τα τελευταία χρόνια οι υπολογιστές εξελίχθηκαν και αποτελούνται από πολλούς επεξεργαστές ή αποτελούνται από διπύρηνους, τετραπύρηνους ή οκταπύρηνους επεξεργαστές. Στους υπολογιστές αυτούς, καλό και σκόπιμο είναι, ένας αλγόριθμος να διαχωριστεί σε μικρότερα κομμάτια, που εκτελούνται παράλληλα, αξιοποιώντας διαφορετικούς υπολογιστικούς πόρους. Διαφορετικά, εάν γράψουμε έναν σειριακό αλγόριθμο, θα εργάζεται μόνο ο ένας επεξεργαστής ή ο ένας πυρήνας του επεξεργαστή και οι άλλοι θα κάθονται, δηλαδή θα έχουμε σπατάλη πόρων.

Συμπέρασμα: Στα παραπάνω υπολογιστικά συστήματα, όπου γράφουμε αλγόριθμους παράλληλης επεξεργασίας, ο χρόνος εκτέλεσης ενός αλγορίθμου είναι ασύγκριτα μικρότερος.

Επαναληπτικοί και αναδρομικοί αλγόριθμοι

Ένας άλλος τρόπος κατάταξης των αλγορίθμων είναι η διάκρισή τους σε *αναδρομικούς* και *επαναληπτικούς*.. Δίνουμε τώρα τους ορισμούς των δύο αυτών κατηγοριών.

Ένας ***επαναληπτικός*** *αλγόριθμος (repetition algorithm), εκτελεί επαναληπτικά τις ίδιες εντολές (διαδικασίες).* Δηλαδή, κάνει πολλές φορές το ίδιο πράγμα όπως για παράδειγμα, κάθε μέρα στο σχολείο μετά από κάθε διδακτική ώρα των 40 ή 45 λεπτών ανάλογα, χτυπάει το κουδούνι για διάλειμμα.

Αναδρομικός λέγεται ο αλγόριθμος ο οποίος καλεί (χρησιμοποιεί) τον ίδιο τον εαυτό του. Ο αναδρομικός αλγόριθμος (recursive algorithm) λύνει ένα πρόβλημα λύνοντας ένα ή περισσότερα στιγμιότυπα του ίδιου προβλήματος..

Ένας αναδρομικός αλγόριθμος καλεί (χρησιμοποιεί) τον ίδιο τον εαυτό του. Ο αναδρομικός αλγόριθμος (recursive algorithm) λύνει ένα πρόβλημα λύνοντας ένα ή περισσότερα στιγμιότυπα του ίδιου προβλήματος. Για να υλοποιήσουν αναδρομικούς αλγόριθμους, πολλές γλώσσες προγραμματισμού χρησιμοποιούν αναδρομικές συναρτήσεις (συναρτήσεις οι οποίες καλούν τους εαυτούς τους).

Για την επίλυση ενός προβλήματος, ένας αναδρομικός αλγόριθμος διαιρεί το πρόβλημα σε μικρότερα, επιμέρους προβλήματα, και «εφαρμόζει τον εαυτό του» για τη επίλυση αυτών των προβλημάτων. Τέλος, συνδυάζει τις λύσεις των επιμέρους προβλημάτων για να δημιουργήσει τη λύση του αρχικού προβλήματος.

Η χρήση της αναδρομής διευκολύνει πολύ τον προγραμματιστή στην ανάπτυξη και στον έλεγχο ενός προγράμματος. Θα πρέπει, όμως, να χρησιμοποιείται με μέτρο, γιατί δημιουργούνται άλλα προβλήματα (π.χ. κόστος μνήμης).

Στο παρακάτω παράδειγμα δίνεται ο αλγόριθμος υπολογισμού της ύψωσης σε δύναμη ενός αριθμού. Αρχικά, θα τον δούμε με τον επαναληπτικό τρόπο και στην συνέχεια με τον αναδρομικό.

Επαναληπτικά: Θέλουμε να υπολογίσουμε το 5^3. Αυτό μπορεί να γραφεί ως 5*5*5. Έτσι, με μία επαναληπτική διαδικασία βλέπουμε τον παρακάτω αλγόριθμο:

Βήμα 1: Θέσε i = 1

Βήμα 2: Θέσε P = 1

Βήμα 3: Αν i > 3 τότε πήγαινε στο Βήμα 7 αλλιώς πήγαινε στο Βήμα 4

*Βήμα 4: Θέσε P = P * 5*

Βήμα 5: Θέσε i = i + 1

Βήμα 6: Πήγαινε στο Βήμα 3

Βήμα 7: Εμφάνισε το αποτέλεσμα P

Αναδρομικά: Θέλουμε να υπολογίσουμε το 5^3. Θέτουμε ως i την τιμή 3. Καλούμε την συνάρτηση Δύναμη με παράμετρο τον αριθμό i. Αν αυτός ο αριθμός είναι μηδέν σταματάει η αναδρομή και επιστρέφει το αποτέλεσμα στο προηγούμενο επίπεδο αλλιώς η συνάρτηση καλεί τον εαυτό της ξανά. Ο αναδρομικός αλγόριθμος φαίνεται παρακάτω:

Συνάρτηση ονομάζεται ένα τμήμα υποπρογράμματος, το οποίο μπορεί να δέχεται μία ή περισσότερες τιμές και να υπολογίζει και να επιστρέφει μόνο μία τιμή.

Παράμετρος ονομάζεται μία μεταβλητή, η οποία χρησιμοποιείται για να περνάει τιμές στα υποπρογράμματα.

Βήμα 1: Θέσε i=3

Βήμα 2: Δύναμη(i) !Αναδρομική συνάρτηση

Βήμα 3: Αν i=0 τότε πήγαινε στο Βήμα 4 αλλιώς πήγαινε στο Βήμα 5

Βήμα 4: Θέσε P=1 και επέστρεψε το αποτέλεσμα στο προηγούμενο επίπεδο

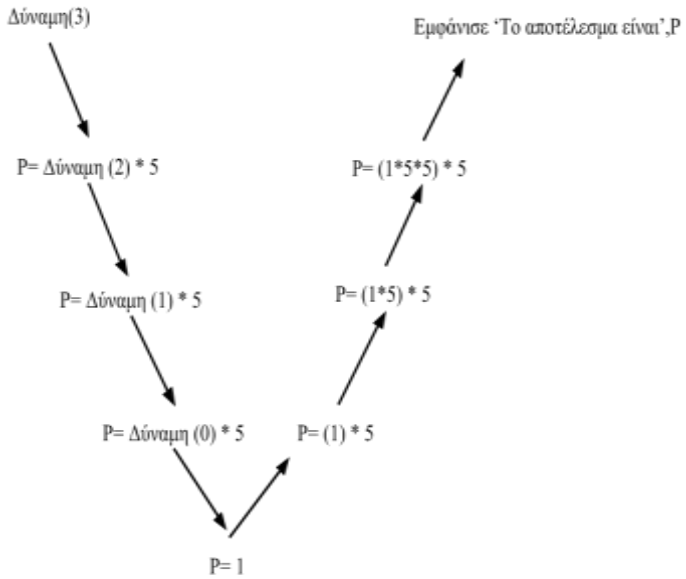
*Βήμα 5: Θέσε P=Δύναμη(i-1) * 5*

Βήμα 6: Πήγαινε στο Βήμα 2 καλώντας τον εαυτό της με παράμετρο το i-1

Βήμα 7: Εμφάνισε το αποτέλεσμα P

Παρακάτω βλέπετε σχηματικά (σχήμα 2.2.2) τον τρόπο λειτουργίας του αναδρομικού αλγορίθμου υπολογισμού της ύψωσης του αριθμού 5 στην δύναμη 3.

Αρχικά ο αλγόριθμος καλεί την συνάρτηση Δύναμη με παράμετρο την τιμή 3. Στην συνέχεια, όπως βλέπουμε στο σχήμα, η συνάρτηση Δύναμη καλεί 3 φορές τον εαυτό της με διαφορετική παράμετρο κάθε φορά. Στην αρχή, με παράμετρο την τιμή 2. Αμέσως μετά, με παράμετρο την τιμή 1 και τέλος, με παράμετρο την τιμή 0.



Σχήμα 2.2.2: Ο τρόπος λειτουργίας ενός αναδρομικού αλγορίθμου υπολογισμού του 5^3

Ερωτήσεις – Δραστηριότητες

1. Διαφορά μεταξύ αναδρομής και επανάληψης
2. Παρακολουθήστε το βίντεο https://www.youtube.com/watch?v=30WcPnvfiKE για αλγόριθμο ταξινόμησης δικτύου και στην συνέχεια υλοποιήστε το στην τάξη.

Χρήσιμες ιστοσελίδες

- ✓ Σημειώσεις για τις Τεχνικές Παράλληλου Προγραμματισμού http://www.it.uom.gr/project/parallel/kef1/1.1.htm

2.2.5. Τρόποι αναπαράστασης αλγορίθμων

Υπάρχουν αρκετοί τρόποι περιγραφής και αναπαράστασης αλγορίθμων. Ενδεικτικά, αναφέρουμε μερικούς τρόπους παρακάτω:

- Φυσική γλώσσα
- Ψευδοκώδικας

- Γλώσσα προγραμματισμού
- Διαγραμματικές τεχνικές

Φυσική γλώσσα

Ένας τρόπος αναπαράστασης αλγορίθμου είναι η *φυσική γλώσσα* κατά βήματα. Δηλαδή, οι εντολές του αλγορίθμου περιγράφονται με βήματα. Σε αυτόν τον τρόπο χρειάζεται ιδιαίτερη προσοχή για να μην παραβιασθεί το κριτήριο της καθοριστικότητας.

Η **μεταβλητή** είναι ένα συμβολικό όνομα κάτω από το οποίο βρίσκεται μια τιμή, η οποία μπορεί να μεταβάλλεται κατά την εκτέλεση του αλγορίθμου.

Θα δείξουμε τώρα ένα απλό παράδειγμα αλγορίθμου, την αντιμετάθεση τιμών δύο μεταβλητών σε φυσική γλώσσα και στη συνέχεια με όλες τις υπόλοιπες αναπαραστάσεις.

Βήμα 1: Διάβασε x,y

Βήμα 2: Θέσε temp = x

Βήμα 3: Θέσε x = y

Βήμα 4: Θέσε y = temp

Λειτουργία του αλγορίθμου: Στο βήμα 1, διαβάζονται οι τιμές δύο μεταβλητών x και y. Στο βήμα 2, θέτουμε σε μία τρίτη προσωρινή μεταβλητή temp την τιμή της μεταβλητής x. Στο βήμα 3, θέτουμε στην μεταβλητή x, την τιμή της μεταβλητής y. Και τέλος, στο βήμα 4, θέτουμε στην μεταβλητή y, την τιμή της προσωρινής μεταβλητής temp (η οποία ισούται με την τιμή της μεταβλητής x). Έτσι πετυχαίνουμε να αντιμεταθέσουμε τις τιμές δύο μεταβλητών x και y.

Ψευδοκώδικας

Ένας δεύτερος τρόπος αναπαράστασης αλγορίθμων είναι ο *ψευδικώδικας*. Αυτός ο τρόπος, χρησιμοποιεί μία απλή γλώσσα γραφής των εντολών του αλγορίθμου που ονομάζεται *ψευδοκώδικας* (ή *ψευδογλώσσα*) και είναι ο τρόπος που χρησιμοποιείται περισσότερο για την αναπαράσταση αλγορίθμων. Σε αυτόν τον τρόπο, οι εντολές γράφονται με κάποιες απλές λέξεις όπως: Αλγόριθμος, Διάβασε, Εμφάνισε, Εκτύπωσε, Αν … τότε … Τέλος_αν κ.λ.π.

Θα δείξουμε τώρα το ίδιο παράδειγμα αντιμετάθεσης των τιμών δύο μεταβλητών x και y, σε μορφή ψευδοκώδικα.

```
Διάβασε x,y
temp ← x
x ← y
y ← temp
```

Βλέπουμε ότι η λειτουργία του αλγορίθμου είναι ακριβώς ίδια με την προηγούμενη, απλά αλλάζει ο τρόπος αναπαράστασης του αλγορίθμου.

Γλώσσα προγραμματισμού

Ένας τρίτος τρόπος αναπαράστασης αλγορίθμων είναι η γλώσσα προγραμματισμού. Δηλαδή, ο αλγόριθμος γράφεται σε μία γλώσσα προγραμματισμού υψηλού επιπέδου που όταν εκτελεστεί θα δώσει ίδια αποτελέσματα με τον αλγόριθμο. Συνήθως, ένας αλγόριθμος γράφεται πρώτα σε ψευδοκώδικα και στην συνέχεια μετατρέπεται σε μία γλώσσα προγραμματισμού υψηλού επιπέδου.

Θα δείξουμε τώρα το ίδιο παράδειγμα αντιμετάθεσης των τιμών δύο μεταβλητών x και y, σε μορφή γλώσσας προγραμματισμού.

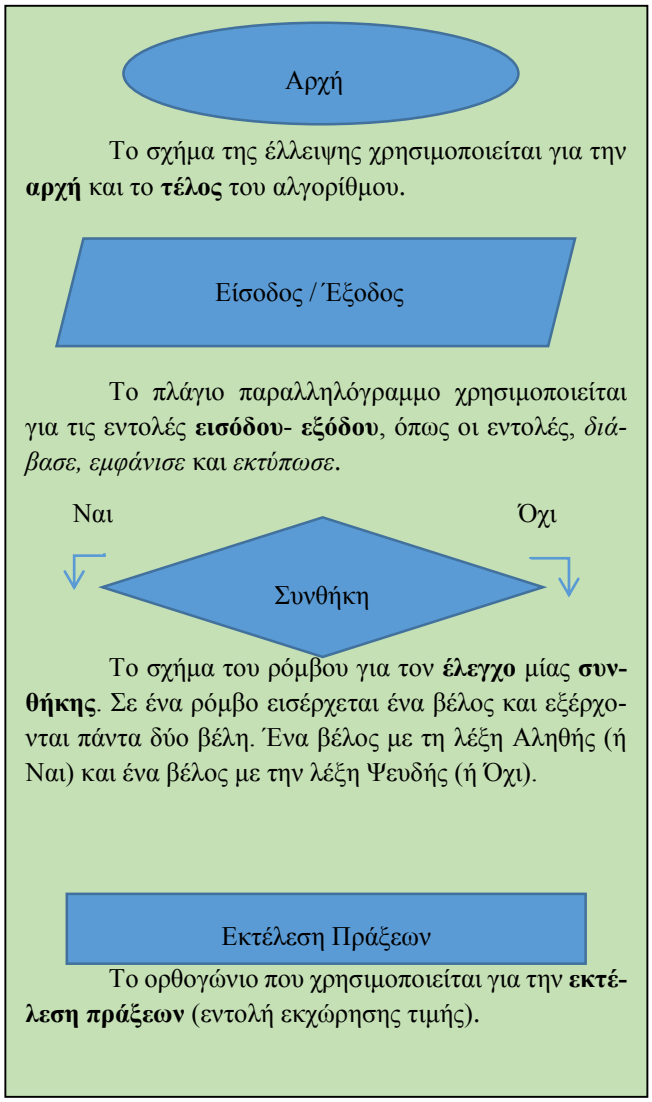
```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: x,y,temp
ΑΡΧΗ
        ΔΙΑΒΑΣΕ x,y
        temp ← x
        x ← y
        y ← temp
ΤΕΛΟΣ ΠΑΡΑΔΕΙΓΜΑ
```

Βλέπουμε ότι η λειτουργία του τμήματος προγράμματος είναι ακριβώς ίδια με την προηγούμενη, απλά αλλάζει ο τρόπος αναπαράστασης.

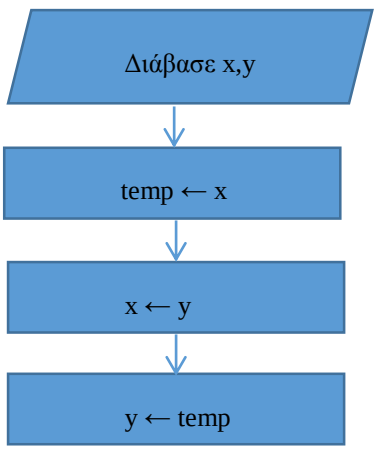
Μεθοδολογίες διαγραμματικής αναπαράστασης αλγορίθμων

Ένας τέταρτος τρόπος αναπαράστασης αλγορίθμων είναι οι διαγραμματικές τεχνικές. Αυτός ο τρόπος χρησιμοποιεί απλά γεωμετρικά σχήματα για να αναπαραστήσει τις εντολές ενός αλγορίθμου, όπως έλλειψη, πλάγιο παραλληλόγραμμο, ορθογώνιο, ρόμβος. Ο πιο γνωστός τρόπος διαγραμματικής αναπαράστασης είναι το *διάγραμμα ροής* (ή *λογικό διάγραμμα*). Χρησιμοποιείται όλο και σπανιότερα στην βιβλιογραφία και στην πράξη, διότι είναι δύσκολο να αποτυπωθεί ένας μεγάλος αλγόριθμος με γεωμετρικά σχήματα.

Το σχήμα της έλλειψης χρησιμοποιείται για την αρχή και το τέλος του αλγορίθμου. Το πλάγιο παραλληλόγραμμο για τις εντολές εισόδου- εξόδου, όπως οι εντολές, *διάβασε*, *εμφάνισε* και *εκτύπωσε*. Το ορθογώνιο που δηλώνει την εκτέλεση πράξεων (εντολή εκχώρησης τιμής). Τέλος, το σχήμα του ρόμβου που δηλώνει τον έλεγχο μίας συνθήκης. Σε ένα ρόμβο εισέρχεται ένα βέλος και εξέρχονται πάντα δύο βέλη. Ένα βέλος με τη λέξη *Αληθής* (ή Ναι) και ένα βέλος με την λέξη *Ψευδής* (ή Όχι).



Στο διπλανό σχήμα παρουσιάζεται το παράδειγμα αντιμετάθεσης των τιμών δύο μεταβλητών x και y, σε μορφή διαγράμματος ροής. Βλέπουμε εδώ ότι η λειτουργία του αλγορίθμου είναι ακριβώς ίδια με τις τρεις προηγούμενες περιπτώσεις, απλά αλλάζει ο τρόπος αναπαράστασης του αλγορίθμου.



Ερωτήσεις – Δραστηριότητες

1. Περιγράψτε τους τρόπους αναπαράστασης αλγορίθμου.
2. Εξηγήστε τη λειτουργία του αλγορίθμου αντιμετάθεσης τιμών δύο μεταβλητών.

Χρήσιμες ιστοσελίδες

- ✓ Σημειώσεις για τις Τεχνικές Παράλληλου Προγραμματισμού: http://www.it.uom.gr/project/parallel/kef1/1.1.htm

2.2.6. Δεδομένα και η αναπαράστασή τους

Εισαγωγή

Η αναπαράσταση της πληροφορίας είναι ζωτικής σημασίας για την Επιστήμη των Υπολογιστών. Βασικός σκοπός ενός υπολογιστικού συστήματος δεν είναι μόνο η εκτέλεση υπολογισμών αλλά η αποθήκευση και η ανάκτηση της πληροφορίας, με τη μεγαλύτερη δυνατή ταχύτητα. Το αντικείμενο των Δομών Δεδομένων και των Αλγορίθμων βρίσκεται στην καρδιά της Επιστήμης των Υπολογιστών και ιδιαίτερα του προγραμματισμού.

Η έννοια της μεταβλητής

Στα περισσότερα προγράμματα που αναπτύσσουμε είναι απαραίτητη η αποθήκευση κάποιων τιμών για να χρησιμοποιηθούν σε περαιτέρω υπολογισμούς. Για το σκοπό αυτό χρησιμοποιούμε **μεταβλητές.**

*Η **μεταβλητή** είναι ένα συμβολικό όνομα κάτω από το οποίο βρίσκεται μια τιμή, η οποία μπορεί να μεταβάλλεται κατά την εκτέλεση του αλγορίθμου.*

Στην πραγματικότητα, η μεταβλητή είναι ένας τρόπος να αποκτήσουμε πρόσβαση στην μνήμη του υπολογιστή για να αποθηκεύσουμε ή να προσπελάσουμε κάποιες τιμές.

Σχήμα 2.2.4 : Μπορούμε να φανταστούμε την μεταβλητή σαν ένα κουτί ικανό να χωρέσει μόνο ένα αντικείμενο. Όταν τοποθετήσουμε ένα νέο αντικείμενο στο κουτί αυτόματα αντικαθιστά το προηγούμενο

Τύποι Δεδομένων

Οι τιμές που δίνουμε σε μεταβλητές χωρίζονται σε 4 βασικούς τύπους :

Ακέραιοι : περιλαμβάνουν όλους τους ακέραιους αριθμούς π.χ. 34, -9, 0

Πραγματικοί : περιλαμβάνουν όλους τους αριθμούς, ακέραιους και δεκαδικούς π.χ. 2.76, 4.0, -9.3

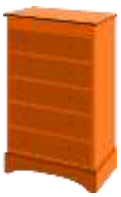
Χαρακτήρες : περιλαμβάνουν έναν ή περισσότερους χαρακτήρες, πχ. "Καλημέρα", "α", "θα πάμε για μπάνιο;"

Λογικοί : περιλαμβάνουν τις τιμές Αληθής και Ψευδής.

Πολλές γλώσσες προγραμματισμού δίνουν τη δυνατότητα στον προγραμματιστή να ορίσει επιπλέον δικούς του τύπους δεδομένων.

Δομές Δεδομένων

Οι Δομές Δεδομένων αφορούν στους διαφορετικούς τρόπους που μπορούμε να επιλέξουμε για την οργάνωση και αποθήκευση των δεδομένων μας.



Σχήμα 2.2.5

Χωρίζονται σε δύο μεγάλες κατηγορίες. Τις στατικές δομές δεδομένων οι οποίες έχουν προκαθορισμένο μέγεθος που δεν μπορεί να μεταβληθεί και τις δυναμικές δομές δεδομένων, το μέγεθος των οποίων μπορεί να μεταβάλλεται κατά την εκτέλεση του αλγορίθμου. Επιπλέον, οι στατικές δομές δεδομένων αποθηκεύονται σε συνεχόμενες θέσεις μνήμης, κάτι που δεν ισχύει για τις δυναμικές δομές.

Μπορούμε να φανταστούμε τις πρώτες σαν μια συρταριέρα (Σχήμα 2.2.5), η οποία περιλαμβάνει ένα συγκεκριμένο αριθμό συρταριών στα οποία μπορούμε να αποθηκεύσουμε αντικείμενα. Το πλήθος των συρταριών δεν μπορεί να μεταβληθεί και αν τελικά χρειαστούμε περισσότερο χώρο αποθήκευσης θα πρέπει να χρησιμοποιήσουμε μια νέα συρταριέρα με περισσότερα συρτάρια.

Σχήμα 2.2.6

Αντίθετα, οι δυναμικές δομές μοιάζουν με τα κουτιά αποθήκευσης (Σχήμα 2.2.6). Μπορούμε ανα πάσα στιγμή να προσθέσουμε ένα καινούριο κουτί αποθήκευσης και να αυξήσουμε τη συνολική χωρητικότητα που έχουμε στη διάθεσή μας ή να αφαιρέσουμε ένα κουτί αποθήκευσης όταν δεν το χρειαζόμαστε πλέον.

Θα παρουσιάσουμε παρακάτω μερικές από τις πιο συχνά χρησιμοποιούμενες δομές δεδομένων καθώς και παραδείγματα χρήσης τους στον προγραμματισμό.

Μονοδιάστατοι Πίνακες

Ο πίνακας είναι μια στατική δομή, στην οποία μπορούμε να αποθηκεύσουμε τιμές του ίδιου τύπου δεδομένων. Ο πίνακας χαρακτηρίζεται από το όνομα που θα του δώσουμε και από την χωρητικότητά του, δηλαδή το πλήθος των τιμών που μπορεί να αποθηκεύσει.

Για να αναφερθούμε σε μια τιμή του πίνακα χρησιμοποιούμε το όνομα που του δώσαμε ακολουθούμενο από έναν αριθμό μέσα σε αγκύλες []. Ο αριθμός αυτός δηλώνει τη θέση της τιμής μέσα στον πίνακα και ονομάζεται δείκτης. Η αρίθμηση των θέσεων του πίνακα ξεκινάει από τον αριθμό 1.

Ο μονοδιάστατος μοιάζει με ένα δρόμο γεμάτο σπίτια. Κάθε σπίτι είναι ένα από τα στοιχεία του πίνακα. Ο αριθμός της διεύθυνσης του σπιτιού είναι ο δείκτης, που προσδιορίζει τη θέση του.

Σχήμα 2.2.7: Ο μονοδιάστατος μοιάζει με ένα δρόμο γεμάτο σπίτια.

Ένα από τα βασικά χαρακτηριστικά ενός πίνακα είναι ότι αποτελεί μια δομή τυχαίας προσπέλασης, δηλαδή ο χρόνος εντοπισμού (προσπέλασης) ενός στοιχείου του είναι ανεξάρτητος από τη θέση του στοιχείου στον πίνακα. Αυτό είναι μια ιδιότητα, που δεν υπάρχει στις περισσότερες δομές δεδομένων.

Για παράδειγμα, μια ιστοσελίδα καταγράφει τον αριθμό των ατόμων που την επισκέπτονται για κάθε ημέρα της εβδομάδας, ώστε να μπορεί να βγάζει στατιστικά στοιχεία σε εβδομαδιαία βάση. Για την αποθήκευση των παραπάνω τιμών μπορεί να χρησιμοποιηθεί ένας μονοδιάστατος πίνακας, χωρητικότητας 7 τιμών, όπως φαίνεται παρακάτω.

1			ΕΠΙΣΚΕΨΕΙΣ			7
1002	785	996	1100	956	1200	1097

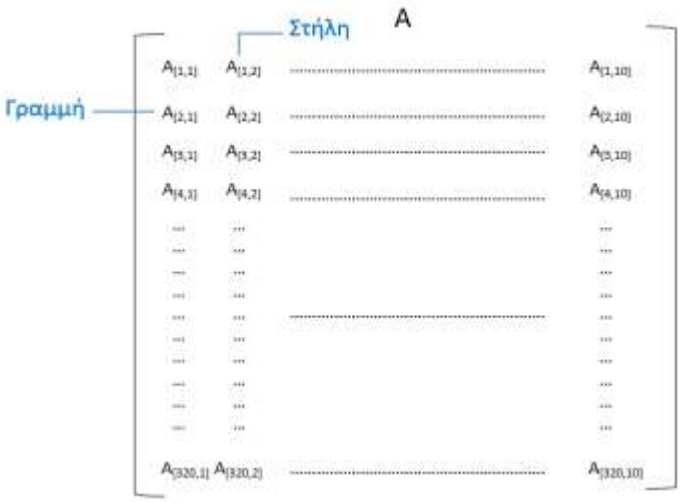
Για να προσπελάσουμε μια από τις τιμές του πίνακα, έστω την πρώτη τιμή, γράφουμε ΕΠΙΣΚΕΨΕΙΣ[1]. Με τον ίδιο τρόπο για να προσπελάσουμε την τελευταία θέση του συγκεκριμένου πίνακα γράφουμε ΕΠΙΣΚΕΨΕΙΣ[7].

Δισδιάστατοι Πίνακες

Πώς θα μπορούσαμε να αναπαραστήσουμε το ταμπλό ενός παιχνιδιού όπως η τρίλιζα ή το σκάκι ή το ωρολόγιο πρόγραμμα ενός σχολείου; Είναι αρκετοί οι μονοδιάστατοι πίνακες για να αποθηκεύσουμε τα δεδομένα μας σε περιπτώσεις όπως οι προηγούμενες;

Ας σκεφτούμε ένα ακόμα παράδειγμα. Έστω ότι ένα σχολείο θέλει να αποθηκεύσει το πλήθος των απουσιών που έκανε κάθε ένας από τους 320 μαθητές του για κάθε μήνα από Σεπτέμβρη μέχρι και Ιούνη. Μια λύση θα ήταν να χρησιμοποιήσει 10 διαφορετικούς μονοδιάστατους πίνακες, έναν για κάθε μήνα. Όμως, μια τέτοια λύση δεν είναι εύχρηστη και εύκολα διαχειρίσιμη.

Σε τέτοιες περιπτώσεις που τα δεδομένα μας μπορούν να οργανωθούν σε γραμμές και στήλες (ανά μαθητή και ανά μήνα στο παράδειγμά μας) χρησιμοποιούμε ένα δισδιάστατο πίνακα.



Ο δισδιάστατος πίνακας είναι ένα σύνολο στοιχείων του ίδιου τύπου, κάθε στοιχείο του οποίου προσδιορίζεται από ένα **ζεύγος** δεικτών.

Με βάση αυτή την απεικόνιση διακρίνουμε σ' ένα δισδιάστατο πίνακα γραμμές, δηλαδή οριζόντιες συλλογές στοιχείων, και στήλες, δηλαδή κατακόρυφες συλλογές στοιχείων.

Ο παραπάνω πίνακας Α έχει 320 γραμμές και 10 στήλες. Κάθε ζεύγος δεικτών προσδιορίζει ένα και μόνο ένα στοιχείο του πίνακα.

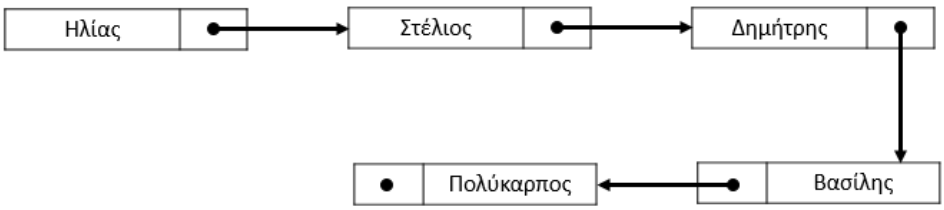
Για να προσπελάσουμε ένα στοιχείο του δισδιάστατου πίνακα χρησιμοποιούμε το όνομα του πίνακα, ακολουθούμενο από δύο αγκύλες [] που μέσα αναγράφουμε τη γραμμή και τη στήλη που βρίσκεται το στοιχείο. Για παράδειγμα A[1,1], A[5,12] κοκ.

Λίστες

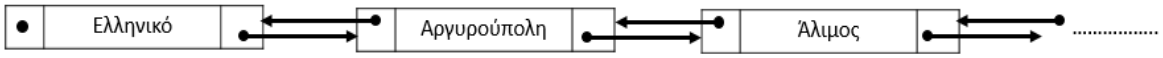
Ας φανταστούμε έναν αγώνα σκυταλοδρομίας. Οι αθλητές που λαμβάνουν μέρος είναι τοποθετημένοι με μια συγκεκριμένη σειρά και ο κάθε αθλητής για να ξεκινήσει να τρέχει πρέπει να περιμένει τον προηγούμενο του. Σε περιπτώσεις όπως αυτή, που κάθε στοιχείο συνδέεται με το αμέσως προηγούμενο και επόμενο στοιχείο, χρησιμοποιούμε τη δομή δεδομένων της **λίστας** για να αποθηκεύσουμε τιμές.

Για την αναπαράσταση μίας συνδεδεμένης λίστας, μαζί με την τιμή κάθε στοιχείου αποθηκεύεται και ένας σύνδεσμος ή δείκτης που «δείχνει» στη θέση του επόμενου (ή και του προηγούμενου) στοιχείου της λίστας.

Το τελευταίο στοιχείο της λίστας δεν περιέχει βέλος, διότι εφόσον είναι ο τελευταίος κόμβος της λίστας δε «δείχνει» πουθενά ή όπως λέμε «δείχνει» στον κενό δείκτη.



Σχήμα 2.2.8: Το παράδειγμα των αθλητών της σκυταλοδρομίας - Απλά συνδεδεμένη λίστα



Σχήμα 2.2.9: Οι σταθμοί του μετρό - Μια διπλά συνδεδεμένη λίστα

Μια λίστα μπορεί να είναι απλά συνδεδεμένη, όπως στο παράδειγμα της σκυταλοδρομίας, δηλαδή να μπορούμε να κινηθούμε προς μία μόνο κατεύθυνσή, ξεκινώντας από το πρώτο στοιχείο και μετακινούμενοι προς το τελευταίο, όπως δείχνουν τα βέλη του σχήματος ή να είναι διπλά συνδεδεμένη, δηλαδή να μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις. Ένα τέτοιο παράδειγμα είναι οι σταθμοί του μετρό, όπως φαίνεται στην παραπάνω εικόνα.

Ίσως εκ πρώτης όψεως κάποιος να συμπεράνει ότι η λίστα δεν διαφέρει από τον πίνακα. Με μια πιο προσεκτική ματιά όμως, θα παρατηρήσει ότι υπάρχουν σημαντικές διαφορές μεταξύ τους.

Η πρώτη είναι ότι ο πίνακας θεωρείται μια δομή τυχαίας προσπέλασης, σε αντίθεση με μια λίστα που είναι στην ουσία μια δομή ακολουθιακής ή σειριακής προσπέλασης. Για να φθάσουμε, δηλαδή, σ' ένα στοιχείο μιας λίστας πρέπει να περάσουμε από όλα τα προηγούμενα ξεκινώντας από το πρώτο. Δεύτερον, το μέγεθος ενός πίνακα παραμένει σταθερό, ενώ το μέγεθος μιας λίστας όχι, διότι συνήθως απαιτούνται διαγραφές παλαιών και εισαγωγές νέων στοιχείων. Δηλαδή, μια λίστα είναι μια δυναμική δομή και όχι στατική, όπως ο πίνακας.

Στοίβα

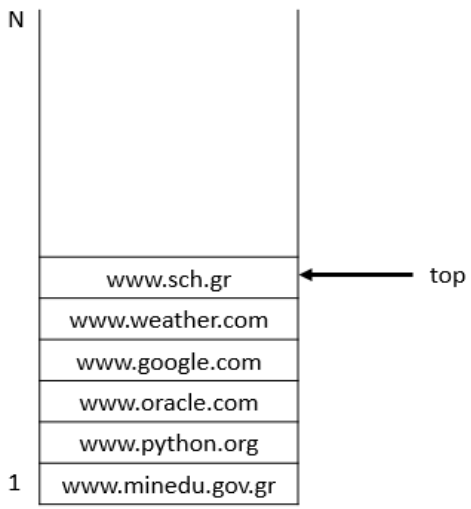
Όταν επισκεφθούμε κάποια σελίδα στο Διαδίκτυο, το πρόγραμμα περιήγησης την αποθηκεύει με μια μορφή ιστορικού, ώστε να μπορούμε να την επισκεφθούμε και στο μέλλον, ακόμα και αν έχουμε ξεχάσει τη διεύθυνσή της.

Ο τρόπος που αποθηκεύονται οι σελίδες στο ιστορικό, είναι με την αντίστροφη χρονολογική σειρά που τις επισκεφθήκαμε. Η τελευταία σελίδα που επισκεφθήκαμε φαίνεται πρώτη στο ιστορικό, η προτελευταία φαίνεται δεύτερη κοκ. Όταν αποθηκευτεί μια νέα σελίδα στο ιστορικό εμφανίζεται στην κορυφή σε σχέση με τις υπόλοιπες. Αυτή η σειρά επεξεργασίας των δεδομένων ονομάζεται Last In First Out (LIFO) και μια δομή δεδομένων που την ακολουθεί είναι η **στοίβα**.

Μπορούμε να φανταστούμε τη στοίβα σαν μια στοίβα από βιβλία ή πιάτα που τοποθετούνται το ένα πάνω στο άλλο. Κάθε νέο στοιχείο που βάζουμε στη στοίβα τοποθετείται στα υπόλοιπα. Αυτή η λειτουργία ονομάζεται **ώθηση**. Με τον ίδιο τρόπο το πρώτο στοιχείο που μπορούμε να αφαιρέσουμε από μια στοίβα είναι αυτό που βρίσκεται στην κορυφή της, λειτουργία που ονομάζεται **απώθηση**.

Επομένως, το μόνο στοιχείο στο οποίο έχουμε πρόσβαση σε μια στοίβα είναι το κορυφαίο της, τη θέση του οποίου συμβολίζουμε με ένα δείκτη που συνήθως ονομάζουμε top (κορυφή).

Στο παρακάτω παράδειγμα φαίνεται μια στοίβα - ιστορικό που περιέχει τις διευθύνσεις 6 ιστοσελίδων. Για να φτάσουμε στο στοιχείο "www.google.com" πρέπει να απωθήσουμε τα δύο στοιχεία που βρίσκονται πάνω από αυτό, δηλαδή το "www.sch.gr" και το "www.weather.com"



Σχήμα 2.2.10: Υλοποίηση στοίβας με πίνακα χωρητικότητας N

Ουρά

Μια ακόμα δομή δεδομένων που χρησιμοποιείται ευρύτατα στην Επιστήμη των Υπολογιστών είναι η δομή της ουράς.

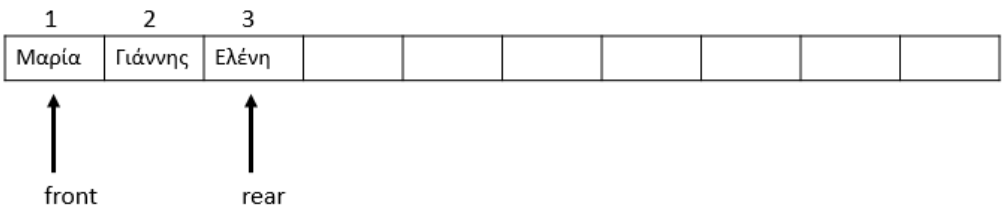
Στην καθημερινότητά μας συναντάμε πολύ συχνά "ουρές", όπως για παράδειγμα στο ταμείο ενός σούπερ μάρκετ, στην τράπεζα, στους σταθμούς των διοδίων και γενικότερα οπουδήποτε δημιουργείται μια ουρά αναμονής.

Δεδομένου ότι κανένας δεν "κλέβει" (!), ένα άτομο που μόλις αφίχθη, θα τοποθετηθεί στο πίσω μέρος της ουράς, ενώ το άτομο που εξυπηρετείται βρίσκεται στο εμπρός μέρος της ουράς, όπως φαίνεται και στο σχήμα 2.2.11.

Επομένως, σε μια ουρά μπορούμε να έχουμε πρόσβαση σε δύο μόνο στοιχεία της. Στο στοιχείο που βρίσκεται στο εμπρός μέρος της ουράς και στο στοιχείο που βρίσκεται στο πίσω μέρος της ουράς. Αντίστοιχα, χρησιμοποιούμε δύο δείκτες για να μας υποδεικνύουν τις παραπάνω θέσεις, οι οποίοι ονομάζονται συνήθως front και rear.

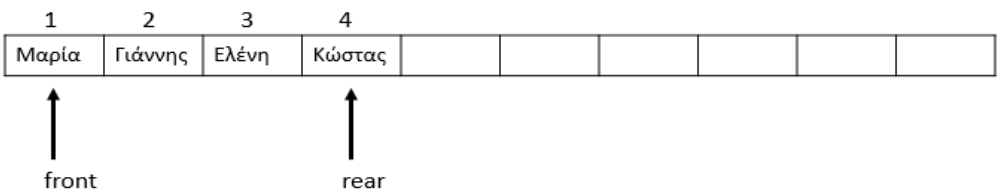
Ας εξετάσουμε το παράδειγμα μιας τράπεζας, που προκειμένου να εξυπηρετεί τους πελάτες της με τη σειρά που έρχονται, έχει υιοθετήσει το σύστημα με τα χαρτάκια που αναγράφουν αριθμούς προτεραιότητας. Για λόγους ευκολίας θα υποθέσουμε ότι η τράπεζα έχει μόνο 1 ταμείο.

Ο πρώτος πελάτης που μπαίνει στην τράπεζα παίρνει ένα χαρτάκι με αριθμό 1, ο δεύτερος με αριθμό 2, κοκ. Ας υποθέσουμε ότι στην τράπεζα έχουν μπει οι 3 πρώτοι πελάτες, όπως φαίνεται παρακάτω:



Σχήμα 2.2.11: Λειτουργία ουράς (α)

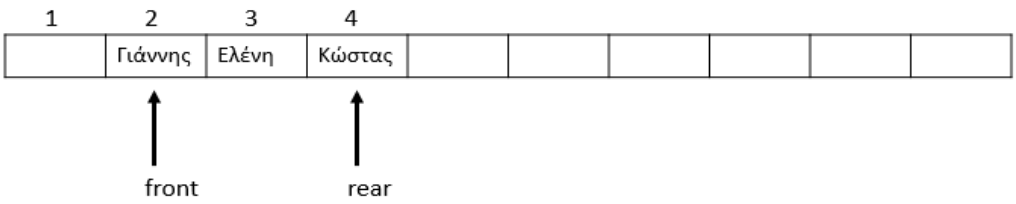
Οι δείκτες front και rear έχουν τιμές 1 και 3 αντίστοιχα. Αν έρθει ένας ακόμα πελάτης θα πάρει το χαρτάκι με αριθμό 4 και θα σταθεί στο τέλος της ουράς. Η εισαγωγή του νέου στοιχείου έχει ως αποτέλεσμα την μεταβολή του δείκτη rear (σχήμα 2.2.12).



Σχήμα 2.2.12: Λειτουργία ουράς (β)

Όταν ο πρώτος πελάτης εξυπηρετηθεί τότε θα φύγει από την ουρά και θα εξυπηρετηθεί ο επόμενος, δηλαδή ο δεύτερος. Η παραπάνω λειτουργία ονομάζεται εξαγωγή και έχει ώς αποτέλεσμα την μεταβολή του δείκτη front (σχήμα 2.2.13).

Παρατηρούμε λοιπόν, ότι σε μια ουρά το πρώτο στοιχείο που έρχεται είναι και το πρώτο που φεύγει. Αυτή η σειρά επεξεργασίας δεδομένων ονομάζεται First In First Out (FIFO).



Σχήμα 2.2.13: Λειτουργία ουράς (γ)

Δέντρα

Σε πολλά προβλήματα χρειάζεται να αναπαραστήσουμε ιεραρχικές δομές. Για παράδειγμα, η αναπαράσταση του οργανογράμματος μιας επιχείρησης, οι πληροφορίες για τη γενεαλογία μας, η δομή του συστήματος αρχείων και φακέλων στον υπολογιστή μας ακόμα και ο τρόπος με τον οποίο οργανώνονται τα περιεχόμενα αυτού του βιβλίου. Σε τέτοιες περιπτώσεις χρειαζόμαστε μια δομή δεδομένων που ονομάζεται δέντρο.

Στο παρακάτω σχήμα φαίνονται οι συνομοταξίες του βασιλείου των ζώων σε δενδρική μορφή.

Σχήμα 2.2.14: Δέντρο

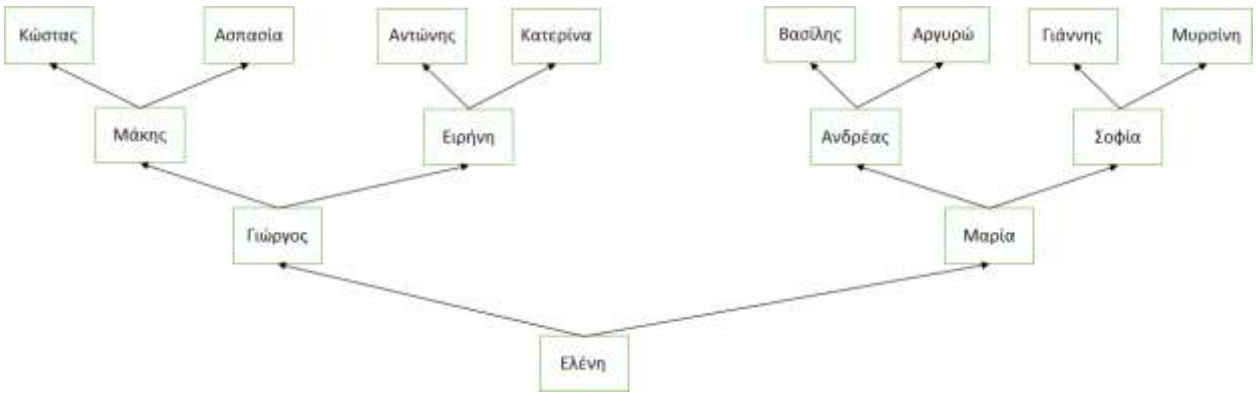
Κάθε δέντρο αποτελείται από κόμβους και ακμές. Οι ακμές είναι ευθύγραμμα τμήματα που συνδέουν τους κόμβους μεταξύ τους. Σε κάθε δέντρο υπάρχει ένας και μοναδικός κόμβος, που ονομάζεται ρίζα του δέντρου. Από αυτήν, μόνο ξεκινούν ακμές, ενώ δεν υπάρχουν ακμές που να καταλήγουν σε αυτή.

Ρίζα : ο ανώτερος κόμβος του δέντρου
Γονέας : κόμβος με τουλάχιστον 1 παιδί.
Αδέλφια : κόμβοι με τον ίδιο γονέα
Φύλλα : Κόμβοι χωρίς παιδιά
Διαδρομή : μια ακολουθία κόμβων που αποτελεί το μονοπάτι από έναν κόμβο σε έναν άλλο κόμβο
Μήκος διαδρομής : το πλήθος των ακμών που περιλαμβάνονται σε αυτή
Επίπεδο ή βάθος κόμβου : το μήκος της διαδρομής από τη ρίζα σε έναν κόμβο
Ύψος δέντρου : το μέγιστο βάθος των τερματικών του κόμβων
Βαθμός κόμβου : ο αριθμός των παιδιών ενός κόμβου

Στο προηγούμενο παράδειγμα ρίζα του δέντρου είναι τα Ζώα, ενώ τα Αμφίβια, Πτηνά, Ερπετά και Θηλαστικά είναι τα φύλλα του. Το ύψος του δέντρου είναι 3.

Μια ειδική περίπτωση δέντρου με πολλές εφαρμογές στην Επιστήμη των Υπολογιστών είναι το **δυαδικό δέντρο**. Κάθε κόμβος σ' ένα δυαδικό δέντρο έχε κατά μέγιστο, δύο παιδιά, το αριστερό παιδί και το δεξιό, που είναι αντίστοιχα οι ρίζες του αριστερού και δεξιού υποδέντρου του κόμβου.

Ένα τέτοιο παράδειγμα είναι το γενεαλογικό δένδρο ενός ατόμου, στο οποίο κάθε κόμβος έχει ακριβώς δύο παιδιά. (Σχήμα 2.2.15)

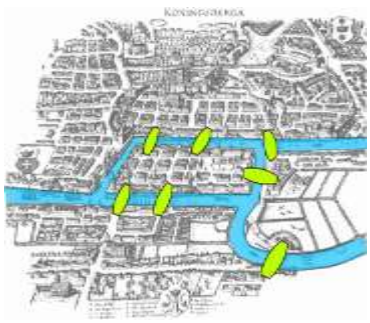


Σχήμα 2.2.15: Δυαδικό δέντρο

Γράφοι

Κατά τον 18ο αιώνα, την πόλη Königsberg της Πρωσίας διέσχιζε ο ποταμός Πρέγκελ και είχαν κατασκευαστεί 7 γέφυρες, για να δίνουν τη δυνατότητα στους κατοίκους να μετακινούνται πάνω από το ποτάμι. (Σχήμα 2.2.16),

Ο γρίφος που προσπαθούσαν να λύσουν οι κάτοικοι της πόλης ήταν ο εξής: Θα μπορούσε κάποιος να περπατήσει στην πόλη περνώντας από κάθε γέφυρα, ακριβώς μια φορά;

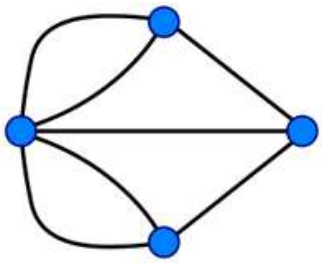


Σχήμα 2.2.16

Την απάντηση στο πρόβλημα έδωσε ο μεγάλος μαθηματικός Euler, ο οποίος απέδειξε ότι το παραπάνω εγχείρημα ήταν αδύνατο. Τι έκανε όμως ο Euler για να λύσει το πρόβλημα; Το αποσυνέδεσε από τις φυσικές του διαστάσεις και έφτιαξε μια απλή αναπαράστασή του στο χαρτί. Κάθε γέφυρα είναι μια ακμή, δηλαδή μια γραμμή που συνδέει δύο κομμάτια γης. Κάθε του "κουκκίδα" ονομάζεται *κορυφή* ή *κόμβος* ενώ οι γραμμές που συνδέουν τις κορυφές λέγονται *ακμές*.

Αυτό το σχήμα ονομάζεται *γράφος*.

Για ποιο λόγο τελικά ο γρίφος δεν μπορεί να λυθεί; Ο Euler συνειδητοποίησε πώς σε κάθε σημείο, εκτός ίσως από την αρχή και τον τερματισμό, θα έπρεπε να φτάνει ένας ζυγός αριθμός γεφυρών, ώστε να μπορούμε να το επισκεφθούμε, διασχίζοντας μια γέφυρα και φεύγοντας από μια άλλη. Κάτι τέτοιο όμως δεν ισχύει για τις γέφυρες του Königsberg!



Σχήμα 2.2.17: Γράφος

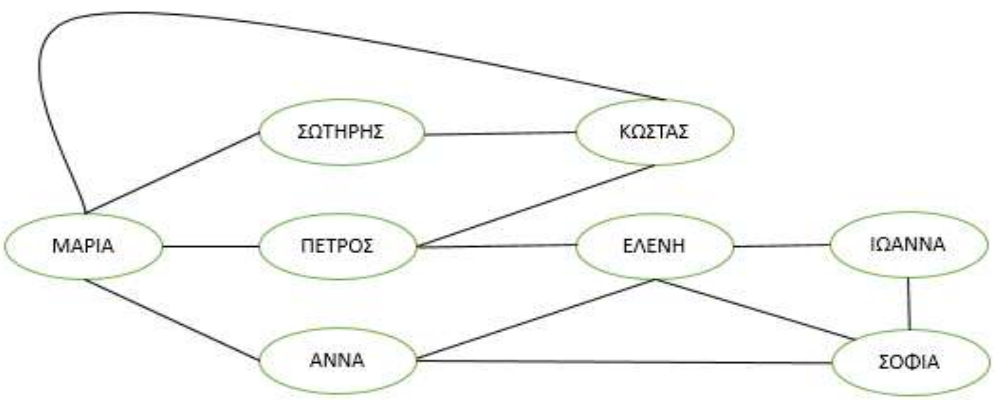
Λύνοντας το πρόβλημα με τον τρόπο που αναφέραμε, ο Euler έβαλε τα θεμέλια για έναν ολόκληρο κλάδο των Μαθηματικών και της Επιστήμης των Υπολογιστών, τη Θεωρία των Γράφων.

Οι γράφοι βρίσκουν πολλές εφαρμογές σε καθημερινά προβλήματα. Ενδεικτικά, αναφέρουμε τα δίκτυα μεταφορών και τηλεπικοινωνιών, τον παγκόσμιο ιστό.

Ας πάρουμε ένα δεύτερο παράδειγμα. Σε ένα κοινωνικό δίκτυο οι σχέσεις μεταξύ των ατόμων που συμμετέχουν σ' αυτό μπορούν να αναπαρασταθούν ως γράφος. Έστω ότι η Μαρία είναι φίλη με τον Πέτρο, την Άννα και τον Σωτήρη.

Ο Πέτρος έχει φίλους τον Κώστα και την Ελένη.
Ο Κώστας έχει φίλους τη Μαρία και το Σωτήρη.
Η Ελένη έχει φίλους τη Σοφία, την Ιωάννα και την Άννα.
Η Σοφία είναι, επίσης,,φίλη με την Ιωάννα και την Άννα.

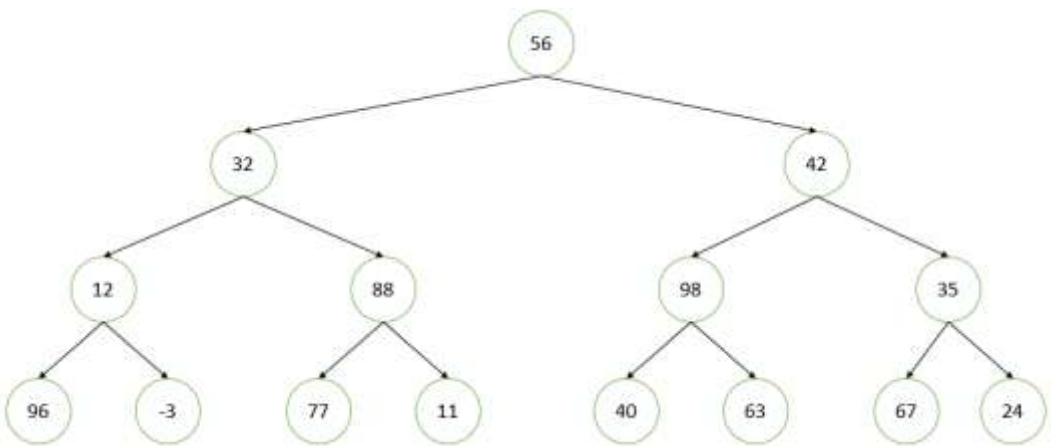
Μπερδευτήκατε; Ας κάνουμε τα πράγματα πιο ξεκάθαρα αναπαριστώντας τις παραπάνω σχέσεις με ένα γράφο (Σχήμα 2.2.18). Παρατηρούμε ότι στον παρακάτω γράφο, οι ακμές δείχνουν προς τις δύο κατευθύνσεις των κόμβων που ενώνουν. Προφανώς, όταν ένα άτομο είναι φίλος μ' ένα άλλο, ισχύει και το αντίστροφο, δηλαδή το δεύτερο άτομο είναι φίλος με το πρώτο. Ένας τέτοιος γράφος λέγεται *μη κατευθυνόμενος*. Αντίθετα, οι ιστοσελίδες στον Παγκόσμιο Ιστό είναι ένας *κατευθυνόμενος* γράφος, δηλαδή οι ακμές του είναι προσανατολισμένες προς μια κατεύθυνση, διότι μια ιστοσελίδα μπορεί να έχει ένα σύνδεσμο προς μια άλλη, αλλά δεν ισχύει απαραίτητα και το αντίστροφο.



Σχήμα 2.2.18:Μη κατευθυνόμενος γράφος

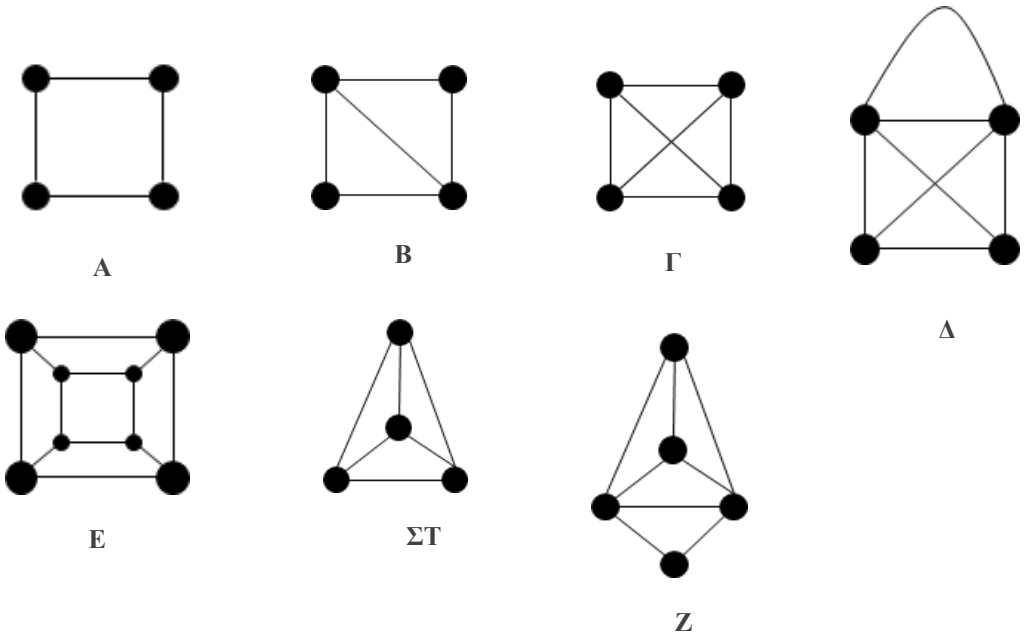
Ερωτήσεις – Δραστηριότητες

1. Δώστε 2 παραδείγματα εφαρμογών από την καθημερινή ζωή:
   - στοίβας
   - ουράς
   - απλά συνδεδεμένης λίστας
   - διπλά συνδεδεμένης λίστας
   - δυαδικού δέντρου
   - δυαδικού δέντρου αναζήτησης
   - γράφου
2. Ποια από τις δομές δεδομένων που συναντήσαμε, θεωρείτε την καταλληλότερη για να εφαρμόσουμε τη λειτουργία της αναίρεσης - της ακύρωσης δηλαδή των προηγούμενων ενεργειών μας - σε ένα πρόγραμμα όπως ο κειμενογράφος; Να αιτιολογήσετε την απάντησή σας.
3. Χρησιμοποιώντας το δυαδικό δέντρο του σχήματος 2.2.19 που ακολουθεί, να απαντήσετε στις παρακάτω ερωτήσεις:
   1. Ποια είναι η τιμή της ρίζας;
   2. Ποια είναι τα αδέλφια του 88;
   3. Ποιος είναι ο γονέας του -3;
   4. Ποιο είναι το ύψος του δέντρου;
   5. Ποια είναι τα παιδιά του 98;



Σχήμα 2.2.19

4. Ποια από τα παρακάτω σχήματα (Σχήμα 2.2.20) μπορούμε να σχεδιάσουμε χωρίς να σηκώσουμε το μολύβι μας από το χαρτί και περνώντας μόνο μια φορά από κάθε γραμμή;



Σχήμα 2.2.20

Χρήσιμες ιστοσελίδες

- ✓ Ένα πολύ ενδιαφέρον πρόβλημα είναι ο Χρωματισμός Γράφων. Στο σύνδεσμο, μπορείτε να διαβάσετε πληροφορίες και να εκτελέσετε τη σχετική δραστηριότητα. http://csunplugged.org/sites/default/files/activity_pdfs_other/graph%20colouring.el_.v6.pdf
- ✓ Επισκεφθείτε τη σελίδα του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου San Francisco για να παρακολουθήσετε μια οπτικοποίηση των δομών δεδομένων που αναφέρθηκαν στην παρούσα ενότητα. http://www.cs.usfca.edu/~galles/visualization/Algorithms.html

2.2.7. Εντολές και δομές αλγορίθμου

Εισαγωγή

Όταν περιγράφουμε έναν αλγόριθμο παίζουν ρόλο οι εντολές που δίνουμε αλλά και η **σειρά** με την οποία θα εκτελεστούν. Ένα μοντέλο προγραμματισμού που αναπτύχθηκε και χρησιμοποιείται ευρύτατα σήμερα είναι ο **δομημένος προγραμματισμός.** Σύμφωνα με τις αρχές του δομημένου προγραμματισμού όλα τα προβλήματα μπορούν να αντιμετωπιστούν με την χρήση 3 βασικών δομών, της δομής ακολουθίας, της επιλογής και της επανάληψης. Τις δομές αυτές θα εξετάσουμε παρακάτω.

Δομή ακολουθίας

Στη δομή της ακολουθίας η σειρά εκτέλεσης των εντολών είναι δεδομένη, δηλαδή οι εντολές εκτελούνται η μια μετά την άλλη με τη σειρά που αναφέρονται.

Θα υλοποιήσουμε έναν απλό αλγόριθμο που θα προσομοιώνει ένα "ρομπότ συζήτησης (chatbot)" με τον χρήστη.

Για αρχή, ο αλγόριθμός μας θα εμφανίζει απλά ένα μήνυμα χαιρετισμού στον χρήστη, όπως φαίνεται παρακάτω.

```
1 Αλγόριθμος ρομποτ
2            Εμφάνισε "Καλως ήρθες. Είμαι ο προσωπικός σου βοηθός"
3 Τέλος ρομποτ
```

Σχήμα 2.2.21: Εντολές αλγορίθμου σε ψευδογλώσσα

Στη γραμμή 1 βρίσκεται η εντολή **Αλγόριθμος** που ακολουθείται από ένα όνομα δικής μας επιλογής (εδώ, η λέξη *ρομπότ*). Κάθε αλγόριθμος που γράφουμε ξεκινάει με αυτό τον τρόπο. Αντίστοιχα, στην γραμμή 3 βρίσκεται η εντολή **Τέλος** και το όνομα που δώσαμε στον αλγόριθμο. Το παραπάνω ισχύει για κάθε αλγόριθμο που γράφουμε.

Στη γραμμή 2 βρίσκεται η εντολή **Εμφάνισε** ακολουθούμενη από ένα μήνυμα γραμμένο μέσα σε διπλά εισαγωγικά "". Η εντολή Εμφάνισε είναι μια εντολή εξόδου και έχει ως αποτέλεσμα την εμφάνιση τιμών στην οθόνη του υπολογιστή.

Θα επεκτείνουμε τώρα τον αλγόριθμό μας (Σχήμα 2.2.22), ώστε να ρωτάει τον χρήστη ποιο είναι το όνομα του και στη συνέχεια να δέχεται την τιμή που πληκτρολογείται. Στη συνέχεια θα του εμφανίζει ένα μήνυμα μαζί με το όνομα που

```
1 Αλγόριθμος ρομποτ
2            Εμφάνισε "Καλως ήρθες. Είμαι ο προσωπικός σου βοηθός"
3            Εμφάνισε "Πως σε λένε?"
4            Διάβασε ονομα
5            Εμφάνισε "Γεια σου ",ονομα,". Χαίρομαι που σε γνωρίζω"
6 Τέλος ρομποτ
```

Σχήμα 2.2.22: Εντολές σε ψευδογλώσσα

δόθηκε.

Οι εντολές που προστέθηκαν στον αλγόριθμο βρίσκονται στις γραμμές 3, 4 και 5. Η εντολή **Διάβασε** στη γραμμή 4 είναι μια εντολή εισόδου που ακολουθείται από ένα ή περισσότερα ονόματα μεταβλητών χωρισμένα με κόμμα (,). Κατά την εκτέλεσή της συγκεκριμένης εντολής (Σχήμα 2.2.22) ο χρήστης καλείται να εισάγει από το πληκτρολόγιο τόσες τιμές, όσες και οι μεταβλητές που ακολουθούν την εντολή **Διάβασε**. Στο παράδειγμά μας ο χρήστης θα πληκτρολογήσει μια τιμή, η οποία θα αποθηκευτεί στην μεταβλητή *όνομα*.

Στη γραμμή 5 χρησιμοποιούμε ξανά την εντολή **Εμφάνισε** με τέτοιο τρόπο ώστε να εμφανίσει μηνύματα σε συνδυασμό με την τιμή της μεταβλητής *όνομα*. Παρατηρούμε ότι η μεταβλητή χωρίζεται από τα μηνύματα με τον χαρακτήρα κόμμα (,).

Θα επεκτείνουμε λίγο ακόμα τον αλγόριθμο του ρομπότ μας, ώστε να κάνει έναν υπολογισμό για τον χρήστη (Σχήμα 2.2.23). Θα του ζητάει το έτος γέννησής του και με βάση το τρέχον έτος θα υπολογίζει και θα εμφανίζει την ηλικία του χρήστη, όπως παρακάτω.

Μια εντολή που συναντάμε για πρώτη φορά είναι η εντολή εκχώρησης τιμής σε μεταβλητές που βρίσκεται στις γραμμές 6 και 9 του αλγορίθμου. Για να εκχωρήσουμε δηλαδή να αποδώσουμε τιμή σε μια μεταβλητή χρησιμοποιούμε το σύμβολο ← που ονομάζεται τελεστής εκχώρησης.

```
1  Αλγόριθμος ρομποτ
2             Εμφάνισε "Καλως ήρθες. Είμαι ο προσωπικός σου βοηθός"
3             Εμφάνισε "Πως σε λένε?"
4             Διάβασε ονομα
5             Εμφάνισε "Γεια σου ",ονομα,". Χαίρομαι που σε γνωρίζω"
6             τρέχον_έτος ← 2014
7             Εμφάνισε "Ποιο είναι το έτος γέννησής σου?"
8             Διάβασε έτος_γεννησης
9             ηλικία ← τρέχον_έτος - έτος_γεννησης
10            Εμφάνισε "Επομένως είσαι ",ηλικία," χρονών!"
11 Τέλος ρομποτ
```

Σχήμα 2.2.23: Εντολές σε ψευδογλώσσα

Στον αλγόριθμο που υλοποιούμε η εντολή εκχώρησης στη γραμμή 6 έχει ως αποτέλεσμα η μεταβλητή *τρέχον_έτος* να πάρει την τιμή 2014, ενώ η εντολή εκχώρησης που βρίσκεται στη γραμμή 9 έχει ως αποτέλεσμα να υπολογιστεί η τιμή της αριθμητικής έκφρασης *τρέχον_έτος - έτος_γέννησης* και το αποτέλεσμα να αποθηκευθεί στην μεταβλητή με όνομα ηλικία.

Θα μπορούσαμε, φυσικά, να κάνουμε απευθείας τον υπολογισμό 2014 - έτος_γέννησης, χωρίς να χρησιμοποιήσουμε την μεταβλητή τρέχον_έτος.

Η γενική μορφή μιας εντολής εκχώρησης είναι `μεταβλητή ← έκφραση` με την οποία η μεταβλητή που βρίσκεται στο αριστερό μέρος του τελεστή εκχώρησης παίρνει την τιμή της έκφρασης που βρίσκεται στο δεξί μέρος. Μια έκφραση μπορεί να είναι αριθμητική ή λογική.

Για τη σύνταξη μιας αριθμητικής έκφρασης, δηλαδή μιας αριθμητικής παράστασης, εκτός από αριθμητικές τιμές και μεταβλητές μπορούμε να χρησιμοποιήσουμε και αριθμητικούς τελεστές (σύμβολα αριθμητικών πράξεων).

Οι αριθμητικοί τελεστές που χρησιμοποιούμε δίνονται στον παρακάτω πίνακα:

Τελεστής	Λειτουργία
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
DIV	Ακέραιο πηλίκο διαίρεσης
MOD	Υπόλοιπο ακέραιας διαίρεσης

Ιεραρχία
()
^
*,/,DIV,MOD
+,-

Σχήμα 2.2.24: Οι αριθμητικοί τελεστές και η ιεραρχία τους

Στην περίπτωση που έχουμε πράξεις τις ίδιας ιεραρχίας, για παράδειγμα το * και το DIV τότε εκτελούνται από αριστερά προς τα δεξιά.

Δομή Επιλογής

Στα περισσότερα προβλήματα που καλούμαστε να επιλύσουμε προκειμένου να πάρουμε τις κατάλληλες αποφάσεις πρέπει να εξετάσουμε μια ή περισσότερες λογικές προτάσεις (συνθήκες). Μια λογική πρόταση μπορεί να έχει μια από δύο τιμές, να είναι **Αληθής** δηλαδή να ισχύει, ή **Ψευδής** δηλαδή να μην ισχύει. Για παράδειγμα, η λογική πρόταση "Η Γη είναι επίπεδη" είναι Ψευδής, ενώ η λογική πρόταση "Η Γη γυρίζει" είναι πάντα Αληθής.

Συνήθως, για να υλοποιήσουμε μια λογική πρόταση χρειάζεται να κάνουμε μια σειρά από συγκρίσεις χρησιμοποιώντας κάποιον ή κάποιους συγκριτικούς τελεστές. Στη διάθεση μας έχουμε τα σύμβολα (σχήμα 2.2.25):

Συγκριτικοί Τελεστές	
>	μεγαλύτερο
>=	μεγαλύτερο ή ίσο
<	μικρότερο
<=	μικρότερο ή ίσο
=	ίσο
<>	διάφορο

Σχήμα 2.2.25: Οι συγκριτικοί τελεστές

Επιπλέον, πολύ συχνά χρειάζεται να συνδυάσουμε δύο ή περισσότερες λογικές προτάσεις μεταξύ τους, ώστε να δημιουργήσουμε μια σύνθετη λογική πρόταση. Τότε, χρησιμοποιούμε τους λογικούς τελεστές, δηλαδή το ΟΧΙ (άρνηση), το ΚΑΙ (σύζευξη) και το Ή (διάζευξη).

Το ΟΧΙ αντιστρέφει την τιμή μιας λογικής πρότασης. Για παράδειγμα, ενώ η πρόταση 30 > 17 είναι Αληθής, όταν βάλουμε μπροστά το ΟΧΙ, δηλαδή ΟΧΙ(30>17), γίνεται Ψευδής.

Ο λογικός τελεστής ΚΑΙ ενώνει 2 λογικές προτάσεις. Η τελική πρόταση που σχηματίζεται είναι Αληθής όταν και οι δύο προτάσεις είναι Αληθείς. Για παράδειγμα, η πρόταση $\chi > 15$ και $\chi < 40$ για μια αριθμητική μεταβλητή χ είναι Αληθής μόνο όταν το χ ανήκει στο διάστημα (15,40).

Τέλος, το Η΄ ενώνει 2 λογικές προτάσεις. Η τελική πρόταση είναι Αληθής όταν μια εκ των 2 προτάσεων ή και οι 2 προτάσεις είναι Αληθείς. Για παράδειγμα, η πρόταση $\chi > 20$ ή $\psi = 6$ για τις αριθμητικές μεταβλητές χ και ψ είναι Αληθής, αρκεί το ψ να είναι 6 ή το χ μεγαλύτερο του 20 ή να ισχύουν και τα 2 (σχήμα 2.2.26).

Πρόταση Α	**Πρόταση Β**	**Α ΚΑΙ Β**	**Α Ή Β**
Αληθής	Αληθής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής

Σχήμα 2.2.26: Οι λογικοί τελεστές

Απλή επιλογή

Ας πάρουμε ώς παράδειγμα τον υπολογισμό της τελικής βαθμολογίας ενός μαθήματος. Υπάρχουν μαθήματα που δεν εξετάζονται με γραπτές εξετάσεις στο τέλος του έτους, οπότε η τελική τους βαθμολογία είναι ο μέσος όρος των 2 τετραμήνων. Αντίθετα, για τα μαθήματα που εξετάζονται γραπτά η τελική βαθμολογία προκύπτει από το μέσο όρο της γραπτής εξέτασης και του μέσου όρου της βαθμολογίας των 2 τετραμήνων.

Θα υλοποιήσουμε έναν αλγόριθμο (σχήμα 2.2.27) που διαβάζει τους βαθμούς των 2 τετραμήνων και υπολογίζει το μέσο όρο τους. Στη συνέχεια θα ρωτάει αν το μάθημα εξετάζεται και γραπτά και αν η απάντηση που δίνει ο χρήστης είναι "ΝΑΙ" θα ζητά τη βαθμολογία της γραπτής εξέτασης και θα υπολογίζει την τελική βαθμολογία, όπως αναφέρθηκε παραπάνω.

```
1  Αλγόριθμος υπολογισμος_βαθμολογίας
2             Εμφάνισε "Δωστε το βαθμό του Α' τετραμήνου"
3             Διάβασε βαθμοςΑ
4             Εμφάνισε "Δωστε το βαθμό του Β' τετραμήνου"
5             Διάβασε βαθμοςΒ
6
7             τελικη_βαθμολογια ← (βαθμοςΑ + βαθμοςΒ)/2
8
9             Εμφάνισε "Το μάθημα εξετάζεται γραπτά;"
10            Διάβασε απαντηση
11            Αν απαντηση = "ΝΑΙ" ή απαντηση = "ναι" τότε
12               Εμφάνισε "Δώστε το βαθμό της γραπτής εξέτασης"
13               Διάβασε γραπτα
14               τελικη_βαθμολογια ← (τελικη_βαθμολογια + γραπτα)/2
15            Τέλος_αν
16
17            Εμφάνισε "Η τελική βαθμολογία του μαθήματος είναι ",τελικη_βαθμολογια
18 Τέλος υπολογισμος βαθμολογίας
```

Σχήμα 2.2.27: Αλγόριθμος σε ψευδογλώσσα

Στη γραμμή 11 βρίσκεται η εντολή απλής επιλογής:

Αν <συνθήκη> **τότε**
...
Τέλος_Αν

Με την εντολή Αν ελέγχουμε αν ισχύει μια λογική πρόταση, εδώ αν η απάντηση του χρήστη είναι το "ΝΑΙ" ή το "ναι". Εφόσον η πρόταση ισχύει, εκτελούνται οι εντολές που βρίσκονται μέσα στο Αν...Τέλος_Αν. Διαφορετικά, ο αλγόριθμος "αγνοεί" τις εντολές αυτές. Σε κάθε περίπτωση, η εκτέλεση συνεχίζει στις εντολές που βρίσκονται μετά το Τέλος_Αν.

Σύνθετη επιλογή

Σε περιπτώσεις που θέλουμε να περιγράψουμε τις ενέργειες που θα εκτελεστούν, όταν μια πρόταση ισχύει, και τις ενέργειες που θα εκτελεστούν, όταν η ίδια πρόταση ΔΕΝ ισχύει, χρησιμοποιούμε την εντολή σύνθετης επιλογής:

Αν <συνθήκη> **τότε**
...
Αλλιώς
...
Τέλος_Αν

Οι εντολές που βρίσκονται μέσα στο Αλλιώς εκτελούνται μόνο όταν η συνθήκη που εξετάζουμε ΔΕΝ ισχύει. Ένα παράδειγμα είναι ένας αλγόριθμος (σχήμα 2.2.28) που θα θέτει ερωτήσεις γνώσεων στον χρήστη και ανάλογα με την απάντησή του θα επιβραβεύει ή θα του εμφανίζει μήνυμα ενημερώνοντας τον για τη σωστή απάντηση. Στο τέλος θα του εμφανίζει τον αριθμό των ερωτήσεων που α-

```
Αλγόριθμος κουιζ_γνωσεων
           Εμφάνισε "Κουίζ στην Επιστήμη των Υπολογιστών"
           Εμφάνισε "Παρακαλώ γράψτε τις απαντήσεις σας με κεφαλαία"
           σωστά ← 0
           Εμφάνισε "Θεωρείται η πρώτη γυναίκα - προγραμματιστής"
           Διάβασε απαντηση
           Αν απαντηση = "ADA LOVELACE" τότε
             Εμφάνισε "Μπράβο"
             σωστά ← σωστά + 1
           αλλιώς
             Εμφάνισε "Η σωστή απάντηση είναι ADA LOVELACE"
             Εμφάνισε "Αγγλίδα μαθηματικός. Έγραψε τον πρώτο 'αλγόριθμο'"
           Τέλος_αν

           Εμφάνισε "Ποιος έβαλε τα θεμέλια της Επιστήμης των Υπολογιστών τον 20ο αιώνα;"
           Διάβασε απαντηση
           Αν απαντηση = "ALAN TURING" τότε
             Εμφάνισε "Μπράβο"
             σωστά ← σωστά + 1
           αλλιώς
             Εμφάνισε "Η σωστή απάντηση είναι ALAN TURING"
             Εμφάνισε "Περισσότερα στο http://en.wikipedia.org/wiki/Alan_Turing"
           Τέλος_αν

           Εμφάνισε "Απάντησες σωστά σε ", σωστά, " ερωτήσεις"
Τέλος κουιζ_γνωσεων
```

Σχήμα 2.2.28: Αλγόριθμος σε ψευδογλώσσα

πάντησε σωστά.

Πολλαπλή επιλογή

Σε πολλά προβλήματα, οι περιπτώσεις που θέλουμε να εξετάσουμε είναι 2 ή περισσότερες. Ας πάρουμε ώς παράδειγμα έναν αλγόριθμο (σχήμα 2.2.29) που δέχεται 2 αριθμούς και το σύμβολο της πράξης (+,-,*) που επιθυμεί ο χρήστης και του εμφανίζει το αποτέλεσμα. Σε περίπτωση που ο χρήστης δώσει κάποιο άλλο σύμβολο, ο αλγόριθμος μας θα εμφανίζει ένα μήνυμα σφάλματος.

Η πολλαπλή επιλογή λειτουργεί ως εξής: Αρχικά, εξετάζεται η πρώτη συνθήκη. Αν ισχύει εκτελούνται οι εντολές που βρίσκονται μέσα σε αυτή και η εκτέλεση συνεχίζει μετά το Τέλος_Αν. Διαφορετικά, εξετάζεται η δεύτερη συνθήκη, που με την ίδια λογική εφόσον ισχύει εκτελούνται οι εντολές που βρίσκονται μέσα σε αυτή. Διαφορετικά εξετάζεται η 3η συνθήκη κ.ο.κ. Εφόσον δεν ισχύει καμία από τις συνθήκες που εξετάζονται και υπάρχει η εντολή Αλλιώς, τότε εκτελούνται οι εντολές που βρίσκονται μέσα στο Αλλιώς.

```
1  Αλγόριθμος αριθμομηχανη
2             Εμφάνισε "Δωστε τον πρωτο αριθμό"
3             Διάβασε αρ1
4             Εμφάνισε "Δώστε το δεύτερο αριθμό"
5             Διάβασε αρ2
6             Εμφάνισε "Δώστε το σύμβολο της πράξης (+,-,*)"
7             Διάβασε πράξη
8
9             Αν πράξη = "+" τότε
10               αποτέλεσμα ← αρ1 + αρ2
11            αλλιώς_αν πράξη = "-" τότε
12               αποτέλεσμα ← αρ1 - αρ2
13            αλλιώς_αν πράξη = "*" τότε
14               αποτέλεσμα ← αρ1 * αρ2
15            αλλιώς
16               Εμφάνισε "Τα επιτρεπόμενα σύμβολα είναι +,-,*"
17            Τέλος_αν
18
19            Αν πράξη = "+" ή πράξη = "-" ή πράξη = "*" τότε
20               Εμφάνισε αρ1," ",πράξη," ",αρ2," = ",αποτέλεσμα
21            Τέλος_αν
22 Τέλος αριθμομηχανη
```

Σχήμα 2.2.29: Αλγόριθμος σε ψευδογλώσσα

Η γενική μορφή της εντολής πολλαπλής επιλογής Αν...Αλλιώς_Αν δίνεται παρακάτω:

Αν <συνθηκη1> **τότε**
…
Αλλιώς_αν <συνθηκη2> **τότε**
…
Αλλιώς_αν <συνθηκη3> **τότε**
…
.**Αλλιώς_αν** <συνθήκηΝ> **τότε**
…
Αλλιώς (προαιρετικά)
…
Τελος_Αν

Δομή Επανάληψης

Οι υπολογιστές είναι ιδιαίτερα καλοί στο να κάνουν το ίδιο πράγμα ξανά και ξανά. Χρησιμοποιούνται για τις βαρετές, επαναλαμβανόμενες ενέργειες για τις οποίες δεν χρειάζεται ευφυΐα, αλλά τυφλή και πιστή τήρηση των εντολών. Μια επαναληπτική διαδικασία μπορεί να εκτελείται για έναν ορισμένο αριθμό φορών ή να ελέγχεται από μια συνθήκη.

Για παράδειγμα, οι οδηγίες που θα δώσουμε σε κάποιον για να σχεδιάσει ένα τετράγωνο στο πάτωμα με πλευρά 10 βημάτων είναι:

προχώρησε 10 βήματα στρίψε 90 μοίρες αριστερά
προχώρησε 10 βήματα στρίψε 90 μοίρες αριστερά
προχώρησε 10 βήματα στρίψε 90 μοίρες αριστερά
προχώρησε 10 βήματα στρίψε 90 μοίρες αριστερά

Παρατηρούμε ότι οι εντολές *προχώρησε 10 βήματα* και *στρίψε 90 μοίρες αριστερά* εκτελούνται 4 φορές. Άρα, μπορούμε πιο απλά να συμπυκνώσουμε τις οδηγίες μας ως εξής:

επανάλαβε 4 φορές τις ενέργειες
    προχώρησε 10 βήματα
    στρίψε 90 μοίρες

Το παραπάνω παράδειγμα είναι μια περίπτωση επαναληπτικής διαδικασίας, που εκτελείται ένα συγκεκριμένο αριθμό φορών.

Το ίδιο ισχύει και στην περίπτωση που ένα άτομο καταγράφει καθημερινά και για μια εβδομάδα τις θερμίδες που καταναλώνει. Ακολουθεί αλγόριθμος (σχήμα 2.2.30) που διαβάζει τις τιμές των θερμίδων για κάθε ημέρα της εβδομάδας, υπολογίζει και εμφανίζει το μέσο όρο των θερμίδων που κατανάλωσε το άτομο ανά ημέρα.

```
1  Αλγόριθμος θερμιδομετρητης
2              συνολικες_θερμιδες ← 0
3              Για ημερα από 1 μέχρι 7 με_βήμα 1
4                  Εμφάνισε "Δώστε τις θερμίδες της ",ημερα,"ης μέρας"
5                  Διάβασε θερμιδες
6                  συνολικες_θερμιδες ← συνολικες_θερμιδες + θερμιδες
7              Τέλος_επανάληψης
8              μέσος_όρος ← συνολικες_θερμιδες / 7
9              Εμφάνισε "Ο ημερήσιος μέσος όρος των θερμίδων που καταναλώσατε είναι ",μέσος_όρος
10 Τέλος θερμιδομετρητης
```

Σχήμα 2.2.30: Αλγόριθμος σε ψευδογλώσσα

Στη γραμμή 3 του αλγορίθμου χρησιμοποιείται η εντολή επανάληψης **Για**. Κατά την εκτέλεση της εντολής Για, η μεταβλητή ημέρα παίρνει αρχικά την τιμή 1. Εφόσον δεν έχει ξεπεράσει την τιμή 7 εκτελούνται οι εντολές που βρίσκονται ανάμεσα στην εντολή Για και την εντολή Τέλος_Επανάληψης. Στη συνέχεια η μεταβλητή ημέρα αυξάνεται κατά 1, γίνεται ο έλεγχος αν έχει ξεπεράσει την τιμή 7, κ.ο.κ.

Η γενική μορφή της εντολής Για δίνεται παρακάτω:

Για <μεταβλητή> **από** <αρχική τιμή> **μέχρι** <τελική τιμή> **με_βήμα** <βήμα>
….
Τέλος_Επανάληψης

Η εντολή Για χρησιμοποιεί μια μεταβλητή που παίζει το ρόλο του μετρητή. Η αρχική τιμή του μετρητή δίνεται από την εντολή *από*, ενώ ο τρόπος που αλλάζει

ο μετρητής σε κάθε επανάληψη δίνεται από την εντολή *με_βήμα*. Το βήμα μπορεί να είναι οποιοσδήποτε ακέραιος ή πραγματικός αριθμός, εκτός από το μηδέν. Αν το βήμα είναι θετικό, ο μετρητής αυξάνεται σε κάθε επανάληψη, ενώ αν είναι αρνητικό μειώνεται. Όταν το βήμα έχει την τιμή 1 μπορούμε να παραλείψουμε την αναγραφή του. Τέλος, η επανάληψη τερματίζεται, όταν ο μετρητής ξεπεράσει ή γίνει μικρότερος (ανάλογα με το πρόσημο που έχει το βήμα) από την τιμή που καθορίζει η εντολή *μέχρι*. Η εντολή Για χρησιμοποιείται μόνο όταν γνωρίζουμε τον αριθμό των επαναλήψεων.

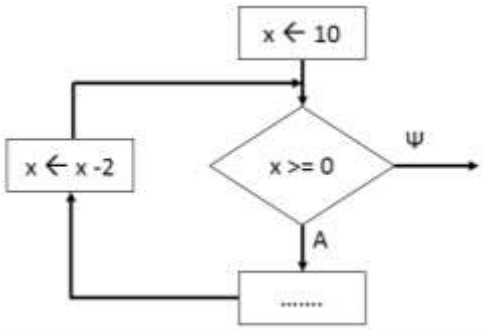
Για παράδειγμα η εντολή

Για χ από 10 μέχρι 0 με_βήμα -2 θα εκτελεστεί 6 φορές (2.2.31)

ενώ η εντολή

Για χ από 10 μέχρι 0 με _βήμα 2 δεν θα εκτελεστεί καμία φορά.

Μπορείτε να το αιτιολογήσετε;



Σχήμα 2.2.31

Ας δούμε ένα διαφορετικό παράδειγμα. Για να συνδεθεί κάποιος σε μια σελίδα κοινωνικής δικτύωσης πρέπει να δώσει το όνομα χρήστη και το συνθηματικό που έχει επιλέξει. Σε περίπτωση που δώσει λανθασμένα στοιχεία του ζητείται να τα δώσει ξανά. Η διαδικασία συνεχίζεται μέχρι το όνομα χρήστη και το συνθηματικό που δίνονται να είναι τα σωστά.

Η παραπάνω διαδικασία αποτελεί περίπτωση επαναληπτικής διαδικασίας που ελέγχεται από συνθήκη, προκειμένου να δοθούν το σωστό όνομα χρήστη και συνθηματικό. Ο αριθμός των επαναλήψεων δεν είναι γνωστός εκ των προτέρων.

```
1  Αλγόριθμος δικαιο_νομισμα
2             Εμφάνισε "Δώστε το αποτέλεσμα της πρώτης ριψης (Κ ή Γ)"
3             Διάβασε ριψη1
4             Εμφάνισε "Δώστε το αποτέλεσμα της δεύτερης ριψης (Κ ή Γ)"
5             Διάβασε ριψη2
6
7             Όσο ριψη1 = ριψη2 επανάλαβε
8                 Εμφάνισε "Πρέπει να ξαναρίξετε το νόμισμα"
9                 Εμφάνισε "Δώστε το αποτέλεσμα της πρώτης ριψης (Κ ή Γ)"
10                Διάβασε ριψη1
11                Εμφάνισε "Δώστε το αποτέλεσμα της δεύτερης ριψης (Κ ή Γ)"
12                Διάβασε ριψη2
13            Τέλος_επανάληψης
14
15            Αν ριψη1 = "Κ" τότε
16               Εμφάνισε "Κορώνα"
17            αλλιώς
18               Εμφάνισε "Γράμματα"
```

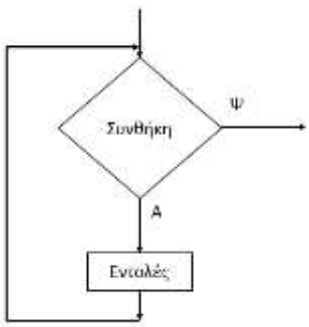
Σχήμα 2.2. 32: Αλγόριθμος σε ψευδογλώσσα

Ένα ακόμα παράδειγμα είναι ο αλγόριθμος του "δίκαιου" νομίσματος. Όταν ρίχνουμε ένα νόμισμα στο παιχνίδι "Κορώνα - Γράμματα" για να είμαστε σίγουροι ότι το νόμισμα είναι δίκαιο το ρίχνουμε 2 φορές. Αν και τις δύο φορές το νόμισμα πέσει από την ίδια πλευρά τότε η διαδικασία θεωρείται άκυρη και την επαναλαμβάνουμε. Αν οι 2 ρίψεις διαφέρουν τότε μετράει η αρχική ρίψη. Δηλαδή:

Κ → Γ κερδίζει η Κορώνα
Γ → Κ κερδίζουν τα γράμματα
Γ → Γ ή Κ→ Κ η διαδικασία επαναλαμβάνεται

Ο αλγόριθμος που προσομοιώνει τη διαδικασία που περιγράψαμε δίνεται στο σχήμα 2.2.32.

Όπως βλέπουμε στο σχήμα 2.2.32, στη γραμμή 7 βρίσκεται η εντολή **Όσο**. Οι εντολές που βρίσκονται ανάμεσα στην εντολή Όσο και στην εντολή Τέλος_Επανάληψης θα εκτελούνται ξανά και ξανά, εφόσον η συνθήκη που εξετάζει η Όσο (ριψη1 = ριψη2) παραμένει Αληθής. Προφανώς, αν αρχικά η συνθήκη που εξετάζει η Όσο δεν ισχύει τότε η επανάληψη δεν εκτελείται ούτε μια φορά.



Παρακάτω δίνεται η γενική μορφή της εντολής Όσο και το διάγραμμα ροής της (Σχήμα 2.2.33).

Όσο <συνθήκη> **επανάλαβε**
….
Τέλος_επαναληψης

Η εντολή Όσο μπορεί να χρησιμοποιηθεί για την επίλυση όλων των προβλημάτων. Συνήθως, την χρησιμοποιούμε, όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων.

Σχήμα 2.2.33

Ερωτήσεις – Δραστηριότητες

1. Να καταγράψετε τα αποτελέσματα του παρακάτω αλγορίθμου.

```
1  Αλγόριθμος φιμπο
2              α ← 0
3              β ← 1
4              Εμφάνισε α
5              Εμφάνισε β
6              Για ι από 3 μέχρι 10
7                  β ← α + β
8                  Εμφάνισε β
9                  α ← β - α
10             Τέλος_επανάληψης
11 Τέλος φιμπο
```

Σχήμα 2.2.34

2. Παρακάτω δίνονται οι οδηγίες για το ξεκλείδωμα ενός κινητού τηλεφώνου με τον αριθμό PIN διατυπωμένες σε φυσική γλώσσα κατά βήματα:
    1. *Εισάγετε τον αριθμό PIN*
    2. *Αν ο αριθμός PIN είναι σωστός το κινητό ξεκλειδώνει.*
    3. *Διαφορετικά εισάγετε για 2η φορά τον αριθμό PIN.*
    4. *Αν ο αριθμός PIN είναι σωστός το κινητό ξεκλειδώνει.*
    5. *Διαφορετικά εισάγετε για 3η φορά τον αριθμό PIN.*
    6. *Αν ο αριθμός είναι σωστός το κινητό ξεκλειδώνει.*
    7. *Διαφορετικά πρέπει να εισάγετε πλέον τον αριθμό PUK, για να ξεκλειδώσει το κινητό.*
    1. Διατυπώστε (σε φυσική γλώσσα κατά βήματα) πιο συνοπτικά τον παραπάνω αλγόριθμο χρησιμοποιώντας μια επαναληπτική διαδικασία.
    2. Διατυπώστε τον ίδιο αλγόριθμο χρησιμοποιώντας ψευδογλώσσα. Διαλέξτε έναν αριθμό της αρεσκείας σας, ως σωστό PIN.
3. Δίνεται αλγόριθμος που προσομοιώνει μια διαδικασία αντίστροφης μέτρησης δευτερολέπτων. Ο χρήστης δίνει την αρχική τιμή των δευτερολέπτων από τα οποία θα ξεκινήσει η μέτρηση.

```
Αλγόριθμος ατερμων
           Εμφάνισε "Πρόγραμμα αντίστροφης μέτρησης"
           Εμφάνισε "Δώσε τον αριθμό των δευτερολέπτων που θέλεις να μετρήσω"
           Διάβασε δευτερολεπτα
           Εμφάνισε "Κάθε φορά θα μειώνω τα δευτερόλεπτα κατά 1 μέχρι να μηδενιστούν"
           Όσο δευτερολεπτα > 0 επανάλαβε
               Εμφάνισε "Δευτερόλεπτα που απομένουν ",δευτερολεπτα
           Τέλος_επανάληψης
Τέλος ατερμων
```

Σχήμα 2.2 35: Αλγόριθμος σε ψευδογλώσσα

Γιατί δε δουλεύει σωστά ο παραπάνω αλγόριθμος; Να προτείνετε τις απαραίτητες τροποποιήσεις, ώστε να επιτελεί τη λειτουργία του.

2.2.8. Βασικές λειτουργίες δομών δεδομένων

Εισαγωγή

Κάθε δομή δεδομένων χαρακτηρίζεται από ένα σύνολο επιτρεπτών λειτουργιών επι των δεδομένων της. Οι βασικότερες λειτουργίες είναι :

- Προσπέλαση
- Αναζήτηση
- Ταξινόμηση
- Εισαγωγή
- Διαγραφή

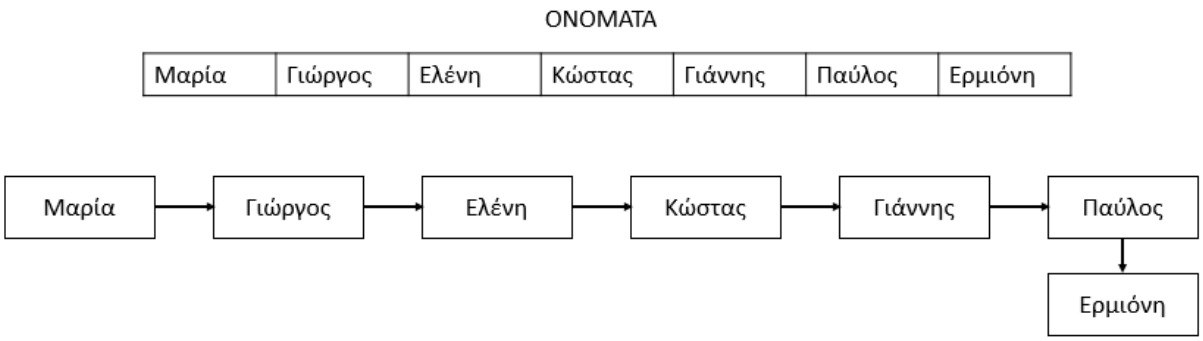
Κάθε δομή υλοποιεί αποδοτικότερα κάποια ή κάποιες από αυτές τις λειτουργίες. Για παράδειγμα, λειτουργίες όπως η εισαγωγή και η διαγραφή υλοποιούνται εύκολα με λίστες.

Παρακάτω θα εξετάσουμε μια σειρά από τεχνικές για την υλοποίηση των παραπάνω λειτουργιών.

Προσπέλαση

Προσπέλαση ονομάζεται η πρόσβαση σε έναν κόμβο μιας δομής δεδομένων. Είναι απαραίτητη λειτουργία, αφού χωρίς αυτή δεν θα μπορούσαμε να επέμβουμε στα περιεχόμενα μιας δομής, ώστε να εξετάσουμε ή να τροποποιήσουμε το περιεχόμενό τους.

Η λειτουργία της προσπέλασης γίνεται αποδοτικότερα σ' έναν πίνακα παρά σε μια λίστα, μιας και η πρόσβαση σε έναν κόμβο του πίνακα είναι άμεση, ενώ στη λίστα πρέπει να διασχίσουμε όλα τα προηγούμενα στοιχεία.



Σχήμα 2.2.36

Στο παραπάνω σχήμα (Σχήμα 2.2.36) φαίνονται τα ονόματα 7 ατόμων αποθηκευμένα σε πίνακα και σε συνδεδεμένη λίστα, αντίστοιχα. Αν θέλαμε να προσπελάσουμε το 4ο στοιχείο του πίνακα θα γράφαμε απλά ΟΝΟΜΑΤΑ[4], ενώ για τη λίστα θα έπρεπε να προσπελάσουμε τον κόμβο με την τιμή "Μαρία", στη συνέχεια τον κόμβο με την τιμή "Γιώργος" και τον κόμβο με την τιμή "Ελένη", προκειμένου να φτάσουμε στο 4ο στοιχείο της.

Στη συνέχεια θα μελετήσουμε κάποια παραδείγματα προσπέλασης σε πίνακα. Έστω ότι σε ένα σχολείο εργάζονται 50 καθηγητές. Για κάθε καθηγητή το σχολείο καταχωρίζει το όνομά του και τις ώρες που διδάσκει.

Για την αποθήκευση των δεδομένων μπορούμε να χρησιμοποιήσουμε δύο μονοδιάστατους πίνακες 50 θέσεων, έναν για τα ονόματα και έναν για τις ώρες διδασκαλίας (Σχήμα 2.2.37).

```
1 Αλγόριθμος σχολειο
2             Για καθ από 1 μέχρι 50
3                 Εμφάνισε "Δώσε το όνομα του ",καθ,"ου καθηγητή"
4                 Διάβασε ΟΝΟΜΑ[καθ]
5                 Εμφάνισε "Δώσε τις ώρες που διδάσκει στο σχολείο ο/η ",ΟΝΟΜΑ[καθ]
6                 Διάβασε ΩΡΕΣ[καθ]
7             Τέλος_επανάληψης
```

Σχήμα 2.2.37

Στις γραμμές 4 και 6 γίνεται προσπέλαση των πινάκων ΟΝΟΜΑ και ΩΡΕΣ, για την καταχώριση των τιμών που πληκτρολογεί ο χρήστης.

Έστω ότι το Υπουργείο Παιδείας ζητάει από το σχολείο τα ονόματα των καθηγητών που διδάσκουν πάνω από 12 ώρες. Ο αλγόριθμος μας θα επεκταθεί, όπως παρακάτω (Σχήμα 2.238).

```
9       Για καθ από 1 μέχρι 50
10          Αν ΩΡΕΣ[καθ] > 12 τότε
11              Εμφάνισε "Ο/Η ",ΟΝΟΜΑ[καθ]," εργάζεται πάνω από 12 ώρες"
12          Τέλος_αν
13      Τέλος_επανάληψης
```

Σχήμα 2.2.38

Οι επεξεργασίες που κάναμε στο παραπάνω παράδειγμα προϋποθέτουν τη λειτουργία της προσπέλασης, δηλαδή της πρόσβασης στα στοιχεία του πίνακα.

Αναζήτηση

Έστω ότι έχουμε μπροστά μας μια τράπουλα με χαρτιά, γυρισμένη έτσι ώστε να βλέπουμε το πίσω μέρος της.

Ας υποθέσουμε ότι ψάχνουμε το χαρτί Άσσος Κούπα. Μπορείτε να περιγράψετε έναν τρόπο για να βρούμε το χαρτί; Αν το χαρτί αυτό δεν υπήρχε στην τράπουλα; Πότε θα μπορούσαμε να απαντήσουμε με σιγουριά ότι δεν θα το βρίσκαμε;

Οι περισσότεροι από εμάς για να επιλύσουν ένα πρόβλημα, όπως το προηγούμενο θα ξεκινούσαν να αναποδογυρίσουν το πρώτο χαρτί της τράπουλας. Αν ήταν ο Άσσος Κούπα θα σταματούσαν το ψάξιμο, διαφορετικά θα συνέχιζαν με το δεύτερο χαρτί, το τρίτο κοκ. Τελικά, θα μπορούσαν να απαντήσουν ότι το χαρτί δεν υπάρχει στην τράπουλα όταν έφταναν και στο τελευταίο χαρτί χωρίς να έχουν βρει τον Άσσο Κούπα.

Αυτός ο τρόπος αναζήτησης ονομάζεται **σειριακή** ή **γραμμική** αναζήτηση και είναι ένας απλός αλγόριθμος που χρησιμοποιούμε στην καθημερινότητά μας. Μερικά ακόμα παραδείγματα είναι ο τρόπος που ψάχνουμε για ένα βιβλίο στη βιβλιοθήκη μας, για ένα αντικείμενο σε μια λίστα με ψώνια, για ένα φαγητό στο μενού ενός εστιατορίου και πολλά άλλα.

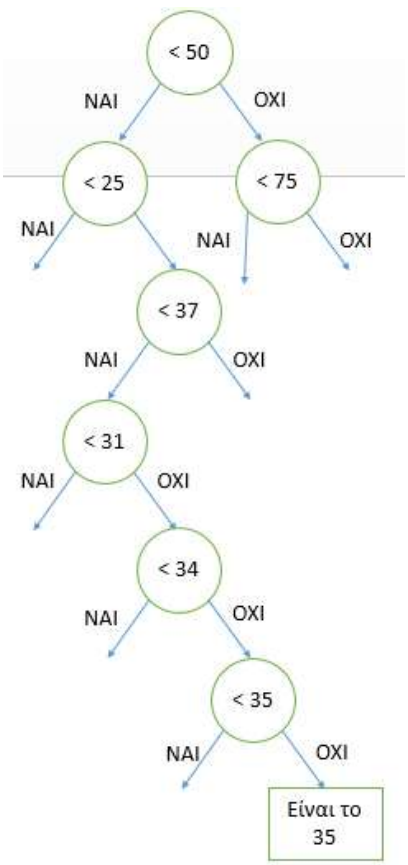
Ακολουθεί τμήμα αλγορίθμου (Σχήμα 2.2.39) που υλοποιεί τη σειριακή αναζήτηση μιας τιμής σε έναν πίνακα Α με Ν στοιχεία.

```
ι ← 1
βρεθηκε ← Ψευδής
Όσο ι ≤ Ν και βρεθηκε = Ψευδής επανάλαβε
    Αν Α[ι] = τιμη_που_ψαχνουμε τότε
       βρεθηκε ← Αληθής
       θεση ← ι
    αλλιώς
       ι ← ι + 1
    Τέλος_αν
Τέλος_επανάληψης

Αν βρεθηκε = Αληθής τότε
   Εμφάνισε "Η τιμή που ψάχνεται βρέθηκε στη θέση ",θεση
αλλιώς
   Εμφάνισε "Η τιμή αναζήτησης δεν υπάρχει"
Τέλος_αν
```

Σχήμα 2.2.39

Δυαδική Αναζήτηση



Σχήμα 2.2.40

Πόσες προσπάθειες χρειάζονται για να μαντέψουμε έναν αριθμό από το 1 έως το 100 που έβαλε ένας φίλος μας στο μυαλό του; Μα φυσικά 7 προσπάθειες το πολύ! Όσο απίθανος και αν ακούγεται ο παραπάνω ισχυρισμός, μπορούμε να εφαρμόσουμε την μέθοδο που ακολουθεί.

Αρχικά θα τον ρωτήσουμε αν ο αριθμός είναι μικρότερος του 50. Έστω ότι απαντάει ΝΑΙ. Η επόμενη ερώτηση θα είναι αν ο αριθμός είναι μικρότερος του 25. Έστω ότι απαντάει ΟΧΙ. Η επόμενη ερώτηση θα είναι αν ο αριθμός είναι μικρότερος του 37 και ούτω καθεξής. Στο σχήμα που φαίνεται δίπλα (Σχήμα 2.2.40) φαίνεται μια πιθανή ακολουθία ερωτήσεων και απαντήσεων.

Σας θυμίζει κάτι το σχήμα αυτό;! Είναι ένα δυαδικό δέντρο αναζήτησης!

Ο παραπάνω τρόπος αναζήτησης μιας τιμής ονομάζεται **δυαδική αναζήτηση** και μπορεί να εφαρμοστεί μόνο όταν οι τιμές είναι ταξινομημένες. Η λογική της μεθόδου είναι ότι χωρίζουμε κάθε φορά το διάστημα που απομένει να ψάξουμε στο μισό, οπότε υπάρχουν 3 περιπτώσεις. Είτε η τιμή που ψάχνουμε να είναι το μεσαίο στοιχείο,

είτε η τιμή που ψάχνουμε να είναι μικρότερη από το μεσαίο στοιχείο, είτε η τιμή που ψάχνουμε να είναι μεγαλύτερη από το μεσαίο στοιχείο.

Ακολουθεί τμήμα αλγορίθμου (Σχήμα 2.241) που υλοποιεί τη δυαδική αναζήτηση σ' έναν ταξινομημένο πίνακα Α μεγέθους Ν:

```
πρωτο ← 1
τελευταιο ← N
βρεθηκε ← Ψευδής
Όσο πρωτο ≤ τελευταίο και βρεθηκε = Ψευδής επανάλαβε
    μεσαιο ← (πρωτο + τελευταιο) div 2
    Αν A[μεσαιο] = τιμη_που_ψαχνουμε τότε
       βρεθηκε ← Αληθής
    αλλιώς_αν A[μεσαιο] < τιμη_που_ψαχνουμε τότε
       πρωτο ← μεσαιο + 1
    αλλιώς
       τελευταιο ← μεσαιο - 1
    Τέλος_αν
Τελος_επαναληψης

Αν βρεθηκε = Αληθής τότε
   Εμφάνισε "Η τιμή που ψάχνεται βρέθηκε στη θέση ",μεσαιο
αλλιώς
   Εμφάνισε "Η τιμή αναζήτησης δεν υπάρχει"
Τέλος_αν
```

Σχήμα 2.2.41

Ταξινόμηση

Πολύ συχνά χρειάζεται να οργανώσουμε τα δεδομένα μας με κάποια διάταξη. Χαρακτηριστικά παραδείγματα αποτελούν, οι ομάδες ποδοσφαίρου σε ένα πρωτάθλημα, οι οποίες κατατάσσονται με βάση τη βαθμολογία τους, ένας καθηγητής, ο οποίος τοποθετεί τα γραπτά των μαθητών του αλφαβητικά, καθως και τα αποτελέσματα μιας μηχανής αναζήτησης, τα οποία τοποθετούνται με βάση το πόσο σχετικές είναι οι σελίδες με αυτό που αναζητούμε.

	ΟΝΟΜΑ
1	Αλέκος
	Σωτήρης
	Θοδωρής
	Πέτρος
	Γιάννης
6	Λευτέρης

	ΧΡΟΝΟΣ
1	18,9
	16,3
	21,4
	17,5
	15,9
6	19,7

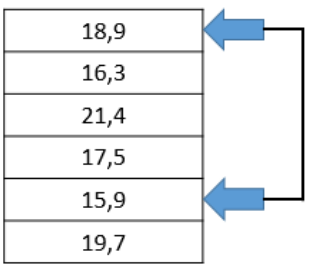
Σχήμα 2.2.42

Ας πάρουμε σαν παράδειγμα έναν αγώνα δρόμου 100μ στον οποίο συμμετείχαν 6 αθλητές. Έχουμε αποθηκεύσει (Σχήμα 2.2.42) σε 2 μονοδιάστατους πίνακες τα ονόματα και τους χρόνους των 6 αθλητών, αντίστοιχα. Ένας τρόπος να τοποθετήσουμε σε σειρά τους αθλητές ανάλογα με τον χρόνο τους είναι ο παρακάτω:

Αρχικά, βρίσκουμε τον μικρότερο χρόνο και τον τοποθετούμε στην πρώτη θέση του πίνακα (Σχήμα 2.2.43), ενώ ο χρόνος που βρισκόταν στην πρώτη θέση τοποθετείται εκεί που ήταν ο μικρότερος. Προφανώς, η εναλλαγή των τιμών εφαρμόζεται και στα ονόματα, ώστε να διατηρηθεί η αντιστοιχία.

	ΟΝΟΜΑ
1	Αλέκος
	Σωτήρης
	Θοδωρής
	Πέτρος
	Γιάννης
6	Λευτέρης

	ΧΡΟΝΟΣ
1	18,9
	16,3
	21,4
	17,5
	15,9
6	19,7



Σχήμα 2.2.43

	ΟΝΟΜΑ
1	Γιάννης
	Σωτήρης
	Θοδωρής
	Πέτρος
	Αλέκος
6	Λευτέρης

	ΧΡΟΝΟΣ
1	15,9
	16,3
	21,4
	17,5
	18,9
6	19,7

	ΟΝΟΜΑ
1	Γιάννης
	Σωτήρης
	Πέτρος
	Αλέκος
	Λευτέρης
6	Θοδωρής

	ΧΡΟΝΟΣ
1	15,9
	16,3
	17,5
	18,9
	19,7
6	21,4

Σχήμα 2.2.44

Στη συνέχεια επαναλαμβάνουμε το προηγούμενο βήμα, βρίσκοντας το μικρότερο χρόνο στον πίνακα, από τη 2η θέση και μετά, και κάνουμε την εναλλαγή του μικρότερου χρόνου με το στοιχείο της 2ης θέσης. Το παραπάνω επαναλαμβάνεται 5 φορές, οπότε και όλα τα στοιχεία του πίνακα ταξινομούνται (Σχήμα 2.2.44).

Η μέθοδος ταξινόμησης που περιγράφηκε ονομάζεται ταξινόμηση με επιλογή. Ακολουθεί τμήμα αλγορίθμου (Σχήμα 2.2.45) που την υλοποιεί σε πίνακα Α με μέγεθος Ν.

```
Για ι από 1 μέχρι Ν-1
    μικροτερο ← Α[ι]
    θεση_μικροτερου ← ι
    Για κ από ι+1 μέχρι Ν
        Αν Α[κ] < μικροτερο τότε
            μικροτερο ← Α[κ]
            θεση_μικροτερου ← κ
        Τέλος_αν
    Τέλος_επανάληψης
    Αντιμετάθεσε Α[ι], Α[θεση_μικροτερου]
Τέλος_επανάληψης
```

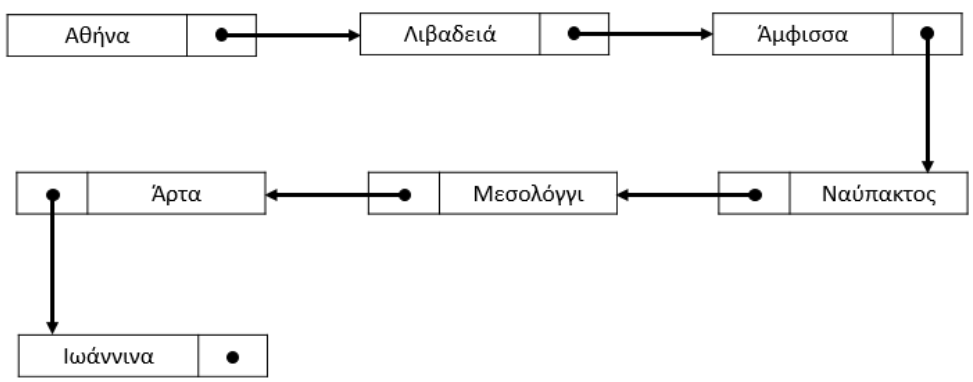
Σχήμα 2.2.45: Ταξινόμηση με επιλογή

Η ταξινόμηση με επιλογή είναι ένας απλός αλλά αργός τρόπος ταξινόμησης. Υπάρχουν πολλές μέθοδοι ταξινόμησης, η κάθε μια με τα πλεονεκτήματα και τα μειονεκτήματα της. Ενδεικτικά αναφέρονται η ταξινόμηση με εισαγωγή, η γρήγορη ταξινόμηση και η ταξινόμηση με συγχώνευση.

Εισαγωγή και διαγραφή

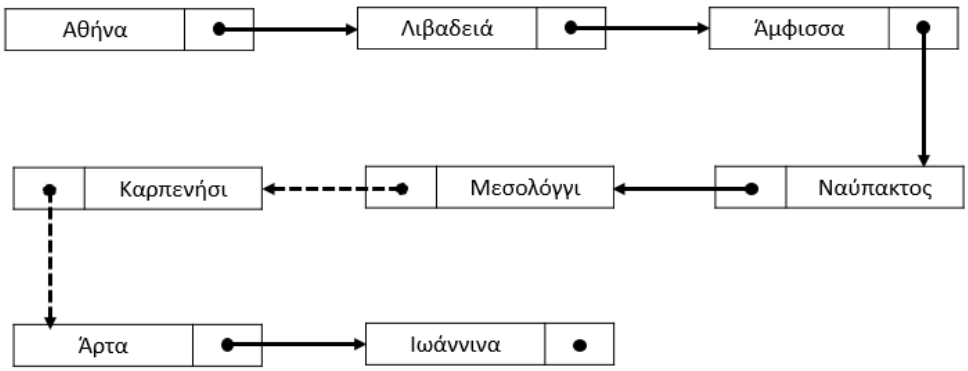
Στις δυναμικές δομές δεδομένων έχουμε τη δυνατότητα να εισάγουμε ή να διαγράφουμε κόμβους κατά βούληση.

Για παράδειγμα, έστω ότι σχεδιάζουμε ένα ταξίδι στην ηπειρωτική Ελλάδα. Η αφετηρία μας είναι η Αθήνα και το τέρμα του ταξιδιού είναι τα Ιωάννινα. Έχουμε αποθηκεύσει σε μια συνδεδεμένη λίστα τις πόλεις που είναι πιθανό να επισκεφθούμε (Σχήμα 2.2.46).



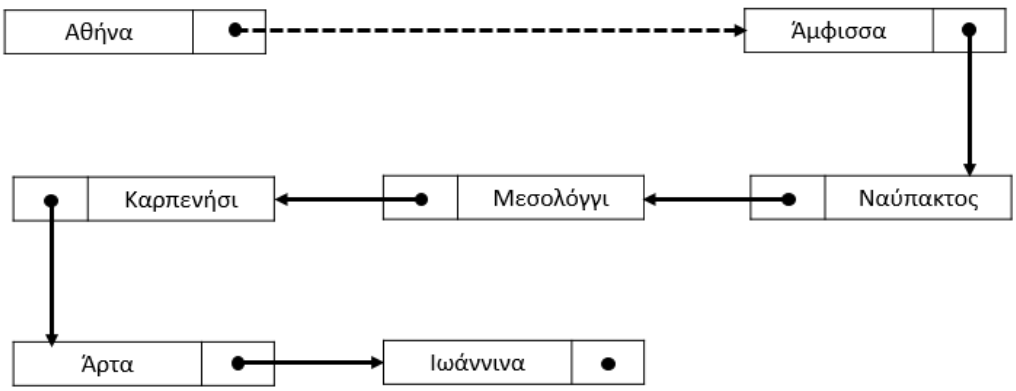
Σχήμα 2.2.46

Λίγο πριν το ταξίδι αποφασίζουμε να επισκεφθούμε το Καρπενήσι, αμέσως μετά το Μεσολόγγι και λίγο πριν φτάσουμε στο προορισμό μας, την Άρτα. Επομένως, χρειάζεται να **εισάγουμε** ένα νέο στοιχείο στη λίστα μας κάνοντας την κατάλληλη διευθέτηση των δεικτών (Σχήμα 2.2.47).



Σχήμα 2.2.47

Αν επιπλέον αποφασίσουμε να μην επισκεφθούμε τη Λιβαδειά θα πρέπει να **διαγράψουμε** το παραπάνω στοιχείο από τη λίστα κάνοντας μια εκ νέου διευθέτηση των αντίστοιχων δεικτών (Σχήμα 2.2.48).



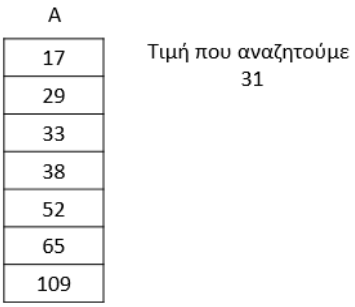
Σχήμα 2.2.48

Ερωτήσεις – Δραστηριότητες

1. Με δεδομένο τον παρακάτω πίνακα A = [13, 89, 21, 44, 99, 56, 9] να δείξετε τα βήματα της ταξινόμησης με επιλογή για τη διάταξη των τιμών του πίνακα σε αύξουσα σειρά, κατασκευάζοντας τα κατάλληλα σχήματα.
2. Εργασία σε ομάδες. Κάθε ομάδα θα αναλάβει να μελετήσει μια από τις παρακάτω μεθόδους ταξινόμησης: ταξινόμηση με εισαγωγή (insertion

sort), ταξινόμηση με συγχώνευση (mergesort), ταξινόμηση ευθείας ανταλλαγής ή φυσαλίδας (bubble sort) και γρήγορη ταξινόμηση (quicksort). Δείξτε τον τρόπο λειτουργίας τους σχηματικά, χρησιμοποιώντας τον πίνακα A = [7, 17, 89, 74, 21, 7, 43, 9, 26, 10]. Μια οπτικοποιημένη εκδοχή των παραπάνω μεθόδων ταξινόμησης μπορείτε να δείτε στη σελίδα http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html.

3. Έστω ότι θέλουμε να εφαρμόσουμε τον αλγόριθμο της σειριακής αναζήτησης σε έναν ταξινομημένο πίνακα. Ποιες τροποποιήσεις μπορούμε να κάνουμε στον τρόπο λειτουργίας του αλγορίθμου, ώστε να τον επιταχύνουμε σε περίπτωση που η τιμή που αναζητούμε δεν υπάρχει στον πίνακα.



.

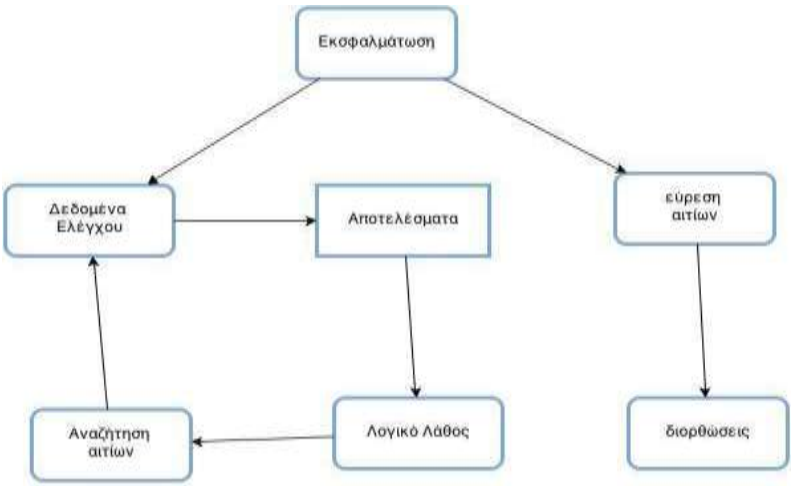
2.2.9. Εκσφαλμάτωση λογικών λαθών

Όταν ένας αλγόριθμος δεν παράγει σωστά αποτελέσματα κατά την εκτέλεσή του, τότε λέμε ότι έχει λογικό λάθος.

Εκσφαλμάτωση *(Debugging) είναι η εύρεση και η διόρθωση των λογικών λαθών ενός αλγορίθμου.*

Η εκσφαλμάτωση γίνεται σε δύο στάδια. Αρχικά, πρέπει να διερευνηθεί αν υπάρχει λογικό λάθος στον αλγόριθμο. Αυτό συμβαίνει όταν τα αποτελέσματα δεν είναι τα αναμενόμενα. Στη συνέχεια πρέπει να προσδιοριστεί σε ποιο σημείο του αλγορίθμου έχει γίνει το λογικό λάθος, ώστε να γίνουν οι κατάλληλες διορθώσεις. Στο Σχήμα 2.2.49 φαίνεται ότι η διαδικασία της εκσφαλμάτωσης αλγορίθμου είναι μια επαναληπτική διαδικασία ελέγχου, με σκοπό την αναζήτηση των αιτιών που δημιουργούν το πρόβλημα. Από τη στιγμή που εντοπιστεί το λάθος η διόρθωση του είναι μια σχετικά εύκολη και γρήγορη διαδικασία.

Η διαδικασία της εκσφαλμάτωσης μπορεί να αποβεί ιδιαίτερα δύσκολη και χρονοβόρα που για να επιτευχθεί χρειάζεται πολύ καλή γνώση της λειτουργίας του αλγορίθμου. Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν εργαλεία που βοηθάνε στην εύρεση των λογικών λαθών, όπως είναι η εκτέλεση του αλγορίθμου βήμα προς βήμα, με παράλληλη παρακολούθηση των τιμών των μεταβλητών.



Σχήμα 2.2.49: Η διαδικασία της εκσφαλμάτωσης.

Παράδειγμα

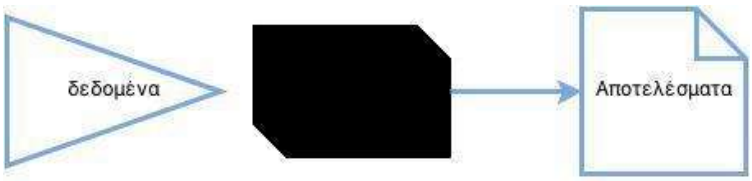
Έστω ο παρακάτω αλγόριθμος που διαβάζει τρεις διαφορετικούς αριθμούς (α, β, γ) και εμφανίζει το μεγαλύτερο από αυτούς.

Να γίνουν οι απαραίτητοι έλεγχοι ώστε: α) να διαπιστωθεί ότι υπάρχει λογικό λάθος β) να εντοπιστεί το μέρος του κώδικα που βρίσκεται το λάθος και γ) να διορθωθεί.

```
Αλγόριθμος μεγαλος_αριθμος
Διάβασε α, β, γ
Αν α > β τότε
        Αν α > γ τότε
          Εμφάνισε 'το', α, 'είναι μεγαλύτερο'
        αλλιώς
          Εμφάνισε 'το', γ, 'είναι μεγαλύτερο'
        Τέλος_αν
αλλιώς
        Αν α > γ τότε
          Εμφάνισε 'το', β, 'είναι μεγαλύτερο'
        αλλιώς
          Εμφάνισε 'το', γ, 'είναι μεγαλύτερο'
        Τέλος_αν
Τέλος_αν
Τέλος
```

Για να διαπιστωθεί αν υπάρχει λογικό λάθος στον αλγόριθμο γίνεται ο **έλεγχος του μαύρου κουτιού** (black box testing). Ο αλγόριθμος αντιμετωπίζεται ως ένα **μαύρο κουτί** που δεν μπορείς να δεις το περιεχόμενό του. Δεν ενδιαφέρει η δομή του, παρά μόνο τα αποτελέσματά του. Εισάγονται δεδομένα και παράγονται

αποτελέσματα τα οποία ελέγχονται για την ορθότητα τους. Τα δεδομένα ελέγχου επιλέγονται, ώστε να καλύπτουν όλες τις δυνατές περιπτώσεις.



Σχήμα 2.2.50: Ο Έλεγχος του Μαύρου Κουτιού

Στην περίπτωσή μας χρησιμοποιούμε τα παρακάτω δεδομένα:

1) α= 1, β=2, γ=3
2) α=3, β =1, γ=2
3) α= 2, β=3, γ =1

Η επιλογή των δεδομένων ελέγχου έγινε ώστε ο μεγαλύτερος αριθμός να είναι σε διαφορετική θέση κάθε φορά.

Από τα αποτελέσματα του ελέγχου διαπιστώνουμε ότι προκύπτει λάθος στην 3η περίπτωση, όταν το β είναι το μεγαλύτερο.

Αφού διαπιστώσαμε ότι υπάρχει λογικό λάθος πρέπει τώρα να προσδιορίσουμε που βρίσκεται αυτό. Αυτό γίνεται με τον έλεγχο του **άσπρου κουτιού** (white box testing). Εξετάζουμε τον αλγόριθμο αναλυτικά και προσπαθούμε να ελέγξουμε αν όλες οι διαδρομές του, παράγουν τα σωστά αποτελέσματα.

Στην περίπτωση μας οι διαδρομές που μπορεί να εκτελέσει ο αλγόριθμος είναι τέσσερις. Για κάθε διαδρομή επιλέγονται τα κατάλληλα δεδομένα ελέγχου όπως φαίνεται στον παρακάτω πίνακα. Η τέταρτη διαδρομή δημιουργεί λανθασμένο αποτέλεσμα και εκεί πρέπει να επικεντρωθεί η προσοχή μας για την επίλυση του σφάλματος.

		α	β	γ	Αποτέλεσμα
1η διαδρομή	α>β και α>γ	20	10	15	20
2η διαδρομή	α>β και γ>α	20	10	30	30
3η διαδρομή	β>α και α>γ	10	20	5	20
4η διαδρομή	**β>α και γ>α**	**10**	**20**	**15**	**15**

Είναι προφανές ότι το λάθος βρίσκεται στο παρακάτω κομμάτι του κώδικα

```
Αλλιώς
        Αν α > γ τότε
                Εμφάνισε 'το', β, 'είναι μεγαλύτερο'
        Αλλιώς
                Εμφάνισε 'το', γ, 'είναι μεγαλύτερο'.
```

Πρέπει να γίνει β > γ για να περιλαμβάνεται η περίπτωση που το β είναι το μεγαλύτερο. Κάνουμε την αλλαγή, ελέγχουμε και διαπιστώνουμε ότι όλα είναι σωστά.

Ερωτήσεις – Δραστηριότητες

1. Γράψτε τους παρακάτω αλγόριθμους χρησιμοποιώντας κατάλληλο προγραμματιστικό περιβάλλον.
2. Χρησιμοποιήστε κατάλληλα δεδομένα ελέγχου για να διαπιστώσετε αν λειτουργούν σωστά. (έλεγχος μαύρου κουτιού)
3. Σε περίπτωση που διαπιστωθεί λογικό λάθος να βρείτε πού ακριβώς είναι αυτό. (έλεγχος άσπρου κουτιού) - Κάντε χρήση των δυνατοτήτων εκσφαλμάτωσης που σας παρέχει το προγραμματιστικό περιβάλλον.
4. Διορθώστε τα λάθη, ώστε οι αλγόριθμοι να εκτελούνται σωστά.

1ος Αλγόριθμος

Εισάγονται 5 αριθμοί από το πληκτρολόγιο. Υπολογίζεται και εμφανίζεται ο μεγαλύτερος από αυτούς καθώς και η σειρά εισαγωγής του.

```
Αλγόριθμος μεγιστο
θεση ← 1
Διάβασε α
μεγ ← α
Για ι από 1 μέχρι 4
  Διάβασε α
  Αν α > μεγ τότε
    μεγ ← α
    θεση ← ι
  Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε 'Σειρά εισαγωγής ', θεση, ' μέγιστο ', μεγ
Τέλος
```

2ος Αλγόριθμος

Δίνονται 5 αριθμοί από το πληκτρολόγιο. Να βρεθεί πόσοι είναι μεγαλύτεροι του 2, πόσοι είναι μεγαλύτεροι του 5 και πόσοι μεγαλύτεροι του 7.

```
Αλγόριθμος μεσος_ορος
κ ← 0
λ ← 0
μ ← 0
Για ι από 1 μέχρι 5
  Διάβασε α
  Αν α > 2 τότε
    κ ← κ + 1
  αλλιώς_αν α > 5 τότε
```

```
   λ ← λ + 1
  αλλιώς_αν α > 7 τότε
   μ ← μ + 1
  Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε 'μεγαλύτεροι του 2 ', κ
Εμφάνισε 'μεγαλύτεροι του 5 ', λ
Εμφάνισε 'μεγαλύτεροι του 7 ', μ
Τέλος
```

2.2.10. Τεκμηρίωση Αλγορίθμου

Τεκμηρίωση *είναι το σύνολο των γραπτών κειμένων που συνοδεύει το κάθε λογισμικό.*

Είναι απαραίτητη γιατί από την μια πλευρά βοηθά το χρήστη να κατανοήσει τη λειτουργία του και από την άλλη είναι η βάση για ενδεχόμενες τροποποιήσεις σε ένα υπάρχον σύστημα.

Η τεκμηρίωση χωρίζεται σε δύο μεγάλες κατηγορίες:

α) τεκμηρίωση που αφορά τους χρήστες και

β) τεχνική τεκμηρίωση.

Η τεκμηρίωση για τους χρήστες είναι το εγχειρίδιο χρήσης ή η διαδικτυακή (online) βοήθεια που συνοδεύει το κάθε λογισμικό. Εκεί περιγράφεται ο τρόπος χρήσης του, όσο πιο αναλυτικά και κατανοητά γίνεται, χωρίς να αναφέρονται τεχνικά θέματα που δεν ενδιαφέρουν τον απλό χρήστη.

Αντίθετα, η τεχνική τεκμηρίωση περιγράφει αναλυτικά όλες τις φάσεις ανάπτυξης του λογισμικού. Καταγράφονται σε ειδικές φόρμες με συστηματικό και καθορισμένο τρόπο η ανάλυση, ο σχεδιασμός, ο κώδικας, τα δεδομένα καθώς και τα αποτελέσματα του ελέγχου. Η τεχνική τεκμηρίωση απευθύνεται σε ειδικούς της πληροφορικής, που ενδιαφέρονται να μελετήσουν τη λειτουργία ενός λογισμικού.

Για κάθε λογισμικό που παράγεται, πρέπει να υπάρχει «**ο φάκελος προγράμματος»,** ο οποίος θα περιέχει όλο το γραπτό υλικό της τεκμηρίωσης του. Ο φάκελος θα είναι διαθέσιμος σε όποιον χρειάζεται να συντηρήσει το συγκεκριμένο λογισμικό.

Επειδή τα παλαιότερα χρόνια ήταν συνηθισμένο φαινόμενο η έλλειψη τεκμηρίωσης σε πολλά λογισμικά, αναπτύχθηκε ένας καινούριος κλάδος της Μηχανικής Λογισμικού που ονομάζεται «**Αντίστροφη Μηχανική**» (Reverse Engineering) και είναι η διαδικασία ανακάλυψης της σχεδίασης ενός λογισμικού, όταν δεν υπάρχει τεκμηρίωση γι' αυτό.

Ερωτήσεις – Δραστηριότητες

1. Αναφέρετε δύο λόγους για τους οποίους είναι απαραίτητη η τεκμηρίωση του λογισμικού;
2. Εργασία σε ομάδες. Βρείτε ένα απλό online παιχνίδι της αρεσκείας σας. Δημιουργήστε έναν οδηγό χρήσης αυτού του παιχνιδιού.

2.3. Προγραμματισμός

2.3.1. Γλώσσες προγραμματισμού και "Προγραμματιστικά Υποδείγματα"

Προγραμματιστικά Υποδείγματα

Η γλώσσα Προγραμματισμού αποτελεί το εργαλείο για την αναπαράσταση ενός αλγορίθμου και είναι εύκολα κατανοητή στους ανθρώπους, ενώ εύκολα επίσης, μετατρέπεται σε γλώσσα μηχανής που είναι κατανοητή από τον υπολογιστή.

Γλώσσα μηχανής *είναι η γλώσσα προγραμματισμού που περιλαμβάνει εντολές γραμμένες σε μορφή ακολουθιών δυαδικών ψηφίων (bit). Οι εντολές αυτές είναι άμεσα εκτελέσιμες από την Κεντρική Μονάδα Επεξεργασίας (CPU). Μέσω αυτής επιτυγχάνεται «επικοινωνία» με τον υπολογιστή.*

Τα πρώτα χρόνια της ζωής των ηλεκτρονικών υπολογιστών η γλώσσα προγραμματισμού αποτέλεσε το κύριο εργαλείο για την ανάπτυξη λογισμικού. Στη συνέχεια, αυτή περιορίστηκε ως εργαλείο της φάσης της υλοποίησης, επηρεάζοντας σημαντικά και τη φάση του σχεδιασμού.

Η ιστορία των γλωσσών προγραμματισμού έχει να παρουσιάσει μια μεγάλη ποικιλία γλωσσών. Ορισμένες από αυτές, όπως οι Fortran, Lisp, Simula και Prolog, εισήγαγαν νέα προγραμματιστικά υποδείγματα, δηλαδή νέους τρόπους σκέψης προγραμματισμού. Άλλες πάλι αποτελούν απογόνους επιτυχημένων γλωσσών όπως η C++ που αποτελεί απόγονο της C.

Τα προγραμματιστικά υποδείγματα εκφράζουν διαφορετικές φιλοσοφίες στον προγραμματισμό.

Προγραμματιστικά υποδείγματα είναι ένα σύνολο από έννοιες, οι οποίες προσδιορίζουν έναν τρόπο σκέψης και έκφρασης της λύσης και επηρεάζουν το σχεδιασμό των προγραμμάτων.

Τα προγραμματιστικά υποδείγματα μπορούν να κατηγοριοποιηθούν ως εξής:

- Προστακτικό ή Διαδικαστικό
- Δηλωτικό Υπόδειγμα
- Αντικειμενοστραφές Υπόδειγμα
- Οπτικό Υπόδειγμα

1. Προστακτικό ή Διαδικαστικό υπόδειγμα

Κατά το προστακτικό (imperative) ή διαδικαστικό (procedural) υπόδειγμα, η διαδικασία προγραμματισμού είναι μια ακολουθία από εντολές, που, όταν εκτελεστούν, ενεργούν πάνω στα δεδομένα και παράγουν το επιθυμητό αποτέλεσμα. Το υπόδειγμα αυτό θεωρεί ότι πρέπει να βρεθεί ο αλγόριθμος για την επίλυση του προβλήματος και έπειτα ο αλγόριθμος να αναπαρασταθεί ώς μια ακολουθία από εντολές. Τα υποδείγματα αυτά είναι τα πιο παραδοσιακό από τα υποδείγματα.

Οι γλώσσες Pascal, Ada, Algol, Fortran και C υποστηρίζουν την προστακτική μορφή προγραμματισμού. Ο προγραμματιστής με τις γλώσσες αυτές προσδιορίζει, βήμα προς βήμα την πορεία επίλυσης του προβλήματος. Χρησιμοποιεί μεταβλητές, για να αναπαραστήσει τα δεδομένα και τις ενέργειες (actions), που αποτελούν τις βασικές μονάδες του προστακτικού στυλ προγραμματισμού, για να αλλάξει τις τιμές των μεταβλητών.

Χαρακτηριστικά του διαδικαστικού υποδείγματος:

- ✓ Τα μεγάλα προγράμματα χωρίζονται σε μικρά τμήματα.
- ✓ Χρησιμοποιεί Top-Down προσέγγιση προγραμματισμού, δηλαδή ξεκινάει από το υψηλότερο εννοιολογικό επίπεδο και προχωράει προς τις λεπτομέρειες
- ✓ Τα δεδομένα κινούνται ελεύθερα από το ένα τμήμα του προγράμματος στο άλλο.
- ✓ Τα περισσότερα τμήματα του προγράμματος μοιράζονται κοινά δεδομένα.
- ✓ Δίνεται έμφαση στους αλγόριθμους.

2. Δηλωτικό υπόδειγμα

Το **δηλωτικό** (declarative) υπόδειγμα, όπου ο προγραμματιστής περιγράφει το προς επίλυση πρόβλημα και όχι τον αλγόριθμο. Το υπόδειγμα αυτό εφαρμόζει ένα γενικό αλγόριθμο επίλυσης προβλημάτων και ο προγραμματιστής πρέπει να περιγράψει κατάλληλα το πρόβλημα. Ο προγραμματιστής περιγράφει τι θέλει να προσομοιωθεί, ο αλγόριθμος πρόγνωσης υπάρχει στην γλώσσα. Με αντιπροσωπευτικη τη γλώσσα Prolog στον τομέα της τεχνητής νοημοσύνης.

Ένα πρόγραμμα αποτελείται από σχέσεις που περιγράφουν τα συστατικά του προς επίλυση προβλήματος και μεταξύ τους συσχετισμούς. Η λύση του προβλήματος προκύπτει από την εφαρμογή του μηχανισμού για εξαγωγή συμπερασμάτων που έχει η γλώσσα υλοποίησης. Το πρόγραμμα τερματίζει όταν βρεθούν όλες οι δυνατές λύσεις του προβλήματος.

3. Αντικειμενοστραφές υπόδειγμα

Μία μορφή προγραμματισμού που τα τελευταία χρόνια γνωρίζει μεγάλη εξάπλωση είναι το **αντικειμενοστραφές** υπόδειγμα. Το αντικειμενοστραφές (object-oriented) υπόδειγμα / Object-Oriented Programming (OOP) είναι κατά

βάση προστακτικό μοντέλο. Οι μονάδες δεδομένων είναι ενεργά αντικείμενα και κάθε αντικείμενο μπορεί να επενεργεί στον εαυτό του ή σε άλλα αντικείμενα. Στα γραφικά προγραμματιστικά περιβάλλοντα (Graphical User Interface) κάθε εικονίδιο αποτελεί ένα αντικείμενο..

Κάθε αντικείμενο περιλαμβάνει μεθόδους (methods), που ορίζουν πως αποκρίνεται σε διάφορα γεγονότα. Στα υποδείγματα αυτά κλάση (class) αντικειμένων είναι το σύνολο ιδιοτήτων που χαρακτηρίζουν ίδια αντικείμενα (που ανήκουν στην ίδια κλάση). Ορίζεται μια κλάση αντικειμένων και έπειτα ορίζονται αντικείμενα της κλάσης. Ο όλος προγραμματισμός βασίζεται σε αντικείμενα (objects) διαφόρων κατηγοριών (classes), τα οποία αλληλεπιδρούν μεταξύ τους. Στην αρχή ορίζονται τα κατάλληλα για το πρόβλημα αντικείμενα και στη συνέχεια χρησιμοποιούνται για να περιγραφεί βήμα προς βήμα η εξεύρεση της λύσης. Το μοντέλο OOP είναι μοντέλο δομημένου προγραμματισμού, όπου ο προγραμματισμός γίνεται χτίζοντας το πρόγραμμα με Μπλοκ. Χαρακτηριστικά παραδείγματα είναι οι γλώσσες C++, Visual Basic, Java, C#.

4. Οπτικό υπόδειγμα

Οπτικός προγραμματισμός -Visual Programming. Παρέχεται ευκολία ανάπτυξης ολοκληρωμένων εφαρμογών. Υπάρχουν έτοιμες βιβλιοθήκες γραφικών που μπορούν να χρησιμοποιηθούν σε διαφορετικές εφαρμογές. Μπορεί να είναι ο ευκολότερος τρόπος για να αρχίζει να προγραμματίζει κάποιος.

Εμπειρία προγραμματισμού

Μέχρι τώρα έχετε έρθει σε επαφή με logo-like προγραμματιστικά περιβάλλοντα. Τα περιβάλλοντα αυτά είναι μια ομάδα γλωσσών/διαλέκτων με ενιαία φιλοσοφία αλλά και με ιδιαίτερες δυνατότητες το καθένα. Αυτά τα προγραμματιστικά περιβάλλοντα διαθέτουν μια απλή, σαφή και ισχυρή γλώσσα προγραμματισμού, που επιτρέπει σύντομο κύκλο ανάπτυξης προγραμμάτων με άμεση εκτέλεση εντολών.

Αξιοποιώντας τα γραφικά της χελώνας έχετε δει τα αποτελέσματα της εκτέλεσης προγραμμάτων με άμεσο τρόπο. Με τα περιβάλλοντα αυτά έχετε έρθει σε επαφή με την αλγοριθμική που μπορεί να εφαρμοστεί μέσα από τον δομημένο προγραμματισμό. Επίσης μπορεί να έχετε έρθει σε επαφή με σύγχρονα logo-like που η διεπαφή με το χρήστη να γίνεται μέσω «Οπτικού Προγραμματισμού με Πλακίδια» όπου ο χρήστης τα συνδυάζει μεταξύ τους όπως τα κομμάτια ενός παζλ (πχ Scratch).

2.3.2. Σχεδίαση και συγγραφή κώδικα

Μέχρι το σημείο αυτό έχει γίνει περιγραφή της λύσης προβλημάτων με τη χρήση αλγορίθμων. Για να μπορέσει ο αλγόριθμος να εκτελεστεί σε ένα ηλεκτρονικό υπολογιστή θα πρέπει να μετατραπεί σε πρόγραμμα.

Το πρόγραμμα λοιπόν είναι μια διαφορετική από τον αλγόριθμο αναπαράσταση και η μετατροπή τους αλγορίθμου σε κώδικα/πρόγραμμα ονομάζεται ***κωδικοποίηση****.*

http://alkisg.mysch.gr/

Ο Διερμηνευτής της ΓΛΩΣΣΑΣ είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης αλγορίθμων σε μορφή ψευδοκώδικα.

Η κωδικοποίηση είναι ο αυστηρός τρόπος περιγραφής ενός αλγορίθμου και αποτελείται από συγκεκριμένες εντολές και υπακούει τους συντακτικούς κανόνες της γλώσσας προγραμματισμού. Όπως αναφέρθηκε στην προηγούμενη ενότητα, υπάρχουν αρκετά και διαφορετικά προγραμματιστικά υποδείγματα. Στο βιβλίο αυτό θα χρησιμοποιηθεί το υπόδειγμα του δομημένου προγραμματισμού και συγκεκριμένα ο **«διερμηνευτής της γλώσσας»** τον οποίο θεωρούμε κατάλληλο για τους σκοπούς του συγκεκριμένου μαθήματος.

Οι διαφορές ανάμεσα στο πρόγραμμα για τη συγκεκριμένη γλώσσα προγραμματισμού και τον αλγόριθμο είναι η αυστηρότητα στη χρήση δεσμευμένων λέξεων και κανόνων και η δήλωση των μεταβλητών που χρησιμοποιούνται

```
ΠΡΟΓΡΑΜΜΑ εμβαδόν_τριγώνου
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: βάση, ύψος
  ΠΡΑΓΜΑΤΙΚΕΣ: εμβαδόν
ΑΡΧΗ
  ΓΡΑΨΕ 'Δώστε τη βάση και το ύψος'
  ΔΙΑΒΑΣΕ βάση, ύψος
  εμβαδόν <- (βάση*ύψος)/2
  ΓΡΑΨΕ "το εμβδαβόν τριγώνου με βάση ", βάση, " και ύψος ", ύψος, " είναι
", εμβαδόν
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

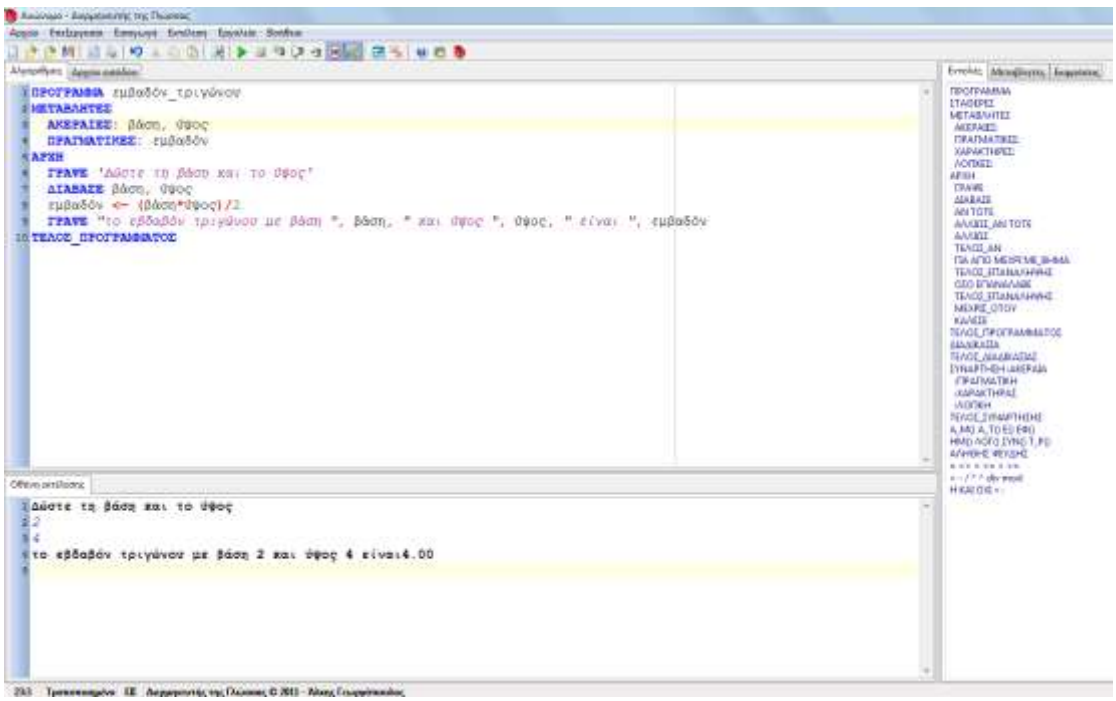
Οι λέξεις που είναι με μπλε χρώμα είναι δεσμευμένες λέξεις για το «διερμηνευτή της γλώσσας». Το πρόγραμμα αυτό έχει το **όνομα** (εμβαδόν_τριγώνου) που περιγράφει το τι κάνει το πρόγραμμα. Είναι σημαντικό ο τίτλος του προγράμματος και τα ονόματα των μεταβλητών, χωρίς να είναι υπερβολικά μεγάλα σε μήκος, να βοηθούν αυτόν που τα διαβάζει να κατανοεί τη χρήση που θα έχουν.

Επίσης είναι σημαντικό, για να είναι ένα πρόγραμμα ευανάγνωστο και ευκολότερα κατανοητό να υπάρχει αυστηρή στοίχιση. Παρατηρήστε ότι οι δηλώσεις των μεταβλητών είναι ένα επίπεδο μέσα από τη λέξη ΜΕΤΑΒΛΗΤΕΣ και οι εντολές του κυρίως προγράμματος είναι ένα επίπεδο μέσα από το ΑΡΧΗ και το ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Αυτό μπορεί να μην φαίνεται απαραίτητο λόγω του πολύ μικρού μεγέθους του συγκεκριμένου προγράμματος, αλλά βοηθάει σημαντικά στην ανάγνωση σε μεγαλύτερα προγράμματα.

Ακολουθεί το **τμήμα δήλωσης μεταβλητών** όπου δηλώνονται όλες οι μεταβλητές που θα χρησιμοποιηθούν και το είδος των δεδομένων που θα πάρουν. Θεωρούμε ότι θα δοθούν ακέραιες τιμές για τη βάση και το ύψος του τριγώνου, ενώ το εμβαδόν που προκύπτει από τη διαίρεση (βάση*ύψος)/2 θα είναι γενικά πραγματικός, οπότε δηλώνεται ως τέτοιος. Στη συνέχεια ξεκινάει το πρόγραμμα

με τη δεσμευμένη λέξη ΑΡΧΗ και ολοκληρώνεται με το ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Ανάμεσα στα δυο αυτά υπάρχουν οι εντολές του προγράμματος που για λόγους ευκολίας στο συγκεκριμένο προγραμματιστικό περιβάλλον βρίσκονται στη δεξιά μεριά της οθόνης και μπορούν να περάσουν στο πρόγραμμα με διπλό κλικ.

Το περιβάλλον αυτό εμφανίζει, επίσης, οθόνη εκτέλεσης ώστε μπορεί ο χρήστης να βλέπει ταυτόχρονα και το πρόγραμμα και την εκτέλεση του (Σχήμα 2.3.1).



Σχήμα 2.3.1 Εκτέλεση του προγράμματος

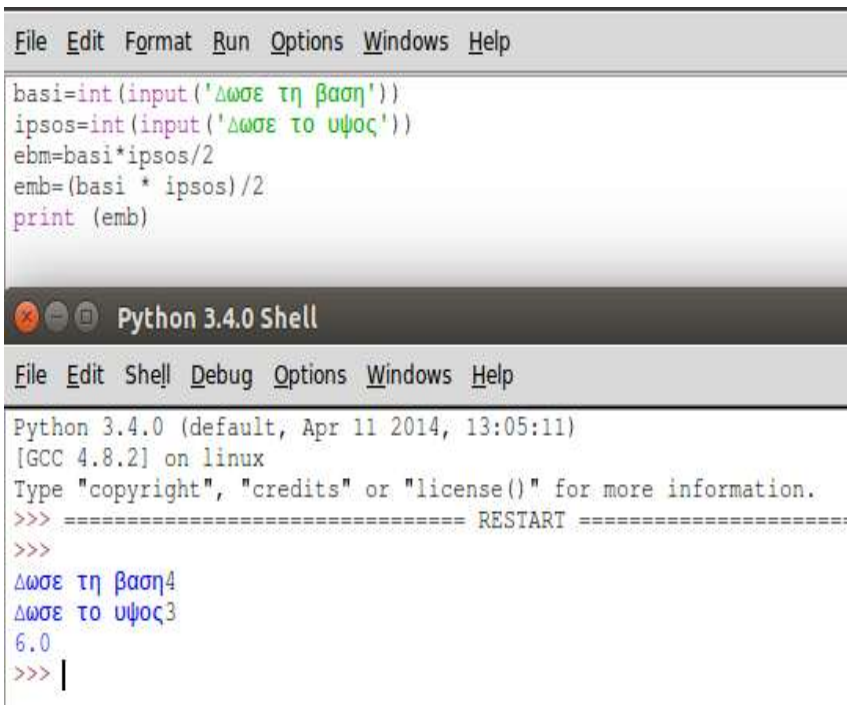
Πέρα από τις δεσμευμένες λέξεις που αντιγράφονται από το πλαίσιο εντολών μπορούν να γίνουν συντακτικά λάθη κατά την πληκτρολόγηση του προγράμματος. Τα λάθη αυτά εντοπίζονται κατά την εκτέλεση του προγράμματος από το διερμηνευτή της γλώσσας και εμφανίζονται τα κατάλληλα μηνύματα ώστε να βοηθήσουν τον προγραμματιστή να κάνει τις απαραίτητες διορθώσεις.

Μερικά συντακτικά λάθη μπορεί να είναι αν στην εντολή που υπολογίζεται και καταχωρείτε το εμβαδόν γραφεί *εμβαδόν= (βάση*ύψος)/2* ο διερμηνευτής εμφανίζει το μήνυμα *Περίμενα το σύμβολο ανάθεσης τιμής* στην οθόνη εκτέλεσης που είναι το ←. Αν δεν υπάρχει η λέξη ΑΡΧΗ μετά τη δήλωση των μεταβλητών εμφανίζει παρόμοιο μήνυμα *Περίμενα τη δεσμευμένη λέξη «ΑΡΧΗ» αλλά βρήκα «ΓΡΑΨΕ»: δεσμευμένη λέξη.* Επίσης αν πατηθεί δυο φορές η λέξη ΑΡΧΗ ο διερμηνευτής δεν την αναγνωρίζει ως δεσμευμένη λέξη και εμφανίζει το αντίστοιχο μήνυμα *Περίμενα τη δεσμευμένη λέξη «ΑΡΧΗ» αλλά βρήκα «ΑΡΧΗΑΡΧΗ»*: Άγνωστο αναγνωριστικό. Ο διερμηνευτής κατά την μετάφραση του προγράμματος που

έχουμε γράψει σε γλώσσα που κατανοεί και μπορεί να εκτελέσει ο υπολογιστής, εντοπίζει τα συντακτικά λάθη και με κατάλληλα μηνύματα προτρέπει στη διόρθωσή τους.

ηγαίος κώδικας: ειρά από εντολές ραμμένες σε μια γλώσσα γραμματισμού. Ο ηγαίος κώδικας μετατρέπεται σε τελέσιμη μορφή.

Η Python είναι μια γλώσσα προγραμματισμού που συνδυάζει σημαντική ισχύ με πολύ σαφή σύνταξη. Οι διερμηνευτές (interpreters) της Python είναι διαθέσιμοι για όλα τα λειτουργικά συστήματα και ο πηγαίος κώδικας (source code) της Python διατίθεται δωρεάν. Η γλώσσα διαθέτει μια κομψή και σχετικά απλοποιημένη σύνταξη, είναι εύκολη στη εκμάθηση για τους αρχάριους και ταυτόχρονα με εξαιρετικές δυνατότητες για τους προχωρημένους. Επιπλέον υπάρχει μεγάλη υποστήριξη από κοινότητες (από τον διεθνή χώρο αλλά και από την Ελλάδα). Επιλέχθηκε λοιπόν αυτή γλώσσα για να δείτε ένα παράδειγμα προγραμματισμού. Το παραπάνω πρόγραμμα γραμμένο σε Python (Σχήμα 2.3.2).



Σχήμα 2.3.2:
Παρατηρήστε και συζητήστε τις διαφορές ανάμεσα στη ΓΛΩΣΣΑ και στην Python.

Τεχνικές επαναχρησιμοποίησης κώδικα

Πολλές φορές μια εργασία σε ένα πρόγραμμα χρειάζεται να επαναλαμβάνεται αρκετές φορές. Για να αποφευχθεί η συγγραφή μεγάλων σε μέγεθος προγραμμάτων, χρησιμοποιείται η τεχνική του **τμηματικού προγραμματισμού**. Τα επαναλαμβανόμενα τμήματα μπορούν να γραφούν μια φορά ως ανεξάρτητες ενότητες και να κληθούν όπου και όσες φορές χρειάζεται μέσα στο κυρίως πρόγραμμα. Στον τμηματικό προγραμματισμό δημιουργούνται προγράμματα από απλούστερα τμή-

ματα προγραμμάτων. Η τεχνική αυτή χρησιμοποιείται συχνά στο δομημένο προγραμματισμό. Με τον τρόπο αυτό τα προγράμματα είναι πιο απλά, ευανάγνωστα και ταυτόχρονα εύκολα στη δημιουργία και τη συντήρηση τους.

Αν θεωρήσουμε ότι έχουμε να βρούμε το μεγαλύτερο από 5 αριθμούς, αν γράφαμε κώδικα που να βρίσκει το μεγαλύτερο από 5 αριθμούς, θα ήταν αρκετά πολύπλοκο και θα ήταν άχρηστο για 4 ή 7 αριθμούς. Είναι δυνατό όμως, να γραφεί ένα απλό τμήμα προγράμματος που να υπολογίζει το μεγαλύτερο από 2 αριθμούς και να κληθεί όσες φορές χρειάζεται ώστε να τους συγκρίνει όλους. Το πρόγραμμα αυτό θα συγκρίνει τον πρώτο με το δεύτερο και θα βγάλει το μεγαλύτερο. Μετά θα συγκρίνει το μεγαλύτερο με τον τρίτο και θα βγάλει το μεγαλύτερο κ.ο.κ. *Αν οι αριθμοί είναι 10 , πόσες φορές θα εκτελεστεί το πρόγραμμα;*

Αν ένα τμήμα προγράμματος έχει γενική χρήση σε πολλά προγράμματα, τότε αυτό μπορεί να μπει σε κάποια βιβλιοθήκη (library).

*Λέγοντας **βιβλιοθήκη** στην Πληροφορική εννοούμε μια συλλογή από έτοιμα υποπρογράμματα που χρησιμοποιούνται στον δομημένο προγραμματισμό.*

Οι βιβλιοθήκες περιέχουν βοηθητικό κώδικα και δεδομένα παρέχοντας με αυτόν τον τρόπο έτοιμες υπηρεσίες σε προγράμματα. Αυτό επιτρέπει το διαμοιρασμό και τη χρήση του κώδικα και των δεδομένων. Οι βιβλιοθήκες μπορεί να περιέχουν ομάδες προγραμμάτων για παρόμοιες εργασίες. Σε κάθε πρόγραμμα που χρειαζόμαστε μια ομάδα εργασιών μπορούμε να καλούμε την αντίστοιχη βιβλιοθήκη και να χρησιμοποιούμε όσα προγράμματα χρειαζόμαστε από αυτή. Θα μπορούσε να υπάρχει μια βιβλιοθήκη με στατιστικές συναρτήσεις ή μια βιβλιοθήκη για μαθηματικά προβλήματα της Β' λυκείου.

2.3.3. Ο κύκλος ζωής εφαρμογής λογισμικού

Εισαγωγή

Κατά τη διάρκεια των πρώτων δεκαετιών ανάπτυξης της Πληροφορικής, η δημιουργία λογισμικού θεωρούνταν τέχνη και όχι μια διαδικασία που όφειλε να ακολουθεί συγκεκριμένους κανόνες.

Αποτέλεσμα αυτής της νοοτροπίας ήταν να υπάρχουν πολλά λογισμικά που δεν ανταποκρίνονταν στις απαιτήσεις των χρηστών και παράλληλα ήταν πολυδάπανα στη συντήρησή τους. Επειδή δε ο τρόπος ανάπτυξής τους βασιζόταν στην ευρηματικότητα και στους πειραματισμούς του κάθε προγραμματιστή, ήταν σχεδόν αδύνατο να διαπιστωθούν οι αιτίες των προβλημάτων τους.

Στα τέλη της δεκαετίας του ΄60 ήταν πλέον αναγκαιότητα η ανάπτυξη μιας δομημένης διαδικασίας για τη δημιουργία των λογισμικών. Έτσι άρχισε να δημιουργείται **ο κύκλος ζωής εφαρμογής λογισμικού,** ο οποίος παρέχει ένα συγκροτημένο τρόπο δημιουργίας για το κάθε λογισμικό.

Φάσεις του κύκλου ζωής εφαρμογής λογισμικού

Ο κύκλος ζωής εφαρμογής λογισμικού αποτελείται από πέντε διαδοχικές φάσεις (Σχήμα 2.3.3):

1. Ανάλυση
2. Σχεδίαση
3. Υλοποίηση
4. Έλεγχος
5. Λειτουργία και Συντήρηση.

Σχήμα 2.3.3: Φάσεις κύκλου ζωής λογισμικού

Ανάλυση

Σε αυτή τη φάση γίνεται ο λεπτομερής καθορισμός των απαιτήσεων που πρέπει να ικανοποιεί το νέο λογισμικό. Είναι η διαδικασία συγκέντρωσης και συσχετισμού όλων των δεδομένων του. Ο προσδιορισμός των απαιτήσεων του χρήστη πρέπει να γίνει με μεγάλη προσοχή διότι θα είναι πολύ δαπανηρό σε χρόνο και σε χρήμα να διορθωθεί κάποιο λάθος σε μεταγενέστερη φάση. Η ανάλυση απαντά στο ερώτημα «*Τι πρέπει να γίνει;*» και όχι στο «*Πως θα γίνει*;».

Σχεδίαση

Σχεδίαση είναι η δημιουργία ενός πλάνου για το νέο λογισμικό με σκοπό την ικανοποίηση των απαιτήσεων του χρήστη. Με τη σχεδίαση παρουσιάζονται οι λειτουργίες του προγράμματος με τέτοιο τρόπο, ώστε να είναι εφικτή η μετατροπή τους σε κώδικα, όπως αντίστοιχα συμβαίνει στην εκπόνηση μελέτης από έναν πολιτικό μηχανικό για την κατασκευή ενός οικήματος.

Υλοποίηση

Αφού έχει προηγηθεί η σχεδίαση του λογισμικού πρέπει να επιλεγεί η γλώσσα προγραμματισμού με την οποία θα γίνει η υλοποίηση του. Η επιλογή γίνεται με τη συνεργασία σχεδιαστή και προγραμματιστή. Επιλέγεται μια γλώσσα που θα επιτρέπει και θα διευκολύνει την ανάπτυξη της εφαρμογής σε συγκεκριμένο περιβάλλον. Εφόσον το σχέδιο είναι ρεαλιστικό η φάση της κωδικοποίησης είναι σχετικά γρήγορη. Η υλοποίηση μπορεί να χωριστεί σε επιμέρους φάσεις όπως η χρήση ψευδοκώδικα, η πληκτρολόγηση του κώδικα στον υπολογιστή, η χρήση έτοιμων υποπρογραμμάτων, ο έλεγχος για συντακτικά λάθη και τέλος η συνένωση όλων των τμημάτων και η δημιουργία του τελικού προϊόντος.

Έλεγχος

Σκοπός αυτής της φάσης είναι να επιβεβαιωθεί ότι το λογισμικό είναι σύμφωνο με τις απαιτήσεις που έχουν τεθεί σε προηγούμενο στάδιο. Έλεγχος είναι η διαδικασία εύρεσης λαθών και διόρθωσής τους. Υπάρχουν δύο στρατηγικές: α) ο

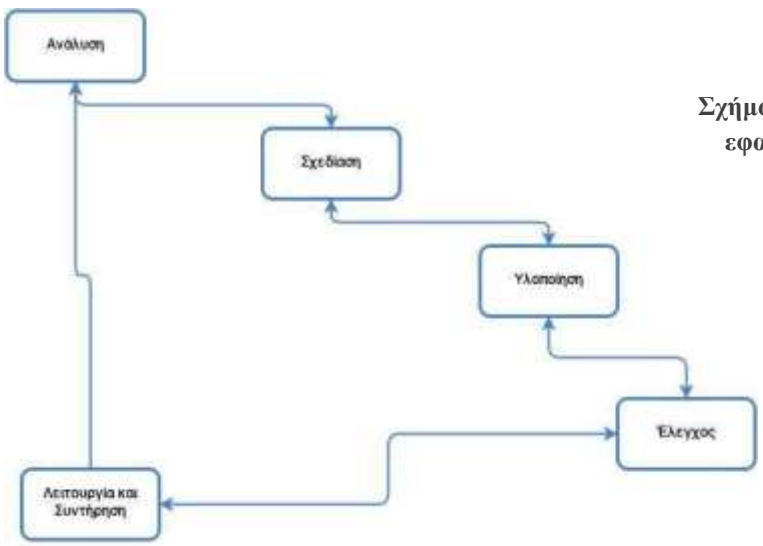
έλεγχος για λογικά λάθη στο πρόγραμμα και β) ο έλεγχος για το άν το πρόγραμμα ικανοποιεί τις απαιτήσεις του χρήστη.

Ο προγραμματιστής θα ελέγξει τον κώδικα για λογικά λάθη, ενώ ο αναλυτής ο οποίος γνωρίζει τις ανάγκες του τελικού χρήστη θα εξετάσει αν το πρόγραμμα που δημιουργήθηκε τις ικανοποιεί.

Αρχικά ελέγχεται κάθε τμήμα του προγράμματος αυτόνομα και στη συνέχεια όλο το πρόγραμμα αφού ενωθούν τα τμήματα μεταξύ τους. Διενεργείται έλεγχος με πραγματικά δεδομένα, αλλά και με λανθασμένα δεδομένα, για να διαπιστωθεί ότι το σύστημα ανταποκρίνεται σε κάθε περίπτωση. Επίσης, ελέγχεται η συμπεριφορά του σε συνθήκες υπερφόρτωσης. Τέλος γίνεται πιλοτική εγκατάσταση του προγράμματος σε υπολογιστές του τελικού χρήστη. Για αρκετό χρονικό διάστημα οι χρήστες σε συνεργασία με τον αναλυτή δοκιμάζουν το νέο λογισμικό καταγράφοντας προβλήματα και βελτιώσεις που μπορούν να γίνουν πριν την τελική του παράδοση.

Λειτουργία και Συντήρηση

Πρόκειται για τη φάση με τη μεγαλύτερη χρονική διάρκεια και το μεγαλύτερο κόστος, αφού αποτελεί το 60% του κύκλου ζωής ενός λογισμικού. Στη φάση αυτή ερευνώνται τυχόν προβλήματα που προέκυψαν από τη χρήση του λογισμικού ή γίνονται απαραίτητες βελτιώσεις που απαιτεί ο χρήστης.



Σχήμα 2.3.4. Ο κύκλος ζωής εφαρμογής λογισμικού.

Εδώ θα πρέπει να σημειωθεί ότι υπάρχει η πιθανότητα επιστροφής σε προηγούμενη φάση του κύκλου, είτε λόγω προβλημάτων, που μπορεί να ανακύψουν, είτε λόγω ενδεχόμενων παραλείψεων, που θα εντοπιστούν. Παράλληλα, από την τελική φάση (Λειτουργία και Συντήρηση) είναι δυνατόν να γίνει επιστροφή στην αρχική (Ανάλυση), όταν κριθεί απαραίτητο να παραχθεί μια καινούρια έκδοση του λογισμικού, που θα αποτελεί βελτίωση της προηγούμενης. (Σχήμα 2.3.4)

Κριτική στη θεωρία του κύκλου ζωής εφαρμογής λογισμικού

Το πλεονέκτημα του κύκλου ζωής λογισμικού είναι ότι υπάρχει σαφής διαχωρισμός του έργου σε απλούστερες φάσεις. Η κάθε φάση παράγει ένα συγκεκριμένο αποτέλεσμα που θα χρησιμοποιηθεί στην επόμενη φάση διευκολύνοντας τη διαχείριση του έργου.

Μειονέκτημα του μοντέλου είναι ότι ο πελάτης βλέπει το λογισμικό αφού αυτό έχει ολοκληρωθεί. Είναι πολύ πιθανό αυτό να μην τον ικανοποιεί, οπότε η διαδικασία να χρειάζεται να ξεκινήσει πάλι από την αρχή. Το πρόβλημα αυτό έρχεται να αντιμετωπίσει «**η εξελικτική ανάπτυξη του λογισμικού**», μια νεότερη θεωρία ανάπτυξης λογισμικού, που συμπληρώνει την θεωρία που περιγράφηκε. Κύριο στοιχείο της νέας προσέγγισης είναι ότι ο δημιουργός του λογισμικού και ο πελάτης έχουν συνεχή συνεργασία. Ο πελάτης βλέπει σε τακτά χρονικά διαστήματα το παραγόμενο έργο διατυπώνοντας τις παρατηρήσεις του, οι οποίες ενσωματώνονται σταδιακά στο νέο λογισμικό. Ο πελάτης συμμετέχει στην ανάπτυξη του νέου λογισμικού, με αποτέλεσμα το τελικό προϊόν να ανταποκρίνεται πλήρως στις απαιτήσεις του.

Λογισμικό Ανοικτού Κώδικα (Λ.Α.Κ.)

Μια διαφορετική προσέγγιση στην ανάπτυξη λογισμικού που ανατρέπει τις παραδοσιακές αντιλήψεις είναι το λογισμικό ανοικτού κώδικα.

Λ.Α.Κ*. είναι το λογισμικό που ο πηγαίος του κώδικας είναι διαθέσιμος σε οποιονδήποτε θέλει να τον μελετήσει, να τον αντιγράψει και να τον αλλάξει.*

Αυτό έρχεται σε αντίθεση με τα «*κλειστά λογισμικά*», όπου ο πηγαίος τους κώδικας είναι στην αποκλειστική διάθεση της εταιρείας παραγωγής. Όπως είναι φανερό, το Λ.Α.Κ. ενισχύει την συνεργασία μεταξύ των κοινοτήτων των προγραμματιστών προσφέροντας ένα συμμετοχικό τρόπο ανάπτυξης λογισμικού.

Το μεγάλο πλεονέκτημα των Λ.Α.Κ. είναι ότι τα περισσότερα προσφέρονται δωρεάν στους χρήστες. Οι χρήστες των Λ.Α.Κ. έχουν στη διάθεσή τους ελκυστικά και αξιόπιστα λογισμικά χωρίς κόστος απόκτησης, αλλά και χωρίς το κόστος των συνεχών αναβαθμίσεων που απαιτούν τα κλειστά λογισμικά. Ο Web Server Apache, το Libre Office, το Λειτουργικό Σύστημα Linux, ο φυλλομετρητής Mozilla Firefox, η βάση δεδομένων MySQL είναι παραδείγματα Λ.Α.Κ. που χρησιμοποιούνται με επιτυχία για πολλά χρόνια από χιλιάδες χρήστες.

Τα Λ.Α.Κ. έχουν κατηγορηθεί ότι ο έλεγχος τους είναι πλημμελής και δεν υπάρχει επαρκής τεκμηρίωση, όμως πλέον οι κοινότητες των Λ.Α.Κ. είναι τεράστιες και υπάρχει άμεση ανταπόκριση σε ερωτήματα και αιτήματα χρηστών. Τέλος, παρουσιάζονται προβλήματα και στην ασφάλεια των Λ.Α.Κ., αφού οι επίδοξοι εισβολείς (hackers) γνωρίζουν τον κώδικα και εντοπίζουν τις αδυναμίες του.

Ερωτήσεις – Δραστηριότητες

1. Ποιο το μειονέκτημα του «κύκλου ζωής εφαρμογής λογισμικού»; Πως το αντιμετωπίζει η «εξελικτική ανάπτυξη λογισμικού»;

2. Να περιγράψετε τις φάσεις για την ανάπτυξη ενός πληροφοριακού συστήματος που προορίζεται για την μηχανοργάνωση ενός σχολείου.
3. Γιατί νομίζετε ότι πολλές γνωστές εταιρίες λογισμικού συμμετέχουν στη δημιουργία λογισμικών ανοιχτού κώδικα τα οποία και προωθούν δωρεάν στην αγορά;
4. <u>Εργασία σε ομάδες:</u>
    1. Αφού χωριστείτε σε ομάδες 2-3 ατόμων δημιουργήστε στο προγραμματιστικό περιβάλλον του υπολογιστή σας ένα απλό πρόγραμμα της αρεσκείας σας.
    2. Αφού βεβαιωθείτε ότι το πρόγραμμά σας είναι σωστό, δημιουργήστε το εκτελέσιμο αρχείο του και προωθήστε το σε μια άλλη ομάδα.
    3. Εκτελέστε τα αρχεία που λάβατε. Ποιο πρόβλημα παρουσιάζεται; Συζήτηση.
    4. Στη συνέχεια προωθήστε στην ίδια ομάδα τον πηγαίο κώδικα του προγράμματος που φτιάξατε.
    5. Γιατί είναι προτιμότερο για κάποιον χρήστη να έχει στη διάθεση του τον πηγαίο κώδικα ενός λογισμικού παρά το εκτελέσιμο αρχείο;
    Συζήτηση.

Στην πληροφορική, ένα εκτελέσιμο αρχείο προκαλεί σ'έναν υπολογιστή την εκτέλεση εργασιών σύμφωνα με κωδικοποιημένες οδηγίες τις οποίες δεν μπορεί να δει ο χρήστης.

3. ΘΕΜΑΤΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Στο κεφάλαιο αυτό εξετάζονται θέματα εφαρμοσμένης επιστήμης υπολογιστών με έμφαση στα Λειτουργικά Συστήματα, στα Πληροφοριακά Συστήματα, στα Δίκτυα Η/Υ και στην Τεχνητή νοημοσύνη.

Στόχοι:

Οι μαθητές μετά την ολοκλήρωση του κεφαλαίου θα μάθουν:

- ✓ Να εντάσσουν τις γνώσεις τους για τα Λειτουργικά Συστήματα στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.
- ✓ Να εντάσσουν τις γνώσεις τους για θέματα σχετικά με τη διαχείριση δεδομένων, δημιουργία, αποθήκευση και ανάκτηση πληροφορίας στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.
- ✓ Να αιτιολογούν ότι τα δεδομένα αποθηκεύονται σε οργανωμένες δομές και ότι αυτά ανακτώνται μέσω συγκεκριμένων συστημάτων και μεθοδολογιών.
- ✓ Να εντάσσουν τις γνώσεις τους για θέματα επικοινωνίας και δικτύωσης συστημάτων στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών.
- ✓ Να οργανώνουν σε νοητικό μοντέλο τα βασικά θέματα που αφορούν τα δίκτυα επικοινωνίας.
- ✓ Να εντάσσουν την Τεχνητή Νοημοσύνη στο σχήμα της Εφαρμοσμένης Επιστήμης των Υπολογιστών, να γνωρίσουν τις επιστημονικές περιοχές της εφαρμογής της Τεχνητής Νοημοσύνης να μπορούν να αναφέρουν τομείς στους οποίους έχει εφαρμογή η Τεχνητή Νοημοσύνη.

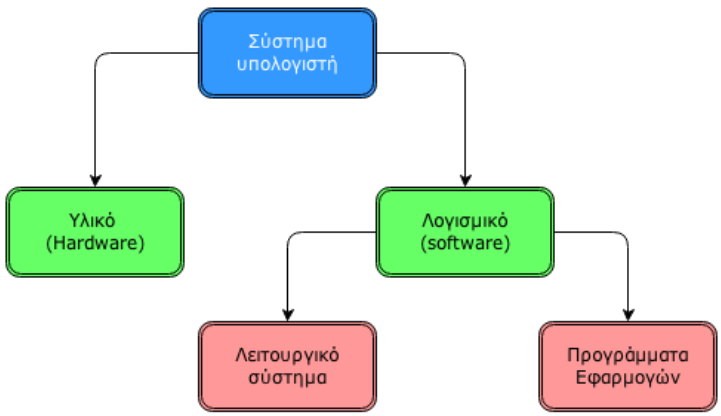
Όροι:

Λειτουργικό σύστημα, Σύστημα Διαχείρισης Βάσης Δεδομένων, Πληροφοριακό Σύστημα, Τοπικά δίκτυα, Μητροπολιτικά δίκτυα, Δίκτυα ευρείας περιοχής, Δρομολογητής, Μεταγωγέας

3.1. Λειτουργικά Συστήματα

Τι είναι τα Λειτουργικά Συστήματα

Όπως γνωρίζουμε, ένα σύστημα υπολογιστή είναι ένας συνδυασμός υλικού και λογισμικού. Υλικό (hardware) είναι το σύνολο συσκευών-εξαρτημάτων που αποτελούν τον υπολογιστή, ενώ το λογισμικό (software) είναι το σύνολο των προγραμμάτων που απαραίτητα τον συνοδεύουν (Σχήμα 3.1). Ο συνδυασμός αυτός είναι απαραίτητος προκειμένου να λειτουργήσει ο οποιοσδήποτε υπολογιστής. Χωρίς το λογισμικό οι πόροι του υλικού του υπολογιστή (επεξεργαστής, κύρια μνήμη, μονάδες αποθήκευσης, περιφερειακές μονάδες κλπ), είναι ουσιαστικά αχρησιμοποίητοι.

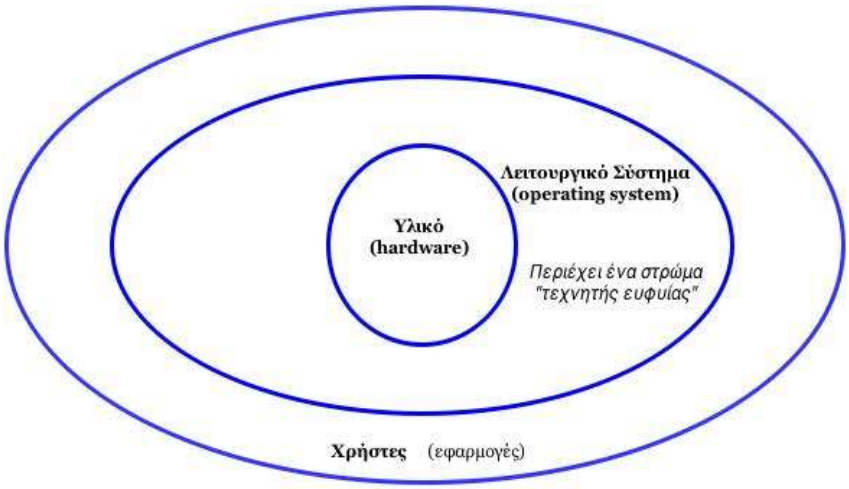


Σχήμα 3.1 Συνιστώσες Υπολογιστικού συστήματος

Το λογισμικό διακρίνεται σε δύο βασικές κατηγορίες: Τα **λειτουργικά συστήματα (ή λογισμικά συστήματος)** και τα **προγράμματα (ή λογισμικό) εφαρμογών.** Τα *προγράμματα εφαρμογών* περιλαμβάνουν τα λογισμικά συγκεκριμένου σκοπού, τα οποία προσδιορίζουν τον τρόπο με τον οποίο οι πόροι του υπολογιστή θα χρησιμοποιηθούν για το σκοπό αυτό (πχ επεξεργαστές κειμένου, φυλλομετρητές ιστού, παιχνίδια κλπ). Αντιθέτως, τα *λειτουργικά συστήματα* έχουν ένα επιτελικό ρόλο:

1. Παρέχουν στον χρήστη την δυνατότητα να χρησιμοποιήσει τους πόρους του υπολογιστή με εύκολο και «διάφανο» τρόπο.
2. Είναι υπεύθυνα για την κατανομή των πόρων του υπολογιστή στους χρήστες, κάνοντας χρήση κάποιας συγκεκριμένης πολιτικής κατανομής τους.
3. Εξασφαλίζουν την προστασία των πόρων και των προγραμμάτων των χρηστών από πιθανά προβλήματα, όπως σφάλματα κατά την εκτέλεση, προσπέλαση μνήμης που ήδη χρησιμοποιείται από άλλον πόρο ή χρηστη και άλλα.

Το λειτουργικό σύστημα είναι το σύνολο των προγραμμάτων που είναι υπεύθυνο για την διαχείριση και τον συντονισμό των εργασιών σε ένα υπολογιστή, καθώς και την κατανομή των διαθέσιμων πόρων του.



Σχήμα 3.2: Σχέση λειτουργικού συστήματος και χρηστών

Τα λειτουργικά συστήματα προγραμματίζονται αποκλειστικά σε συμβολική γλώσσα ή με συνδυασμό ειδικών γλωσσών υψηλού επιπέδου (πχ C/C++). Δημιουργοί μπορεί να είναι εταιρίες κατασκευής υπολογιστών (Apple), ανεξάρτητες εταιρίες λογισμικού (Microsoft), αλλά και ομάδες προγραμματιστών που διαθέτουν εντελώς δωρεάν τα λειτουργικά συτήματα που κατασκευάζουν (πχ διανομές UNIX).

Βασικές εργασίες Λειτουργικών Συστημάτων

Ουσιαστικά λοιπόν το λειτουργικό σύστημα παίζει το ρόλο του ενδιάμεσου μεταξύ των προγραμμάτων εφαρμογών και των πόρων του υπολογιστή. Για το σκοπό αυτό το λειτουργικό σύστημα αναλαμβάνει την διαχείριση της κεντρικής μονάδας επεξεργασίας, της μνήμης, του συστήματος αρχείων, αλλά και των μονάδων εισόδου/εξόδου.

- ✓ *Διαχείριση Κεντρικής Μονάδας Επεξεργασίας:*

Όταν εκτελείται στον υπολογιστή ένα οποιοδήποτε πρόγραμμα, αυτό επιτελεί διάφορες λειτουργίες, οί οποίες ονομάζονται *διεργασίες*. Η αρχιτεκτονική ενός τυπικού υπολογιστή είναι τέτοια, που να επιτρέπει να εκτελείται μόνο μία διεργασία κάθε χρονική στιγμή. Παρόλα αυτά η «αίσθηση» που δημιουργείται σε ένα χρήστη μιας εφαρμογής, είναι ότι το πρόγραμμα μπορεί να εκτελεί παράλληλα διάφορες διεργασίες (πχ πληκτρολόγηση και διαμόρφωση κειμένου, ενώ ταυτόχρονα γίνεται εκτύπωση αλλά και αναπαραγωγή μουσικής). Στη πραγματικότητα το λειτουργικό σύστημα είναι υπεύθυνο για τον διαμοιρασμό (εναλλαγή) της ΚΜΕ σε τακτά χρονικά διαστήματα («ψευδοπαράλληλη» επεξεργασία).

Βέβαια, σε συστήματα με πολλαπλούς επεξεργαστές, το λειτουργικό σύστημα φροντίζει για την πραγματικά παράλληλη επεξεργασία διεργασιών (*πολυδιεργασία*).

- *Διαχείριση μνήμης*

Υπάρχουν διάφοροι τύποι μνήμης σε ένα τυπικό υπολογιστή. Κάθε τύπος έχει διαφορετικό σκοπό και χρησιμότητα. Έτσι υπάρχουν οι καταχωρητές, η κρυφή μνήμη, η κύρια μνήμη (RAM) , η δευτερεύουσα μνήμη (σκληροί δίσκοι κλπ). Ο *διαχειριστής μνήμης* είναι το τμήμα του λειτουργικού συστήματος, υπεύθυνο για συντονισμό των διαφόρων τύπων μνήμης του υπολογιστή. Ο *διαχειριστής μνήμης* γνωρίζει ποια τμήματα μνήμης είναι κατειλημμένα, μπορεί να αναθέτει μνήμη σε διεργασίες όταν την χρειάζονται, ανακαλεί μνήμη που απελευθερώνεται από διεργασίες, και διαχειρίζεται την εναλλαγή (μεταφορά) διεργασιών μεταξύ κύριας μνήμης και δίσκου.

- *Διαχείριση συστήματος αρχείων*

Τα αρχεία ως έννοια, για ένα απλό χρήστη είναι το ισοδύναμο ενός εγγράφου, τα οποία παραδοσιακά ήταν αποθηκευμένα σε ειδικούς αποθηκευτικούς χώρους, φοριαμούς ή βιβλιοθήκες. Για τον υπολογιστή ως **αρχείο** (file) ορίζεται ένα σύνολο από δεδομένα τα οποία είναι διαθέσιμα ή απαιτούνται από ένα πρόγραμμα του υπολογιστή. Τα αρχεία είναι αποθηκευμένα στην δευτερεύουσα μνήμη και άρα είναι μόνιμη η αποθήκευσή τους. Αυτό σημαίνει ότι ακόμα και μετά την διακοπή εκτέλεσης του προγράμματος, ή και του υπολογιστή, τα αρχεία εξακολουθούν να είναι αποθηκευμένα. Τα αρχεία είναι οργανωμένα σε ομάδες που καλούνται φάκελοι (directories).

Ο *διαχειριστής συστήματος αρχείων* είναι το τμήμα του λειτουργικού συστήματος που είναι υπεύθυνο για την οργάνωση και διαχείριση των αρχείων έτσι ώστε να μπορεί ο χρήστης να έχει πρόσβαση στη δευτερεύουσα μνήμη, προκειμένου να δημιουργούν, να διαγράφουν, να μετακινούν, να αντιγράφουν αρχεία (ή και φακέλους), καθώς και να δίνουν συμβολικά ονόματα σε αυτά.

- *Διαχείριση μονάδων εισόδου εξόδου*

Κάθε υπολογιστή διαθέτει πολλαπλές μονάδες εισόδου/εξόδου (input/output, I/O). Τέτοιες μονάδες είναι ο εκτυπωτής, ο σαρωτής, η οθόνη, η συσκευή κατάδειξης (ποντίκι, γραφίδα κλπ), οι μονάδες αποθήκευσης και άλλες. Η μεγάλη ποικιλία περιφερειακών συσκευών, συνεπάγεται την αυξημένη πολυπλοκότητα στο τρόπο επικοινωνίας με τον υπολογιστή αλλά και στον προγραμματισμό που απαιτείται για την αξιοποίησή τους.

Το λειτουργικό σύστημα απαλλάσσει το χρήστη από το βάρος της κατανόησης αυτής της πολυπλοκότητας, παρέχοντας σε αυτόν μία «αφηρημένη» μηχανή, με ποικίλες δυνατότητες εισόδου/εξόδου αλλά συγχρόνως με ιδιαίτερη απλότητα στη χρήση τους.

Σύγχρονα λειτουργικά συστήματα

Τα λειτουργικά συστήματα έχουν παρουσιάσει σημαντικότατη εξέλιξη μέσα στην τελευταία εικοσαετία. Η δικτύωση των υπολογιστών και η ραγδαία εξάπλωση του διαδικτύου (σε συνδυασμό με τις απεριόριστες υπηρεσίες που μπορεί να προσφέρει), δημιούργησαν νέα δεδομένα, και οδήγησαν στην τροποποίηση των αρχών

σχεδιασμού των λειτουργικών συστημάτων, προκειμένου να είναι σε θέση να διαχειριστούν και να αξιοποιήσουν δικτυακούς πόρους.

Τα σύγχρονα λειτουργικά συστήματα διακρίνονται για την ανεξαρτησία από δεσμεύσεις του υλικού, φιλικότητα προς τον χρήστη, και δικτυακό προσανατολισμό. Σε επίπεδο προσωπικών υπολογιστών έχουμε κυριαρχία των Windows της Microsoft, αλλά και άλλα λειτουργικά συστήματα όπως MacOS, Ubuntu κλπ. Σε επίπεδο μεγάλων συστημάτων, διακομιστών κλπ επικρατούν εκδόσεις του UNIX αλλά υπάρχουν και οι εκδόσεις Windows Server.

Εικόνα 3.3: Λειτουργικό σύστημα σε έξυπνο κινητό

Η εξέλιξη των κινητών τηλεφώνων οδήγησε σε μια ιδιαίτερη ανάπτυξη των ΛΣ για τις συσκευές αυτές. Καθώς έχουμε μεταβεί πλέον στην εποχή των έξυπνων κινητών (smartphones), εξειδικευμένα λειτουργικά συστήματα όπως το iOS της Apple, το Android της Google, και άλλα λιγότερο δημοφιλή, εκμεταλλεύονται τις ποικίλες δυνατότητες του υλικού ενός σύγχρονου κινητού τηλεφώνου.

Η εξέλιξη των ήδη δημοφιλών υπολογιστικών συσκευών, αλλά και η εμφάνιση στην παγκόσμια αγορά νέων έξυπνων συσκευών (tablets, έξυπνα ρολόγια χειρός κλπ), δημιουργούν συνεχώς νέα δεδομένα και στα λειτουργικά συστήματα που καλούνται να τα υποστηρίξουν.

Ερωτήσεις – Δραστηριότητες

1. Περιγράψτε την βασική λειτουργία ενός λειτουργικού συστήματος.
2. Ποιες είναι οι κύριες εργασίες ενός λειτουργικού συστήματος;
3. Δώστε ένα παράδειγμα προσομοίωσης της λειτουργίας του λειτουργικού συστήματος, από την καθημερινή ζωή.
4. Ένας χρήστης ενός τυπικού υπολογιστή έχει ανοικτές πολλές εφαρμογές όπως επεξεργαστή κειμένου, λογισμικό αναπαραγωγή μουσικής, περιηγητή Ίντερνετ κ.α. Με ποιο τρόπο το λειτουργικό σύστημα διαχειρίζεται τις λειτουργίες τους ώστε να λειτουργούν όλες «παράλληλα»;

Χρήσιμες ιστοσελίδες

- ✓ Online προσομοίωση λειτουργικού συστήματος PC-DOS με παιχνίδια. http://jamesfriend.com.au/pce-js/ibmpc-games/

3.2. Πληροφοριακά συστήματα – Συστήματα διαχείρισης δεδομένων

Το **Πληροφοριακό Σύστημα** είναι ένα οργανωμένο σύνολο το οποίο αποτελείται από έξι στοιχεία:

- Ανθρώπινο δυναμικό (το σύνολο των ανθρώπων που εργάζονται με το πληροφοριακό σύστημα σε διάφορες θέσεις όπως χρήστες, διαχειριστές κ.ά.)
- Διαδικασίες (το σύνολο των οδηγιών για τη χρήση και το συνδυασμό όλων των στοιχείων υποδομής ενός πληροφοριακού συστήματος)
- Βάση Δεδομένων (Database)
- Software (λογισμικό)
- Hardware (υλικός εξοπλισμός)
- Network (δίκτυο)

Ένα ΠΣ βοηθάει στον έλεγχο, στο συντονισμό, στην ανάλυση προβλημάτων, στη λήψη αποφάσεων και στην ανάπτυξη νέων προϊόντων. Κάθε πληροφοριακό σύστημα πρέπει να:

✓ προσδιορίζει, αποδοτικά και αποτελεσματικά, τις ανθρώπινες ανάγκες αυτών που το χρησιμοποιούν και
✓ επεξεργάζεται όλες τις πληροφορίες με αποτέλεσμα την ικανοποίηση αυτών των αναγκών.

Αυτό γίνεται με:

✓ την πιο αποτελεσματική ανάκτηση, αποθήκευση, επεξεργασία, παρουσίαση και διάδοση των πληροφοριών,
✓ την παροχή των απαραίτητων μέσων και του κατάλληλου περιβάλλοντος μάθησης στους εμπλεκόμενους χρήστες, ώστε να βελτιωθεί η αποτελεσματικότητα της διαδικασίας λήψης απόφασης,
✓ την υποστήριξη των διαδικασιών λειτουργίας, ελέγχου και στρατηγικού σχεδιασμού την επιχείρησης ή του οργανισμού.

Ένα ΠΣ σχεδόν πάντα ολοκληρώνει ένα κύκλο ζωής: δημιουργείται, αναπτύσσεται, εξελίσσεται και αποσύρεται. Η ύπαρξή του αρχίζει από τη στιγμή που ο οργανισμός θα αποφασίσει τη δημιουργία του. Στο επόμενο στάδιο προσδιορίζονται οι βασικές απαιτήσεις που θα ικανοποιεί, και σχεδιάζονται οι λειτουργίες που υλοποιούν τις απαιτήσεις αυτές. Έπειτα αρχίζει το στάδιο στο οποίο πραγματοποιείται η ανάπτυξή και η διαρκής εξέλιξή του ώστε να ικανοποιεί τις ανάγκες της επιχείρησης ή του οργανισμού τον οποίο θα εξυπηρετήσει. Τέλος όταν ο οργανισμός αποφασίσει ότι είναι πια αναποτελεσματικό, και μη αποδοτικό, το πληροφοριακό σύστημα αποσύρεται.

Τα πληροφοριακά συστήματα υποστηρίζουν τις ανθρώπινες δραστηριότητες και εστιάζουν σε απαιτήσεις, που αναφέρονται στις σχέσεις του ανθρώπου αλλά και του συστήματος, ή υποσυστημάτων με τις μηχανές.

Αρχιτεκτονικές αποθήκευσης

Στην πληροφορική, η αρχιτεκτονική δεδομένων αποτελείται από τα μοντέλα, τις πολιτικές, τους κανόνες ή τα πρότυπα που διέπουν τα δεδομένα που θα συλλέγονται, καθώς και από τον τρόπο που αποθηκεύονται και οργανώνονται ώστε να αξιοποιηθούν στα συστήματα διαφόρων οργανισμών.

Αρχιτεκτονικές αποθήκευσης βασισμένες σε δίκτυο

Ένα Σύστημα αποθήκευσης δεδομένων (Storage area network- SAN) είναι ένα ειδικό δίκτυο που παρέχει πρόσβαση σε ενοποιημένα δεδομένα. Τα SANs είναι συσκευές αποθήκευσης, όπως οι συστοιχίες δίσκων, βιβλιοθήκες ταινιών και οπτικών jukeboxes, που έχουν πρόσβαση σε εξυπηρετητές (servers) και εμφανίζονται ώς τοπικά συνδεδεμένες συσκευές από το λειτουργικό σύστημα. Το κόστος και η πολυπλοκότητά τους είναι τέτοια, που να επιτρέπουν την ευρύτερη υιοθέτηση τους σε όλες τις επιχειρήσεις.

Αρχιτεκτονικές αποθήκευσης βασισμένες σε σύννεφο

Σ' έναν οργανισμό που υπάρχουν πολλοί εργαζόμενοι χρειάζονται πολλοί υπολογιστές, με αντίστοιχα λογισμικά και χώρο αποθήκευσης. Πολλοί από αυτούς συνεργάζονται και ανταλλάσσουν πληροφορίες ή χρησιμοποιούν κοινά αρχεία. Υπάρχει η δυνατότητα αυτός ο οργανισμός να χρησιμοποιήσει λογισμικά και αποθηκευτικό χώρο που θα ανήκουν σε κάποια άλλη εταιρία, η οποία διαθέτει υπολογιστικά συστήματα μεγάλης κλίμακας προσβάσιμα μέσω του διαδικτύου. Τα μέλη του οργανισμού θα πρέπει να συνδέονται εκεί για να εργαστούν. Το συγκεκριμένο μοντέλο (Σχήμα 3.4) είναι γνωστό ώς υπολογιστικό νέφος (cloud computing).

Σχήμα.3.4.:Οι ποικίλες υπηρεσίες που προσφέρει το «σύννεφο»

Ο όρος "**σύννεφο**" (cloud) πηγάζει από τον τρόπο της αφαιρετικής αναπαράστασης του internet από τα πρώτα χρόνια της διάδοσής του. Τα τελευταία χρόνια ολοένα και πληθαίνουν οι υπηρεσίες οι οποίες προσφέρουν αποθηκευτικό χώρο βασισμένο στην αρχιτεκτονική του "σύννεφου". Στην εποχή μας δημοφιλείς υπηρεσίες αποθήκευσης στο σύννεφο προσφέρουν εταιρίες όπως η Google, Microsoft, Dropbox και άλλες.

Τα πλεονεκτήματα της αρχιτεκτονικής αυτής είναι:

- φθηνή υπολογιστική ισχύς,
- απαλλαγή από διαχειριστικά κόστη ανάπτυξης και συντήρησης ανάλογης υποδομής
- κόστος υπηρεσίας με βάση την χρήση
- πρόσθετες υπηρεσίες όπως αντίγραφα ασφαλείας, ιστορικό αλλαγών, διαμοιρασμό κλπ.

Παρόλ' αυτά, ερωτήματα εγείρονται σχετικά με το θέμα της προστασίας του απορρήτου των αποθηκευμένων δεδομένων και στο κατά πόσο αυτά είναι εκτεθειμένα στους διαχειριστές των υπηρεσιών αυτών. Η απάντηση που δίνεται από τις εταιρίες, έχει να κάνει με την αυτόματη κρυπτογράφηση των δεδομένων που διαχειρίζονται.

Συστήματα Διαχείρισης Δεδομένων

Ένα **σύστημα διαχείρισης βάσης δεδομένων** (ΣΔΒΔ) (Database Management System - DBMS) αποτελείται από το σύνολο δεδομένων και τα προγράμματα πρόσβασης στα δεδομένα αυτά. Το σύνολο των δεδομένων καλείται Βάση Δεδομένων (database). Στόχος του ΣΔΒΔ είναι η εύκολη και γρήγορη χρήση και ανάκτηση των δεδομένων. Η διαχείριση των δεδομένων περιλαμβάνει:

- τον ορισμό δομών για τη αποθήκευση των δεδομένων και
- τις μεθόδους για τη διαχείριση των δεδομένων

Το ΣΔΒΔ, ένα από τα παλαιότερα στοιχεία που σχετίζονται με τους υπολογιστές, είναι ένα πρόγραμμα λογισμικού που έχει σχεδιαστεί ως μέσο για τη διαχείριση όλων των δεδομένων που υπάρχουν εγκατεστημένα σε έναν σκληρό δίσκο του συστήματος ή του δικτύου.

Υπάρχουν αρκετά εργαλεία που χρησιμοποιούνται στη διαχείριση βάσεων δεδομένων, το ΣΔΒΔ διατίθεται στο εμπόριο σε πολλές μορφές. Στα ΣΔΒΔ υπάρχουν μια σειρά δικαιώματα ή προνόμια που μπορεί να αποδοθούν σε κάθε συγκεκριμένο χρήστη. Αυτό σημαίνει ότι είναι δυνατόν να οριστούν ένας ή περισσότεροι διαχειριστές βάσεων δεδομένων, που έχουν τη δυνατότητα ελέγχου κάθε λειτουργίας, καθώς και την παροχή δικαιωμάτων διαχείρισης σε άλλους χρήστες. Αυτή η ευελιξία καθιστά το έργο της επίβλεψης ενός ΣΔΒΔ ευκολότερο αφού μπορεί να ελέγχεται κεντρικά από ένα άτομο ή να κατανέμεται σε διάφορους ανθρώπους.

Το σχεσιακό μοντέλο (relational model) δεδομένων υλοποιεί τα δεδομένα και τις μεταξύ τους σχέσεις ως ένα σύνολο πινάκων. Κάθε πίνακας (table) αποτελείται από στήλες (columns) με μοναδικά ονόματα. Η κάθε γραμμή (row) του πίνακα παριστάνει μια σχέση (relationship) ανάμεσα σε ένα σύνολο από τιμές.

Πίνακας **Τηλέφωνα** σε Βάση δεδομένων **Ατζέντα**

Όνομα	**Τηλέφωνο**	**Διεύθυνση**
Γιώργος	32560	Ιουλιανού 23
Μαρία	61359	Ασκληπιού 90
Θανάσης	98756	Αγ. Δημητρίου 40
Λίνα	78999	Σαρανταπόρου 17
Πέτρος	12356	Ιουλιανού 12

Γλώσσες ερωταποκρίσεων (SQL, XML)

Η **SQL** (Structured Query Language) αποτελεί την πιο διαδεδομένη γλώσσα διαχείρισης σχεσιακών βάσεων δεδομένων. Η SQL είναι μια γλώσσα ερωταποκρίσεων με την οποία μπορεί να δημιουργηθεί μια βάση δεδομένων, οι πίνακες και οι

μεταξύ τους σχέσεις, να εισαχθούν δεδομένα σε αυτούς και να γίνουν ερωτήματα πάνω σε αυτά. Επίσης, υπάρχει η δυνατότητα καθορισμού διαφορετικών δικαιωμάτων σε χρήστες. Για παράδειγμα σε μια εταιρία, ο εργαζόμενος στην αποθήκη μπορεί να βλέπει και να αλλάζει τα αποθέματα που υπάρχουν στην αποθήκη όσα χρειάζεται, ενώ ο υπεύθυνος για τη μισθοδοσία μπορεί να βλέπει και να επεξεργάζεται τους μισθούς των υπαλλήλων. Αλλά δεν έχει νόημα να μπορούν να βλέπουν όλοι όλα, ούτε φυσικά να έχουν δικαίωμα επεξεργασίας.

Παράδειγμα ερωτήματος στον πίνακα "Τηλέφωνα" που ζητάμε να εμφανιστούν τα ονόματα και τα τηλέφωνο αυτών που μένουν στην διεύθυνση Ιουλιανού σε οποιοδήποτε αριθμό.

SELECT Όνομα, Τηλέφωνο

FROM Τηλέφωνα

WHERE Διεύθυνση LIKE «Ιουλιανού %»

Με δεδομένα αυτά στον παραπάνω πίνακα θα εμφανιστούν

Όνομα	**Τηλέφωνο**
Γιώργος	32560
Πέτρος	12356

***HTML* είναι το ακρώνυμο από το *H*yper *T*ext *M*arkup *L*anguage που σημαίνει *γλώσσα χαρακτηρισμού* υπερκειμένου. Η χρήση μιας γλώσσας χαρακτηρισμού σημαίνει ότι γράφεται πρώτα το κείμενο και έπειτα προσθέτονται ειδικά σύμβολα γύρω από τις λέξεις ή από ολόκληρες προτάσεις ώστε να καθοριστεί η εμφάνιση τους στην οθόνη. Τα ειδικά σύμβολα στην HTML λέγονται *ετικέτες* (tags).**

Σήμερα (2014) υπάρχει πολύ πληροφορία, το ζητούμενο είναι η ευκολία στην πρόσβαση, την μεταφορά και την ευελιξία της. Η **XML** (eXtensible Markup Language) δεν είναι μία σημειακή γλώσσα όπως η HTML, είναι μία γλώσσα που χρησιμοποιείται για την περιγραφή μίας σημειακής γλώσσας.

Ο τεχνικός όρος μιας τέτοιας γλώσσας είναι μετα-γλώσσα. Σχεδιάστηκε για να μειώσει την πολυπλοκότητα της HTML. Η XML χρησιμοποιείται για την απλή και εύχρηστη αποθήκευση και διανομή δεδομένων. Οι εφαρμογές της μοιάζουν με βάσεις δεδομένων παρά με σελίδες περιεχομένου που θα επιστραφούν στο χρήστη μέσα από έναν φυλλομετρητή. Ως εφαρμογές/δυνατότητες της XML αναφέρονται:

- η *ανταλλαγή δεδομένων μεταξύ μη συμβατών συστημάτων.*
- *η δυνατότητα διαμοίρασης δεδομένων διαφόρων εφαρμογών.*
- *η δυνατότητα απεικόνισης των ίδιων δεδομένων με διαφορετικό τρόπο ανάλογα με τον σκοπό της χρήσης τους (π.χ. παρουσίαση λογιστικών δεδομένων σε πίνακα ή σε διάγραμμα).*
- *η δημιουργία νέων γλωσσών (π.χ. η γλώσσα WAP που χρησιμοποιείται για την αναπαράσταση πληροφορίας του Internet σε κινητά τηλέφωνα, είναι γραμμένη σε XML).*
- *Web Browsers Internet*
- *το ηλεκτρονικό εμπόριο*
- *η ανάκτηση πληροφοριών* από βάσεις δεδομένων

Ερωτήσεις – Δραστηριότητες

1. Δώστε παραδείγματα από ΠΣ που γνωρίζετε.
2. Να γίνει συζήτηση σε ομάδες για τη χρήση και την αξιοποίηση των ΠΣ.
3. Να γίνει συζήτηση σε ομάδες για τα πλεονεκτήματα αποθήκευσης στο σύννεφο.
4. Μελετήσετε 3 διαφορετικές υπηρεσίες αποθήκευσης στο σύννεφο και γράψετε τις απόψεις σας σε Padlet (www.padlet.com), που θα έχει δημιουργήσει ο εκπαιδευτικός.

3.3. Δίκτυα Υπολογιστών

Ορισμός δικτύου Η/Υ

Σε καθένα από τους τρεις προηγούμενους αιώνες, κυριάρχησε μία συγκεκριμένη τεχνολογία. Κατά το 18 αιώνα, την Βιομηχανική επανάσταση καθόρισε η εποχή των μεγάλων μηχανικών συστημάτων. Ο 19ος αιώνας ήταν η εποχή της ατμομηχανής. Στον 20 αιώνα η εγκατάσταση και εξάπλωση των τηλεφωνικών δικτύων σε παγκόσμια κλίμακα, η εφεύρεση του ραδιοφώνου και της τηλεόρασης, συνέβαλλαν καθοριστικά στην τεχνολογία της συλλογής, επεξεργασίας και διανομής της πληροφορίας. Στις πρώτες δεκαετίες του 21ου αιώνα παρατηρούμε πλέον έντονα την τάση οι διάφοροι αυτοί τεχνολογικοί τομείς διαχείρισης της πληροφορίας να συγκλίνουν. Αυτό έχει ως αποτέλεσμα να εξαφανίζονται οι διαφορές μεταξύ συλλογής, μεταφοράς, αποθήκευσης και επεξεργασίας της πληροφορίας. Σ' αυτό συμβάλουν ασφαλώς σε μέγιστο βαθμό τα **δίκτυα υπολογιστών.**

Δίκτυα υπολογιστών (computer networks): *Ένα πλήθος από διασυνδεδεμένους υπολογιστές (συχνά αναφέρονται και ως "συσκευές" ή "μηχανές"), οι οποίοι εξυπηρετούν τις ανάγκες ενός οργανισμού. Αυτό επιτυγχάνεται με ένα κατάλληλο συνδυασμό από ποικίλες τεχνολογίες υλικού (hardware) και λογισμικού (software).*

Τα βασικά δομικά στοιχεία ενός δικτύου υπολογιστών είναι

- *Τα υπολογιστικά συστήματα*, δηλαδή οι κάθε είδους αυτόνομοι υπολογιστές, οι οποίοι είναι διασυνδεδεμένοι στο συγκεκριμένο δίκτυο. Στην εποχή μας ένας τέτοιος υπολογιστής μπορεί να είναι ένας κεντρικός διακομιστής ενός μεγάλου οργανισμού, ο υπολογιστής γραφείου, ο φορητός υπολογιστής, το κινητό τηλέφωνο, αλλά και οποιαδήποτε άλλη σύγχρονη συσκευή, που ελέγχεται από μικροεπεξεργαστή (πχ "Έξυπνη" τηλεόραση).
- *Οι γραμμές μετάδοσης*, οι τρόποι και τα μέσα με τα οποία επιτυγχάνεται η διασύνδεση των υπολογιστικών συστημάτων. Οι γραμμές μετάδοσης, προκειμένου να επιτύχουν την επικοινωνία μεταξύ των συστημάτων, συνοδεύονται και από τις κατάλληλες τεχνολογίες (πρωτόκολλα επικοινωνίας), οι οποίες είναι κοινές για όλους τους υπολογιστές του δικτύου. Στην εποχή μας οι γραμμές μετάδοσης μπορεί να είναι ενσύρματες (χάλκινα καλώδια, οπτικές ίνες) ή ασύρματες (πχ δίκτυα κινητής τηλεφωνίας).

- *Οι μονάδες μεταγωγής*, δηλαδή οι συσκευές οι οποίες είναι υπεύθυνες για την σωστή μετάδοση, δρομολόγηση των δεδομένων που ανταλλάσσονται μεταξύ υπολογιστών διαφορετικών τμημάτων (υποδικτύων) που αποτελούν το δίκτυο.

Στην εποχή μας πλέον, ο όρος "**οργανισμός"** στον παραπάνω ορισμό του δικτύου υπολογιστών, μπορεί να είναι:

- το σπίτι μας, όπου στο οικιακό δίκτυο διασυνδέονται ο υπολογιστής του γραφείου, το κινητό τηλέφωνό, η τηλεόραση κλπ)
- μία εταιρία, όπου στο εταιρικό της δίκτυο διασυνδέονται υπολογιστές, φορητοί, τηλέφωνα κλπ
- ολόκληρος ο πλανήτης, όπου πρακτικά, δίκτυα υπολογιστών, μικρά και μεγάλα από κάθε σημείο της γης, διασυνδέονται και ανταλλάσσουν κάθε μορφής πληροφορίες και υπηρεσίες.

Κατηγορίες Δικτύων

Τα δίκτυα υπολογιστών μπορούν να κατηγοριοποιηθούν με βάση συγκεκριμένα κριτήρια. Τα κριτήρια αυτά καθορίζουν και τις ανάλογες τεχνολογίες υλικού και λογισμικού που απαιτούνται για την κάθε υλοποίηση. Στην εποχή μας, βέβαια, τα σύγχρονα δίκτυα ενσωματώνουν μέσα στο ίδιο δίκτυο τεχνολογίες από διαφορετικές κατηγορίες.

1. *Με κριτήριο την έκταση της περιοχής που καλύπτουν (Σχήμα 3.4)*

- *Τοπικά δίκτυα (LAN, local area networks)*

Είναι δίκτυα υπολογιστών, που καλύπτουν ένα κτίριο ή ένα κτιριακό συγκρότημα (πχ σχολείο) ή ακόμα και μερικά χιλιόμετρα (πχ πανεπιστημιούπολη). Συνήθως σκοπός ενός τέτοιου δικτύου είναι η ανταλλαγή δεδομένων μεταξύ των υπολογιστών, πληροφοριών μεταξύ των χρηστών του δικτύου, αλλά και η κοινή χρήση πόρων του δικτύου (πχ κεντρικοί αποθηκευτικοί χώροι, εκτυπωτές κ.α.).

- *Μητροπολιτικά δίκτυα (MAN, metropolitan area networks)*

Τα δίκτυα αυτά καλύπτουν την έκταση μιας πόλης. Μπορεί να είναι γενικού σκοπού (ανταλλαγή δεδομένων, διαμοιρασμό πόρων) αλλά και ειδικού (πχ δίκτυα καλωδιακής τηλεόρασης)

- *Δίκτυα ευρείας περιοχής (WAN, wide area networks)*

Πρόκειται για δίκτυα υπολογιστών, που εκτείνονται σε μια μεγάλη γεωγραφική περιοχή, όπως μία χώρα ή μια ήπειρο. Αποτελούνται ουσιαστικά από ποικίλα άλλα μικρότερα δίκτυα υπολογιστών (υποδίκτυα), τα οποία με την σειρά τους συνδέονται, σχηματίζοντας ένα δίκτυο ευρείας περιοχής. Βασικά δομικά στοιχεία των συγκεκριμένων δικτύων είναι οι ποικίλες γραμμές μετάδοσης, αλλά και τα στοιχεία μεταγωγής που τις διασυνδέουν, προκειμένου να επιτευχθεί η σωστή δρομολόγηση των δεδομένων μέσα στο δίκτυο. Η διασύνδεση των δικτύων ευρείας περιοχής δημιουργεί ένα δίκτυο παγκόσμιας κλίμακας, γνωστό με τον όρο **διαδίκτυο.**

Απόσταση ανάμεσα στους επεξεργαστές	Επεξεργαστές που βρίσκονται στο ίδιο	Χαρακτηρισμός δικτύου
1 μέτρο	Τετραγωνικό μέτρο	Δίκτυο προσωπικής περιοχής
10 μέτρα	Δωμάτιο	Δίκτυο τοπικής περιοχής
100 μέτρα	Κτίριο	
1 χιλιόμετρο	Πανεπιστημιούπολη	
10 χιλιόμετρα	Πόλη	Μητροπολιτικό δίκτυο
100 χιλιόμετρα	Χώρα	Δίκτυο ευρείας περιοχής
1000 χιλιόμετρα	Ήπειρο	
10000 χιλιόμετρα	Πλανήτη	Το ίντερνετ

Σχημα 3.4 Κατηγοριοποίηση δικτύων

2. Με κριτήριο το μέσο μετάδοσης (Σχήμα 3.5)

Σχήμα 3.5: Ενσύρματα και ασύρματα δίκτυα υπολογιστών

- *Ενσύρματα δίκτυα (wired networks)*

Είναι τα δίκτυα υπολογιστών στα οποία η διασύνδεση επιτυγχάνεται με ενσύρματα μέσα, όπως ομοαξονικά (coaxial) καλώδια, συνεστραμμένα (twisted) καλώδια ή οπτικές ίνες (optical fibers). Τέτοια περίπτωση αποτελεί και το εργαστήριο

υπολογιστών ενός σχολείου, όπου ο κάθε σταθμός εργασίας διασυνδέεται μέσω καλωδίων συνεστραμμένων ζευγών.

- *Ασύρματα δίκτυα (wireless networks)*

Πρόκειται για δίκτυα υπολογιστών οποιαδήποτε κλίμακας, στα οποία η διασύνδεση επιτυγχάνεται με ασύρματες ζεύξεις (ραδιοκύματα). Σε αυτή την κατηγορία ανήκουν οι περιπτώσεις διασύνδεσης υπολογιστών και συσκευών του δικτύου μέσω bluetooth (ασύρματη ζεύξη μικρής εμβέλειας), τα ασύρματα δίκτυα τοπικής περιοχής (WLAN, ή γνωστά και ώς Wifi) αλλά και τα δίκτυα κινητής τηλεφωνίας (μεγάλης κλίμακας ασύρματη δικτύωση).

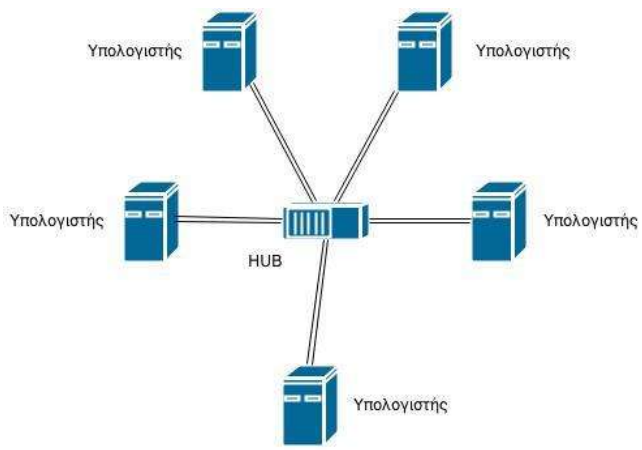
3. Με κριτήριο την τεχνολογία μετάδοσης του σήματος

- *Δίκτυα εκπομπής (Broadcasting networks)*,
στα οποία υπάρχει ένα μόνο κοινός δίαυλος επικοινωνίας για όλους τους σταθμούς του δικτύου. Τα μηνύματα (σήματα) που στέλνονται από ένα υπολογιστή, λαμβάνονται απ' όλους του υπολογιστές του δικτύου αλλά γίνονται αποδεκτά μόνο από τον υπολογιστή, στον οποίο απευθύνονται. Αυτό γίνεται μέσω ενός πεδίου διεύθυνσης, όπου καταγράφεται ο προορισμός του μηνύματος.
- *Δίκτυα μεταγωγής (Switched networks).*
Στη περίπτωση αυτή οι συνδέσεις στο δίκτυο είναι σημείο προς σημείο. Αυτό σημαίνει ότι πολλές φορές συμβαίνει τα μηνύματα προκειμένου να φτάσουν στο προορισμό τους, επισκέπτονται πολλές ενδιάμεσες μηχανές. Στα δίκτυα αυτά παρεμβάλλονται μεταξύ των συνδέσεων, ενδιάμεσοι κόμβοι/συσκευές οι οποίες δρομολογούν προς την κατάλληλη σύνδεση τα μηνύματα Οι συσκευές αυτές ονομάζονται μεταγωγείς (switches)

Τοπολογίες δικτύων

Με τον όρο τοπολογία ενός δικτύου υπολογιστών, εννοούμε την διάταξη, τον τρόπο με τον οποίο οι μηχανές και οι γραμμές σύνδεσης απαρτίζουν το συγκεκριμένο δίκτυο.

Οι βασικές κατηγορίες είναι οι παρακάτω:



Σχήμα 3.6: Τοπολογία αστέρα

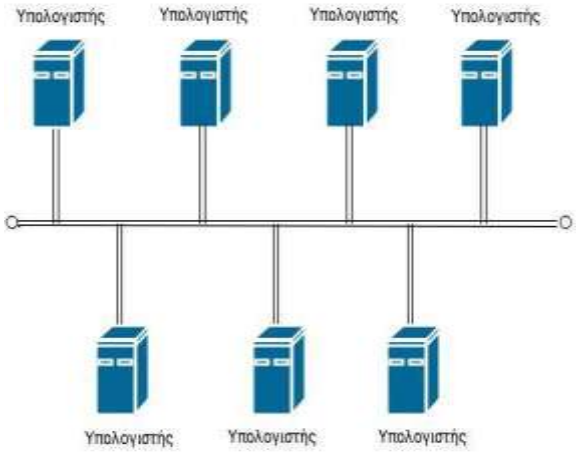
➢ *Τοπολογία αστέρα (Star topology) (Σχήμα 3.6)*
Κάθε συσκευή του δικτύου συνδέεται απευθείας με ένα κεντρικό κόμβο (hub). Συνεπώς τα δεδομένα προκειμένου να φτάσουν στη συσκευή-προορισμό τους, διέρχονται πάντα από το κόμβο.

Πλεονεκτήματα**:**

✓ Ασφαλή, σταθερή μετάδοση.
✓ Αν πάθει βλάβη μία συσκευή, οι υπόλοιπες επικοινωνούν κανονικά.

Μειονεκτήματα:

- ✓ Προφανώς αν πάθει βλάβη ο κόμβος, τότε το δίκτυο βγαίνει εκτός λειτουργίας.



Σχήμα 3.7: Τοπολογία διαύλου

➢ *Τοπολογία διαύλου (bus topology) (Σχήμα 3.7)*

Υπάρχει μία κοινή γραμμή-ραχοκοκαλία (backbone) του δικτύου, και όλες οι συσκευές συνδέονται πάνω σε αυτή.

Πλεονεκτήματα**:**

- ✓ Μικρός αριθμός καλωδιώσεων
- ✓ Εύκολη προσθαφαίρεση συσκευών στο δίκτυο

Μειονεκτήματα:

- ✓ Δυνατότητα σύνδεσης περιορισμένου αριθμού συσκευών
- ✓ Δυσκολία στην διόρθωση σφαλμάτων και στην επαναφορά του δικτύου σε περίπτωση δυσλειτουργίας

➢ *Τοπολογία δακτυλίου (Ring topology) (Σχήμα 3.8)*

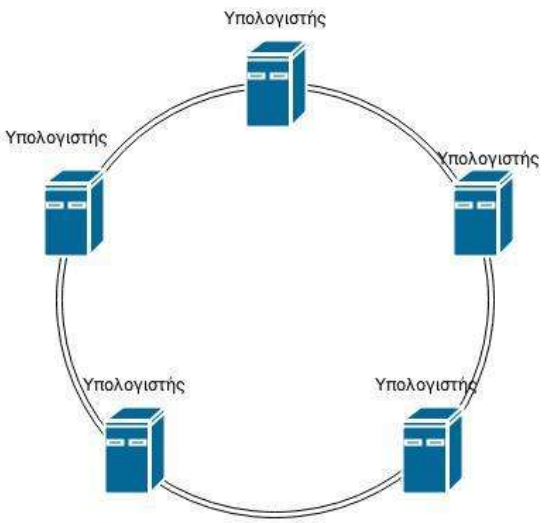
Κάθε συσκευή του δικτύου συνδέεται με τις δύο διπλανές στο δακτύλιο συσκευές. Με αυτό τον τρόπο σχηματίζεται ένας δακτύλιος.

Πλεονεκτήματα:

- ✓ Ευκολότερη ανίχνευση και διόρθωση σφαλμάτων
- ✓ Ευκολότερη εγκατάσταση λόγω των περιορισμένων καλωδιώσεων.

Μειονεκτήματα:

- ✓ Περιορισμένος αριθμός συσκευών που μπορούν να συνδεθούν
- ✓ Πιθανή βλάβη σε ένα σταθμό, προκαλεί κατάρρευση όλου του δικτύου.



Σχήμα 3.8: Τοπολογία δακτυλίου

Σύγχρονες υπηρεσίες δικτύων:

Τα δίκτυα υπολογιστών προσφέρουν διάφορες πολύτιμες για τους χρήστες τους, όπως διαμοιρασμό των πόρων του δικτύου (αποθηκευτικά μέσα, εκτυπωτές κλπ), ανταλλαγή μηνυμάτων, κοινή χρήση εφαρμογών και άλλα.

Στην εποχή μας τα δίκτυα υπολογιστών διασυνδέονται μεταξύ τους δημιουργώντας ένα παγκόσμιο *δίκτυο των δικτύων*, γνωστού και ως **διαδίκτυο** (internet). Αυτό έχει ώς αποτέλεσμα να προσφέρονται επιπλέον υπηρεσίες, όπως:

- *Παγκόσμιος Πληροφοριακός ιστός (WWW, World Wide Web)*

Πλοήγηση και αναζήτηση πληροφορίας σε σελίδες υπερκειμένου (οι οποίες συνδυάζουν κείμενο, εικόνα, ήχο και βίντεο), αλλά και δημιουργία και διάθεση ιστοσελίδων και περιεχομένου από τους ίδιους τους χρήστες. Επίσης προσφέρει την δυνατότητα ηλεκτρονικών συναλλαγών και εμπορίου (e-commerce) χωρίς γεωγραφικό περιορισμό.

- *Επικοινωνία*

Ανταλλαγή ηλεκτρονικών μηνυμάτων (ηλεκτρονική αλληλογραφία, email) μεταξύ των χρηστών του διαδικτύου. Επίσης, προσφέρει ανταλλαγή γραπτών μηνυμάτων, ήχου και βίντεο σε πραγματικό χρόνο με την χρήση διαφόρων υπηρεσιών. Ηλεκτρονικές κοινότητες χρηστών μπορούν να ανταλλάξουν απόψεις μέσω ποικίλων υπηρεσιών (usenet, irc, forums κλπ)

- *Μεταφορά δεδομένων*

Ανταλλαγή αρχείων (κειμένου, ήχου, βίντεο κλπ) μεταξύ χρηστών από όλο τον κόσμο με διάφορους τρόπους (ftp, p2p sharing και άλλους).

Ερωτήσεις – Δραστηριότητες

1. Τι είναι ένα δίκτυο υπολογιστών; Ποια τα πλεονεκτήματα που προσφέρει;
2. Ποιες οι βασικές τοπολογίες δικτύων υπολογιστών;
3. Περιηγηθήτε στη διεύθυνση http://photodentro.edu.gr/v/item/ds/5145 όπου απεικονίζεται η δομή ενός τυπικού σχολικού εργαστηρίου. Συζητήστε για την χρησιμότητα των βασικών δομικών στοιχείων που απεικονίζονται.
4. Παρατηρήστε το εργαστήριο του σχολείου σας και αναγνωρίστε:
   1. τα δομικά στοιχεία του δικτύου.
   2. εάν υπάρχει ασύρματο δίκτυο ή όχι
   3. ποια η τοπολογία του δικτύου
   4. τι είδους δίκτυο είναι με βάση τα κριτήρια που αναφέρονται στην θεωρία.
   5. Σχεδιάστε μια διαγραμματική αναπαράσταση που θα απεικονίζει τη συνδεσμολογία του εργαστηρίου του σχολείου σας.

3.4. Τεχνητή Νοημοσύνη

Τι είναι η Τεχνητή Νοημοσύνη (Artificial Intelligence)

Η Τεχνητή Νοημοσύνη έχει ερευνηθεί για δεκαετίες και έχει οδηγήσει σε πολλά χρήσιμα προϊόντα, αλλά εξακολουθεί να μην υπάρχει σύστημα που είναι τόσο έξυπνο όσο ένας άνθρωπος. Συγκεκριμένα, οι υπολογιστές δεν έχουν σκέψη ούτε συνείδηση. Απλά εκτελούν τις εντολές που παίρνουν από τον άνθρωπο.

Τεχνητή Νοημοσύνη *είναι ο τομέας της επιστήμης των υπολογιστών, που ασχολείται με τη σχεδίαση ευφυών υπολογιστικών συστημάτων, δηλαδή συστημάτων που επιδεικνύουν χαρακτηριστικά που σχετίζουμε με τη νοημοσύνη στην ανθρώπινη συμπεριφορά.*

Τα συστήματα της Τεχνητής Νοημοσύνης επιδεικνύουν στοιχειώδη ευφυή συμπεριφορά, όπως μάθηση, εξαγωγή συμπερασμάτων, προσαρμοστικότητα, επίλυση προβλημάτων, κατανόηση από τα συμφραζόμενα.

Εξέλιξη της Τεχνητής Νοημοσύνης

Κατά τη δεκαετία του 1940 εμφανίστηκε η πρώτη μαθηματική περιγραφή τεχνητού νευρωνικού δικτύου (ΤΝΔ) με πολύ περιορισμένες δυνατότητες επίλυσης αριθμητικών προβλημάτων. Στα ΤΝΔ η γνώση αποκτάται από το δίκτυο μέσω μιας διαδικασίας εκμάθησης. Το 1950 ο μαθηματικός Alan Turing, πατέρας της θεωρίας υπολογισμού, πρότεινε το τέστ Τούρινγκ· μία απλή διαδικασία που θα μπορούσε να εξακριβώσει αν μία μηχανή διαθέτει ευφυΐα.

Στα μέσα του '70 υπήρξε μία αναθέρμανση του ενδιαφέροντος για τον τομέα της Τ.Ν. λόγω των εμπορικών εφαρμογών που απέκτησαν τα *έμπειρα συστήματα* (expert systems), μηχανές Τ.Ν. με αποθηκευμένη γνώση για έναν εξειδικευμένο τομέα και δυνατότητα ταχείας εξαγωγής λογικών συμπερασμάτων, τα οποία συμπεριφέρονται, όπως ένας άνθρωπος ειδικός στον αντίστοιχο τομέα.

Κατά τη δεκαετία του '90, με την ραγδαία εξάπλωση και χρήση του διαδικτύου, ανάπτυξη γνώρισαν οι *ευφυείς πράκτορες* (intelligent agents). Πρόκειται για αυτόνομο λογισμικό Τ.Ν. τοποθετημένο σε κάποιο περιβάλλον που στοχεύει στην παροχή βοήθειας στους χρήστες του, στη συλλογή ή ανάλυση πολλών δεδομένων ή στην αυτοματοποίηση επαναλαμβανόμενων εργασιών.

Τέλος, τη δεκαετία του 2000 η Τεχνητή Νοημοσύνη συνεχίζει να παίζει ένα πολύ σπουδαίο ρόλο στο χώρο της νευροβιολογίας, των νευροεπιστημών, της ρομποτικής, των παιχνιδιών και της διάγνωσης ασθενειών.

Τομείς Εφαρμογής της Τεχνητής Νοημοσύνης

Η ΤΝ έχει εφαρμογή σε πολλούς τομείς των επιστημών όπως: η επιστήμη των υπολογιστών, η νευροβιολογία, η φιλοσοφία, η ψυχολογία, η γλωσσολογία και η γνωσιακή επιστήμη, με στόχο τη σύνθεση ευφυούς συμπεριφοράς, με στοιχεία

συλλογιστικής, μάθησης και προσαρμογής στο περιβάλλον, ενώ συνήθως εφαρμόζεται σε μηχανές ή υπολογιστές ειδικής κατασκευής.

Επίσης, ένας σημαντικός τομέας εφαρμογής της Τ.Ν. είναι η ψηφιακή αναγνώριση φωνής όπως και τα διαδικτυακά ρομπότ (bots). Από τα τελευταία τα πιο γνωστά είναι πράκτορες που συμμετέχουν σε συζητήσεις (chatbots). Ουσιαστικά πρόκειται για ένα πρόγραμμα υπολογιστή που προσομοιώνει την ανθρώπινη συνομιλία. Η συνομιλία αυτή μπορεί να είναι είτε γραπτή είτε λεκτική.

Ένα απλό παράδειγμα με την γλώσσα Prolog:

1. Μερικές απλές δηλώσεις..
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).

2. Τα παρακάτω ερωτήματα οδηγούν στις συγκεκριμένες απαντήσεις που βλέπετε:
| ?- likes(mary,food).
yes.
| ?- likes(john,wine).
yes.
| ?- likes(john,food).
no.

Σχήμα 3.9: Κώδικας σε Prolog

Γλώσσες προγραμματισμού που χρησιμοποιούνται στην Τεχνητή Νοημοσύνη

Οι δύο βασικές γλώσσες προγραμματισμού που χρησιμοποιούνται στο χώρο της Τεχνητής Νοημοσύνης είναι η Lisp και η Prolog (Σχήμα 3.9).

Οι δύο αυτές γλώσσες προγραμματισμού είναι γλώσσες λογικού προγραμματισμού που χρησιμοποιούνται σε πολλούς τομείς της Τεχνητής Νοημοσύνης, όπως τα έμπειρα συστήματα, γενετικοί αλγόριθμοι, τεχνητά νευρωνικά δίκτυα, ασαφής λογική, καθώς και το προβλεπόμενο αρχικό τομέα της χρήσης, δηλαδή την επεξεργασία φυσικής γλώσσας.

Ερωτήσεις – Δραστηριότητες

1. Δώστε τον ορισμό της Τ.Ν. καθώς και 3 σημαντικές εφαρμογές της Τ.Ν..
2. Αναφέρατε συνοπτικά τα στάδια εξέλιξης της Τ.Ν. καθώς και δύο γλώσσες με τις οποίες μπορεί να υλοποιηθεί.
3. Δημιουργήσετε μία χρονοσειρά, χρησιμοποιώντας ένα εργαλείο web 2.0 όπως το εργαλείο timetoast (www.timetoast.com) ή το εργαλείο timerine (www.timerime.com), όπου θα φαίνονται οι εξελίξεις στην πρόοδο της Τεχνητής Νοημοσύνης. Πάρτε πληροφορίες από την διεύθυνση: http://users.sch.gr/jenyk/index.php/artificialintelligence/ai-historicalreview/5-historicalroute
4. Παρακολουθήστε την ομιλία με τίτλο "Η συναισθηματική νοημοσύνη των υπολογιστών" και συζητήστε σε ομάδες των 3 μαθητών τι σας κάνει μεγαλύτερη εντύπωση.

   www.youtube.com/watch?v=8iDub4Mg64U .

Παράρτημα

Ελληνική βιβιογραφία

- Βαζιργιάννης, Μ., & Χαλκίδη, Μ. (2005). *Εξόρυξη από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό*. Αθήνα: Τυπωθήτω – Γιώργος Δαρδάνος.
- Βλαχάβας, Ι., Βασιλειάδης, Ν., Κόκκορας, Φ., & Σακελλαρίου, Η. (2006). *Τεχνητή Νοημοσύνη*. Αθήνα: Εκδόσεις Γκιούρδας.
- Ζησιμόπουλος, Β. (2008). *Αλγόριθμοι και Πολυπλοκότητα* Σημειώσεις από το Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Ανασύρθηκε στις 14 Ιουνίου 2014, από http://cgi.di.uoa.gr/~vassilis/ac/VZalgorithms08.pdf
- Κοτίνη Ι. & Τζελέπη Σ. (2012). *Η Συμβολή της Υπολογιστικής Σκέψης στην Προετοιμασία του Αυριανού Πολίτη*. 4th CIE2012, Conference on Informatics in Education 2012 – Η Πληροφορική στην Εκπαίδευση. Πανεπιστήμιο Πειραιώς.
- Lister, A.M., & Eager, R.D. (Επιμ. Π. Γεωργιαδης) (1991). *Λειτουργικά Συστήματα* (Χ. Χαλδαιόπουλος & Α. Ζήση Μτφρ.). Αθήνα: Δίαυλος.
- Ούτσιος, Ε.Γ. (2004). *ΑΛΓΟΡΙΘΜΟΙ & ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ*. Σημειώσεις από ΤΕΙ Σερρών. Ανασύρθηκε στις 5 Ιουνίου 2014, από ftp://teiser.gr/pliroforiki/Algorithmoi-DomesDedomenon/labNotesADS.pdf
- Russell, S., & Norvig, P. (2004). *Τεχνική Νοημοσύνη Μία Σύγχρονη Προσέγγιση* (2η Αμερικάνικη έκδ.). Αθήνα: Κλειδάριθμος.
- Σάββας, Η.Κ. (2005). *Αλγόριθμοι & Πολυπλοκότητα* (Σημειώσεις για το μάθημα). Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών, ΤΕΙ Λάρισας, Ιανουάριος 2005. Ανασύρθηκε 14 Ιουνίου 2014, από http://www.teilar.gr/dbData/ProfAnn/profann-2a82b2bb.pdf
- Σπυρακης, Π. (2001). *Λειτουργικά συστήματα Ι*. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο.
- Tanenbaum, A. S. [1996] (2000). *Δίκτυα Υπολογιστών* (3η έκδ.) Πρώτη Ελληνική Έκδοση. Αθήνα: Εκδόσεις Παπασωτηρίου.
- Χατζηλυγερούδης, Ι. (2000). *Δομές Δεδομένων*. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο.

Ξένη βιβλιογραφία

- Abelson, H. (1986). *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press.
- Brunskill, D., & Turner, J. (1997). *Understanding Algorithms and Data Structures*. USA: McGraw-Hill.
- Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). Cambridge: MIT press.
- Curzon, P. (2002). *Computing Without Computers*. Unpublished booklet.
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). *Data Structures and Algorithms in Python*. Wiley Publishing.
- Jones D.N. (1997). Computability and Complexity. From a Programming Perspective. Massachusetts: MIT Press.
- Klein, R., & Kamphans, T. (2011). *Pledge's Algorithm-How to Escape from a Dark Maze*. *In Algorithms Unplugged* (pp. 69-75). Springer Berlin Heidelberg.
- Knuth, E.D. (1968). *The Art of Computer Programming*. Reading: Addison - Wesley Publish Company.
- Knuth, D. E. (1997). *The Art of Computer Programming: Fundamental Algorithms, Vol.1* (3rd ed.). USA: Addison Wesley.
- Lister, A., (Andrew), 1945-Eager, R. D.
- Mehta, D. P. (Ed.). (2004). *Handbook of data structures and applications*. CRC Press.
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web*. Ανακτήθηκε στις 18 Ιουνίου 2014, από http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
- Ryba, A. J., & Kruse, R. L. K. (1999). *Data Structures and Program Design in C++*. Prentice Hall.
- Sedgewick, W. (2011). *Algorithms* (4th ed.). Pearson Education.
- Senn, J.A. (1989). *Analysis and Design of Information Systems* (2nd ed.). London: McGraw-Hill International Editions.
- Shaffer, C. A. (2011). *Data Structures & Algorithm Analysis in Java*. Courier Dover Publications.
- Skiena, S. (2008). *The Algorithm Design Manual* (2nd ed.). USA : Springer.
- Sommerville I.(2007), Software Engineering. London: Addison-Wesley Publishing Company, 8th edition.
- Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., & Wagner, D. (2011). *Algorithms Unplugged*. Springer-Verlag Berlin Heidelberg.
- Zurada, J. M. (1992). *Introduction to Artificial Neural Systems*. St. Paul: West Publishing Company.

Διευθύνσεις διαδικτύου

- http://www.acm.org/about/class
  ACM Computing Classification System
- http://el.wikipedia.org/wiki/Πληροφορική
  Λήμμα της Wikipedia για τον γνωστικό κλάδο της επιστήμης υπολογιστών
- http://www.cs.ucy.ac.cy/courses/EPL003/e-book.pdf
  Εισαγωγή στην Επιστήμη των Υπολογιστών από το ΕΑΠ
- http://www.archimedes-lab.org/game_nim/nim.html
  Πληροφορίες για το Nim Game και δυνατότητα online gaming
- http://www.it.uom.gr/project/parallel/kef1/1.1.htm
  Σημειώσεις για τις Τεχνικές Παράλληλου Προγραμματισμού
- http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html
  Παραδείγματα συναρτησιακού προγραμματισμού από τον Kurt Nørmark, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Aalborg, Δανία
- http://cdn.oreillystatic.com/news/graphics/prog_lang_poster.pdf
  Ιστορία των γλωσσών προγραμματισμού
- http://www.ustudy.in/node/1975
  Δικτυακός τόπος με παραδείγματα γλωσσών προγραμματισμού
- http://alkisg.mysch.gr/
  Η σελίδα της καθηγητή Πληροφορικής, κ. Άλκη Γεωργόπουλο για τον Διερμηνευτή της Γλώσσας
- http://www.ecedu.upatras.gr/flowchart/
  Δικτυακός τόπος για τον Δημιουργό Διαγραμμάτων Ροής (Visual Flowchart)
- http://www.pseudoglossa.gr/
  On-line μεταγλωττιστής για αλγορίθμους σε ψευδογλώσσα, του κ. Στάθη Στέργου
- http://www.fme.aegean.gr/sites/default/files/dsampson_xml_lectures-notes-dec2003.pdf
  Πανεπιστημιακές σημειώσεις για το αντικείμενο "Η Γλώσσα Σήμανσης XML" από τον Επίκουρο Καθηγητή, Δημήτριο Σαμψών, Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιά
- http://computer.howstuffworks.com/cloud-computing/cloud-computing.htm
  Δικτυακός τόπος με την λειτουργία του Διαδικτυακού σύ
- http://aetos.it.teithe.gr/~iliou/cs4804/dialexeis/tmp/8.cloud_computing.pdf
  Σημειώσεις για την Υπολογιστική Νέφους από τον Χρ. Ηλιούδη
- http://en.wikipedia.org/wiki/Storage_area_network
  Λήμμα της Wikipedia για το δίκτυο περιοχής αποθήκευσης
- http://photodentro.edu.gr/
  Μαθησιακά Αντικείμενα από το Ψηφιακό Φωτόδεντρο

- http://en.wikipedia.org/wiki/Computer_network
  Λήμμα από την Wikipedia για το δίκτυο υπολογιστών
- http://www.eetn.gr/
  Η σελίδα της Ελληνικής Εταιρείας Τεχνητής Νοημοσύνης
- http://aitopics.org/
  Ενδιαφέροντα θέματα για την Τεχνητή Νοημοσύνη όπως η έρευνα, οι άνθρωποι και οι εφαρμογές της Τεχνητής Νοημοσύνης
- http://www-formal.stanford.edu/jmc/whatisai/whatisai.html/
  Ένας δικτυακός τόπος με τίτλο «Τι είναι η Τεχνητή Νοημοσύνη;» από τον John McCarthy στο Πανεπιστήμιο του Standford
- http://chaturing.com/artwork/chatbot/
  Είναι ένα chatbot, που λειτουργεί με γραπτό κείμενο στην Αγγλική γλώσσα
- http://sheepridge.pandorabots.com/pandora/talk?botid=fef38cb4de345ab1&skin=iframe-voice
  Ένα chatbot – συνομιλία με γραπτό κείμενο στην Αγγλική γλώσσα, με τον James Kirk από το Star Trek
- http://www.chatbots.org/
  Σελίδα με λίστα από διαθέσιμα chatbots και τους σκοπούς που εξυπηρετούν
- https://www.apple.com/ios/siri/
  Σελίδα της εταιρείας Apple, με το πολύ γνωστό λογισμικό αναγνώρισης φωνής Siri
- http://csunplugged.org/sites/default/files/activity_pdfs_other/intelligent%20piece%20of%20paper.el_.v6.pdf
  Η πολύ γνωστή δραστηριότητα για το έξυπνο χαρτί, μεταφρασμένη στα Ελληνικά
- http://www.chatbots.org/
  Σελίδα με λίστα από διαθέσιμα chatbots και τους σκοπούς που εξυπηρετούν.
- https://www.apple.com/ios/siri/
  Σελίδα της εταιρείας Apple, με το πολύ γνωστό λογισμικό αναγνώρισης φωνής Siri.
- http://csunplugged.org/sites/default/files/activity_pdfs_other/intelligent%20piece%20of%20paper.el_.v6.pdf
  Η πολύ γνωστή δραστηριότητα για το έξυπνο χαρτί, μεταφρασμένη στα Ελληνικά.

Λεξικό Όρων

SQL (Structured Query Language) η πιο διαδεδομένη γλώσσα διαχείρισης σχεσιακών βάσεων δεδομένων.

XML (eXtensible Markup Language) γλώσσα που χρησιμοποιείται για την περιγραφή μίας σημειακής γλώσσας. eXtensible= επεκτάσιμη από τον κάθε συγγραφέα, Markup= σήμανση όπως η HTML, Language: περιγραφική γλώσσα, που είναι διαφορετική από μια γλώσσα προγραμματισμού.

Αλγόριθμος, η σαφής και ακριβής περιγραφή μια συγκεκριμένης σειράς από εντολές με σκοπό την επίλυση ενός προβλήματος σε πεπερασμένο χρόνο.

Άλυτα προβλήματα, είναι τα προβλήματα για τα οποία έχει αποδειχθεί ότι δεν λύνονται.

Αναδρομικός λέγεται ο **αλγόριθμος** ο οποίος επιλύει ένα πρόβλημα επιλύνοντας ένα ή περισσότερα μικρότερα κομμάτια του ίδιου προβλήματος. Ένας αναδρομικός αλγόριθμος καλεί (χρησιμοποιεί) τον ίδιο τον εαυτό του.

Ανάλυση Αλγορίθμων: Ο τομέας της Πληροφορικής που ασχολείται με την αξιολόγηση της αποδοτικότητας των αλγορίθμων.

Ανοικτά προβλήματα, είναι τα προβλήματα για τα οποία δεν έχει δοθεί λύση αλλά παράλληλα δεν έχει αποδειχθεί ή μη ύπαρξη λύσης.

Βιβλιοθήκη είναι μια συλλογή από έτοιμα υποπρογράμματα που χρησιμοποιούνται στον δομημένο προγραμματισμό.

Γλώσσα προγραμματισμού, είναι μία τεχνητή γλώσσα που μπορεί να χρησιμοποιηθεί για τον έλεγχο και προγραμματισμό μιας μηχανής.

Διάγραμμα ροής, είναι ένας τρόπος αναπαράστασης αλγορίθμων με την χρήση γεωμετρικών σχημάτων.

Δίκτυα ευρείας περιοχής (WAN, wide area networks) Δίκτυα υπολογιστών που εκτείνονται σε μια μεγάλη γεωγραφική περιοχή, όπως μία χώρα ή μια ήπειρο.

Δρομολογητής (Router) Συσκευές που ενώνουν δυο ή περισσότερα διαφορετικά δίκτυα μεταξύ τους και είναι υπεύθυνες για την σωστή δρομολόγηση των πακέτων δεδομένων που αποστέλλονται από ή προς τα δίκτυα αυτά.

Εκσφαλμάτωση (Debugging) είναι η εύρεση και η διόρθωση των λογικών λαθών ενός αλγορίθμου.

Επαναληπτικός αλγόριθμος (repetition algorithm), είναι ο αλγόριθμος που εκτελεί επαναληπτικά τις ίδιες εντολές (διαδικασίες).

Επιλύσιμα προβλήματα, είναι τα προβλήματα των οποίων η λύση είναι ήδη γνωστή και έχει διατυπωθεί.

Κύκλος ζωής εφαρμογής λογισμικού: είναι η δομημένη διαδικασία για τη δημιουργία των λογισμικών. Αποτελείται από πέντε φάσεις: Ανάλυση, Σχεδίαση, Υλοποίηση, Έλεγχος, Συντήρηση.

Κωδικοποίηση είναι ο αυστηρός τρόπος περιγραφής ενός αλγορίθμου και αποτελείτε από συγκεκριμένες εντολές και υπακούει τους συντακτικούς κανόνες της γλώσσας προγραμματισμού.

Λειτουργικό σύστημα: το σύνολο των προγραμμάτων που είναι υπεύθυνο για την διαχείριση και συντονισμό των εργασιών σε ένα υπολογιστή, καθώς και την κατανομή των διαθέσιμων πόρων του

Λογισμικό Ανοικτού Κώδικα (Λ.Α.Κ.): είναι το λογισμικό που ο πηγαίος του κώδικας είναι διαθέσιμος σε οποιονδήποτε θέλει να τον μελετήσει, να τον αντιγράψει και να τον αλλάξει.

Μεταγωγέας (switch) Συσκευή πάνω στην οποία συνδέονται 2 ή περισσότεροι υπολογιστές ενός δικτύου, η οποία είναι υπεύθυνη για την σωστή δρομολόγηση των δεδομένων που ανταλλάσσονται μεταξύ αυτών.

Μητροπολιτικά δίκτυα (MAN, metropolitan area networks) Δίκτυα υπολογιστών τα οποία καλύπτουν την έκταση μιας πόλης.

Πληροφοριακό Σύστημα (ΠΣ) είναι ένα οργανωμένο σύνολο το οποίο αποτελείται από Ανθρώπινο δυναμικό, Διαδικασίες, Βάση Δεδομένων, λογισμικό, υλικός εξοπλισμός και δίκτυο

Πρόβλημα, κατά βήματα είναι ένας τρόπος αναπαράστασης αλγορίθμων με απλά βήματα.

Σύννεφο (cloud) υπηρεσίες που προσφέρουν αποθηκευτικό χώρο βασισμένο στην αρχιτεκτονική του διαδικτύου.

Σύστημα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ) (Database Management System - DBMS) αποτελείται από το σύνολο δεδομένων και τα προγράμματα πρόσβασης στα δεδομένα αυτά.

Τεκμηρίωση είναι το σύνολο των γραπτών κειμένων που συνοδεύει το κάθε λογισμικό.

Τμηματικός προγραμματισμός είναι μια τεχνική όπου τα επαναλαμβανόμενα τμήματα κώδικα γράφονται μόνο μια φορά ως ανεξάρτητες ενότητες και καλούνται όπου και όσες φορές χρειάζεται μέσα στο κυρίως πρόγραμμα.

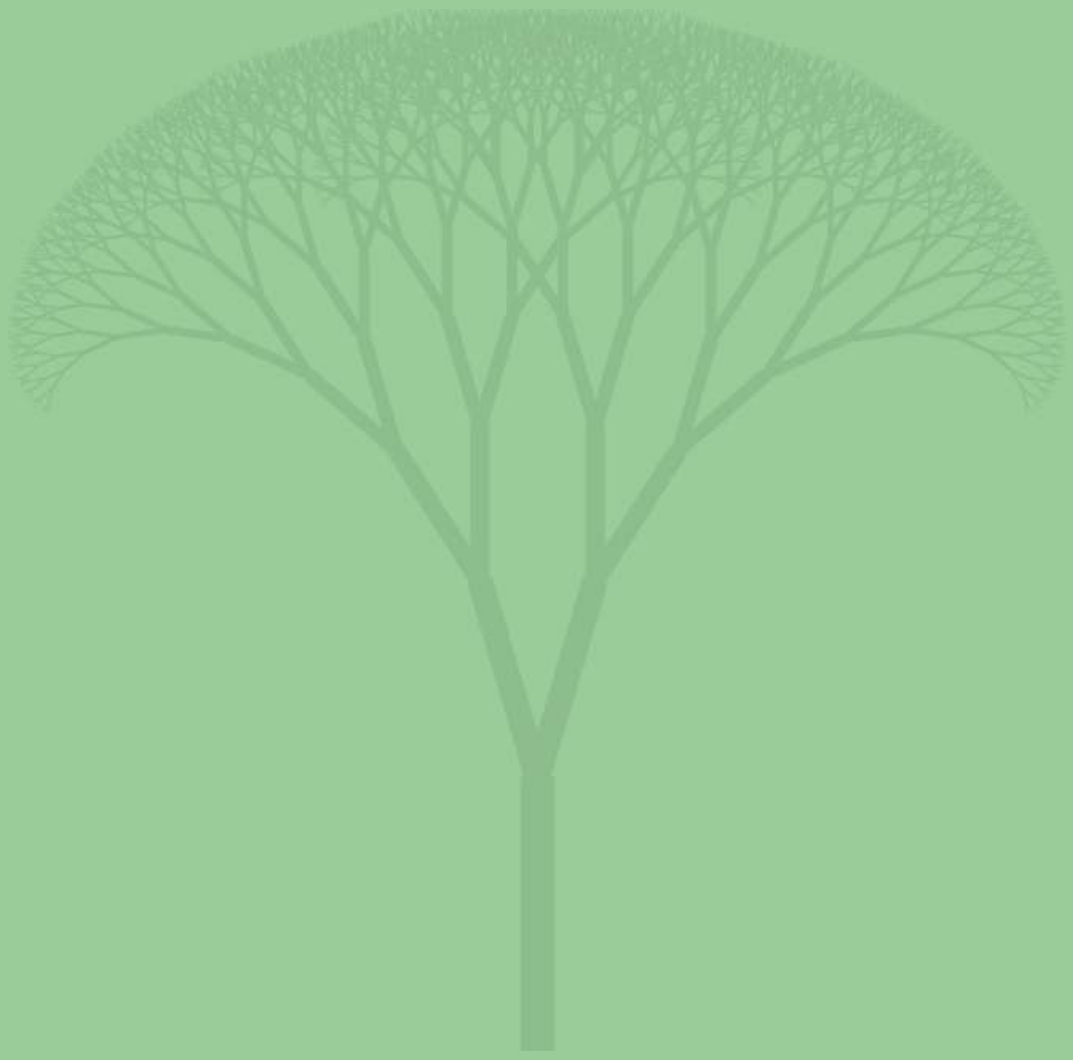
Τοπικά δίκτυα (LAN, local area networks) είναι δίκτυα υπολογιστών που καλύπτουν ένα κτίριο ή ένα κτιριακό συγκρότημα (πχ σχολείο) ή ακόμα και μερικά χιλιόμετρα (πχ πανεπιστημιούπολη).

Υπολογιστικά προβλήματα, είναι τα προβλήματα για τα οποία απαιτείται η διενέργεια υπολογισμών προκειμένου να δοθεί απάντηση σε αυτά.

Φυσική γλώσσα, κατά βήματα είναι ένας τρόπος αναπαράστασης αλγορίθμων με απλά βήματα.

Χαρακτηριστικά Αλγορίθμων: Είσοδος, Έξοδος, Περατότητα, Καθοριστικότητα, Αποτελεσματικότητα.

Ψευδοκώδικας, είναι ένας τρόπος αναπαράστασης αλγορίθμων με περιεκτικές λέξεις και προτάσεις.



ISBN: 978-960-99789-3-4