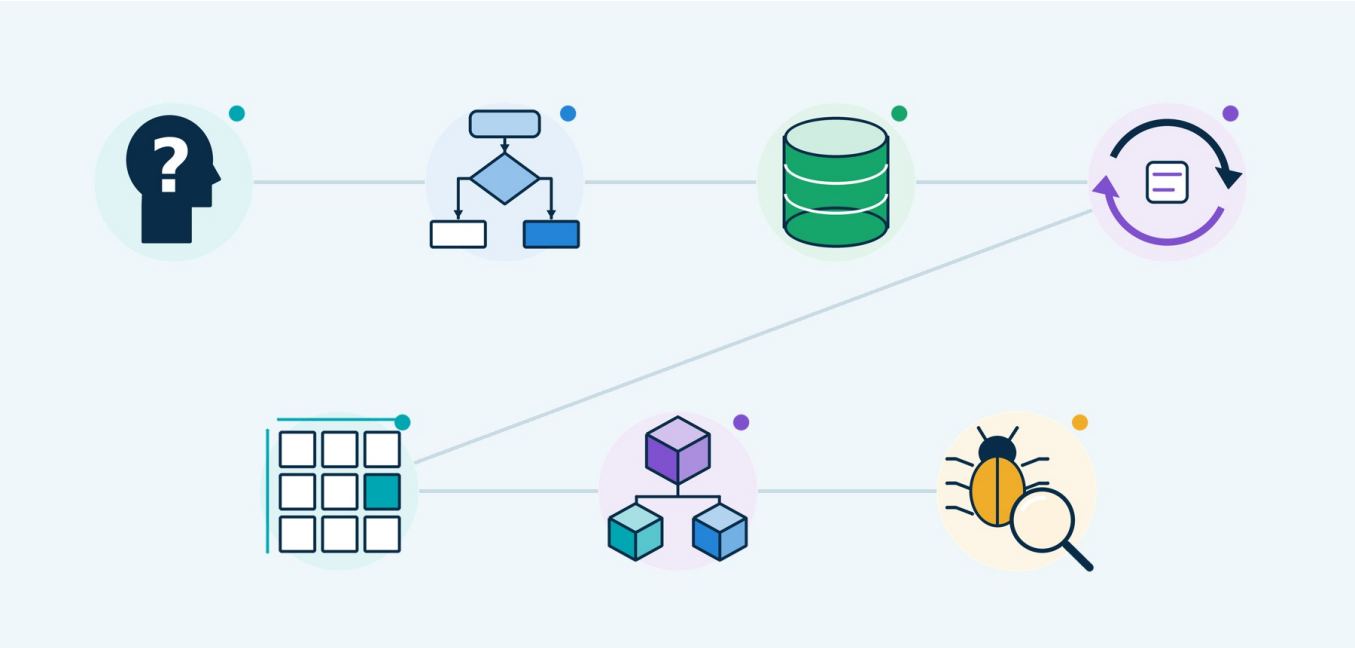
ΠΛΗΡΟΦΟΡΙΚΗ

Ερωτήσεις θεωρίας

Εφ’ όλης της Ύλης



176 ερωτήσεις • 4 επίπεδα αποστήθισης

Σχολικό Έτος 2026 – 2027

Επιμέλεια

Μερεντίτης Νίκος

Περιεχόμενα

Κεφάλαια και ενότητες

Ενότητα	Ερωτήσεις	Σελίδα
Προβλήματα ΚΕΦΑΛΑΙΟ 1ο	001–010	3
Αλγόριθμοι ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η	011–016	5
Δεδομένα και εκφράσεις ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η	017–032	8
Επιλογή και επανάληψη ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η	033–048	16
Ανάλυση και σχεδίαση ΚΕΦΑΛΑΙΟ 4ο • ΕΝΟΤΗΤΑ 2η	049–057	23
Δομές δεδομένων και πίνακες ΚΕΦΑΛΑΙΑ 3ο 9ο	058–084	26
Αρχές προγραμματισμού ΚΕΦΑΛΑΙΟ 6ο	085–105	36
Υποπρογράμματα ΚΕΦΑΛΑΙΟ 10ο	106–123	45
Στοίβα και ουρά ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ	124–127	53
Συνδεδεμένες λίστες ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ	128–139	57
Δέντρα και γράφοι ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ	140–152	66
Αντικειμενοστραφής προγραμματισμός ΕΝΟΤΗΤΑ 4η • ΚΕΦΑΛΑΙΟ 6ο	153–164	82
Λάθη και εκσφαλμάτωση ΕΝΟΤΗΤΑ 5η • ΚΕΦΑΛΑΙΟ 6ο	165–176	91

Δυσκολία αποστήθισης

★☆☆☆	★★☆☆	★★★☆	★★★★
Σύντομη	Μέτρια	Απαιτητική	Πολύ απαιτητική

Ενδεικτική εκτίμηση με βάση το μήκος της απάντησης, τα επιμέρους σημεία και τις διακρίσεις που χρειάζεται να θυμάσαι. Τα τέσσερα αστέρια δείχνουν τη μεγαλύτερη δυσκολία αποστήθισης.



ΚΕΦΑΛΑΙΟ 1ο

Προβλήματα

ΚΕΦΑΛΑΙΟ 1ο

001 ★☆☆☆

Τι είναι πρόβλημα;



Με τον όρο **Πρόβλημα** εννοείται μια κατάσταση η οποία χρήζει αντιμετώπισης, **απαιτεί λύση**, η δε λύση της δεν είναι γνωστή, ούτε προφανής. (σελ. 15 Βιβλίο Μαθητή)

002 ★☆☆☆

Ποιοι παράγοντες συμβάλουν στην κατανόηση ενός προβλήματος;



Η κατανόηση ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων:

1. **Σωστή διατύπωση** του προβλήματος από τον δημιουργό του
2. **Σωστή ερμηνεία** του προβλήματος από εκείνον που πρόκειται να το αντιμετωπίσει. (σελ 17 Βιβλίο Μαθητή)

003 ★☆☆☆

Με ποιους τρόπους μπορεί να διατυπωθεί ένα πρόβλημα;



Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός είτε ο γραπτός. (σελ 17 Βιβλίο Μαθητή)

004 ★★☆☆

Τι ονομάζουμε:
δεδομένο;
Πληροφορία
Επεξεργασία Δεδομένων



Δεδομένο ονομάζουμε οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από ένα τουλάχιστον παρατηρητή με μια από τις **πέντε αισθήσεις** του.



Πληροφορία ονομάζουμε οποιοδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων.



Επεξεργασία δεδομένων είναι η διαδικασία κατά την οποία ένας «μηχανισμός» (ανθρώπινος εγκέφαλος ή υπολογιστής) **δέχεται δεδομένα**, τα επεξεργάζεται σύμφωνα με προκαθορισμένο τρόπο και **αποδίδει πληροφορίες**. (σελ 19 Βιβλίο Μαθητή)

005 ★☆☆☆
Ποιοι είναι οι μηχανισμοί επεξεργασίας δεδομένων;



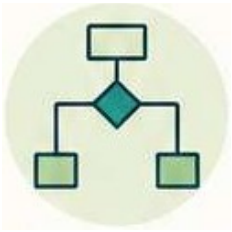

Ο **ανθρώπινος εγκέφαλος** και ο **ηλεκτρονικός υπολογιστής**. (σελ 19 Βιβλίο Μαθητή)

006 ★☆☆☆
Τι ονομάζουμε δομή ενός προβλήματος;



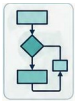
Δομή ενός προβλήματος ονομάζουμε τα **συστατικά του μέρη**, τα επιμέρους τμήματα του το αποτελούν και τον τρόπο που αυτά συνδέονται μεταξύ τους. (σελ 19 Βιβλίο Μαθητή)

007 ★☆☆☆
Ποιοι οι τρόποι περιγραφής και αναπαράστασης της δομής ενός προβλήματος;

Φραστικά και **Διαγραμματικά**. (σελ 20 Βιβλίο Μαθητή)

008 ★☆☆☆
Τι προσφέρει η διαγραμματική αναπαράσταση;



Η διαγραμματική αναπαράσταση προσφέρει:

a. μια απτή **απεικόνιση της δομής** του προβλήματος

b. βοηθάει στην **καλύτερη κατανόηση** του ίδιου του προβλήματος

c. βοηθάει στη **σχεδίαση της λύσης** του. (σελ 20 Βιβλίο Μαθητή)

009 ★☆☆☆
Τι είναι ο καθορισμός απαιτήσεων;



Είναι ο σωστός **εντοπισμός και προσδιορισμός των δεδομένων** και η λεπτομερειακή καταγραφή των ζητουμένων. (σελ 21 Βιβλίο Μαθητή)

010 ★★☆☆
Περιγράψτε τα στάδια αντιμετώπισης ενός προβλήματος;



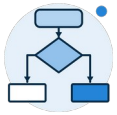
1. **κατανόηση**: όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των **δεδομένων και των ζητούμενων** του προβλήματος



2. **ανάλυση**: όπου το αρχικό πρόβλημα διασπάται σε άλλα επιμέρους απλούστερα προβλήματα.



3. **επίλυση**: όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων. (σελ 25 Βιβλίο Μαθητή)



ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η

Αλγόριθμοι

ΚΕΦΑΛΑΙΟ 2ο – 7ο – 8ο

Ενότητα 3η – Βιβλίο «ΠΛΗΡΟΦΟΡΙΚΗ»

011 ★★☆☆
Τι είναι αλγόριθμος;



Αλγόριθμος είναι μια **πεπερασμένη σειρά ενεργειών**, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος. (σελ 33 Βιβλίο Μαθητή)

012 ★★★★
Ποια κριτήρια πρέπει να ικανοποιεί ο αλγόριθμος;

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια.



a. Είσοδος (input). Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες **πρωτογενείς τιμές** με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.



b. Έξοδος (output). Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.



c. Καθοριστικότητα (definiteness). Κάθε εντολή πρέπει να καθορίζεται **χωρίς καμία αμφιβολία** για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή.



d. Περατότητα (finiteness). Ο αλγόριθμος να τελειώνει μετά από **πεπερασμένα βήματα** εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά **υπολογιστική διαδικασία** (computational procedure).



e. Αποτελεσματικότητα (effectiveness). Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι **απλή**. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και **εκτελέσιμη**. (σελ 33 Βιβλίο Μαθητή)

013 ★☆☆☆
Η έννοια του αλγορίθμου συνδέεται αποκλειστικά με την πληροφορική;



Η έννοια του αλγορίθμου **δεν συνδέεται αποκλειστικά** και μόνο με προβλήματα της Πληροφορικής. Για παράδειγμα μια **συνταγή** για την κατασκευή ενός φαγητού αποτελεί αλγόριθμο. (σελ 34 Βιβλίο Μαθητή)

014 ★★★★
Από ποιες σκοπιές μελετά η πληροφορική τους αλγορίθμους;

Η Πληροφορική, λοιπόν, μπορεί να ορισθεί ως η επιστήμη που μελετά τους αλγορίθμους από τις ακόλουθες σκοπιές (**ΓΥΘΑ**):



1. **Γλωσσών Προγραμματισμού** (programming languages). Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλαδή, χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει **τη δομή και τον αριθμό των εντολών** ενός αλγορίθμου. Γενικά μία γλώσσα που είναι χαμηλότερου επιπέδου (όπως η assembly ή η γλώσσα C) είναι ταχύτερη από μία άλλη γλώσσα που είναι υψηλοτέρου επιπέδου (όπως η Basic ή Pascal). Ακόμη, σημειώνεται ότι διαφορές συναντώνται μεταξύ των γλωσσών σε σχέση με το πότε εμφανίσθηκαν. Για παράδειγμα, παλαιότερα μερικές γλώσσες προγραμματισμού δεν υποστήριζαν την αναδρομή.



2. **Υλικού** (hardware). Η **ταχύτητα εκτέλεσης** ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού, δηλαδή από τον τρόπο που είναι δομημένα σε μία **ενιαία αρχιτεκτονική** τα διάφορα συστατικά του υπολογιστή (δηλαδή ανάλογα με το αν ο υπολογιστής έχει κρυφή μνήμη και πόση, ανάλογα με την ταχύτητα της κύριας και δευτερεύουσας μνήμης κ.ο.κ.).



3. **Θεωρητική** (theoretical). Το ερώτημα που συχνά τίθεται είναι αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική, γιατί προσδιορίζει τα όρια της λύσης που θα βρεθεί σε σχέση με ένα συγκεκριμένο πρόβλημα.



4. **Αναλυτική** (analytical). Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για λειτουργίες CPU και για λειτουργίες εισόδου/εξόδου κ.λπ. (σελ 34 – 35 Βιβλίο Μαθητή)

015 ★★★★
Ποιοι είναι οι τρόποι αναπαράστασης των αλγορίθμων;

Οι κυριότεροι τρόποι αναπαράστασης αλγορίθμων είναι:



a. με **ελεύθερο κείμενο** (free text), που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Έτσι εγκυμονεί τον κίνδυνο ότι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.



b. με **διαγραμματικές τεχνικές** (diagramming techniques), που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί, η πιο παλιά και η πιο γνωστή ίσως, είναι το **διάγραμμα ροής** (flow chart). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση, γι' αυτό και εμφανίζονται όλο και σπανιότερα στη βιβλιογραφία και στην πράξη.

c. με **φυσική γλώσσα** (natural language) **κατά βήματα**. Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, όπως προσδιορίσθηκε προηγουμένως, δηλαδή το κριτήριο του καθορισμού.

d. με **κωδικοποίηση** (coding), δηλαδή με ένα πρόγραμμα γραμμένο είτε σε μία **ψευδογλώσσα** είτε σε κάποια **γλώσσα προγραμματισμού** που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο. (σελ 35 Βιβλίο Μαθητή)

016 ★★☆☆

Ποια είναι τα κυριότερα γεωμετρικά σχήματα σε ένα διάγραμμα ροής και ποια η λειτουργία τους;

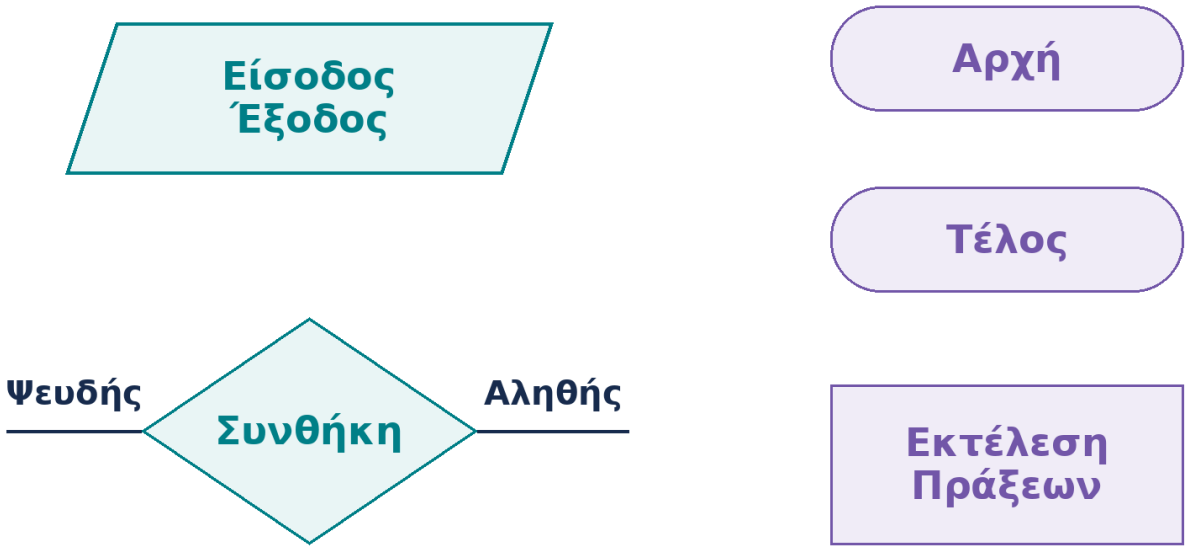
Τα κυριότερα χρησιμοποιούμενα γεωμετρικά σχήματα είναι τα εξής:

a. **έλλειψη**, που δηλώνει **την αρχή και το τέλος** του κάθε αλγορίθμου,

b. **ρόμβος**, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση,

c. **ορθογώνιο**, που δηλώνει την **εκτέλεση μίας ή περισσότερων πράξεων**

d. **πλάγιο παραλληλόγραμμο**, που δηλώνει είσοδο ή έξοδο στοιχείων. Πολλές φορές το σχήμα αυτό μπορεί να διαφοροποιείται προκειμένου να προσδιορίζεται και το είδος της συσκευής απ' όπου γίνεται η είσοδος ή η έξοδος. (σελ 36 Βιβλίο Μαθητή)



ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η

Δεδομένα και εκφράσεις

017 ★★★☆
Τι ονομάζουμε Σταθερές και τι Μεταβλητές;

Οι **σταθερές** (constants) είναι **προκαθορισμένες τιμές** που δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι σταθερές είναι αντίστοιχου τύπου δεδομένων, δηλαδή ακέραιες, πραγματικές, αλφαριθμητικές ή λογικές.

Μεταβλητές (variables). Μια μεταβλητή είναι ένα γλωσσικό αντικείμενο, που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Στη μεταβλητή **εκχωρείται μια τιμή**, η οποία **μπορεί να αλλάζει** κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε αριθμητικές (ακέραιες ή πραγματικές), χαρακτήρες και λογικές.

Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα αντιστοιχούνται από το μεταγλωττιστή σε **συγκεκριμένες θέσεις μνήμης** του υπολογιστή. Η τιμή της μεταβλητής είναι η τιμή που βρίσκεται στην αντίστοιχη θέση μνήμης και όπως αναφέρθηκε μπορεί να μεταβάλλεται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, αυτό που μένει υποχρεωτικά αναλλοίωτο είναι ο τύπος της μεταβλητής. (σελ 129 Βιβλίο Μαθητή)

018 ★★★☆
Τι ονομάζουμε Τελεστές και τί Εκφράσεις;

Τελεστές (operators). Πρόκειται για τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε **αριθμητικούς, λογικούς και συγκριτικούς**.

Εκφράσεις (expressions). Οι εκφράσεις διαμορφώνονται από τους **τελεστέους** (operands), που είναι σταθερές και μεταβλητές και από τους τελεστές. Η διεργασία **αποτίμησης** μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Η τελική τιμή μιας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση των παρενθέσεων. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση. (σελ 37 Βιβλίο Μαθητή)

019 ★★★☆
Ποιοι είναι οι τύποι δεδομένων που υποστηρίζει η ΓΛΩΣΣΑ;

a. **Ακέραιος τύπος**. Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι **θετικοί, αρνητικοί ή μηδέν**. Παραδείγματα ακεραίων είναι οι αριθμοί 1, 3409, 0, -980.



b. **Πραγματικός τύπος**. Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι αριθμοί 3.14159, 2.71828, -112.45, 0.45 είναι πραγματικοί αριθμοί. Και οι πραγματικοί αριθμοί μπορούν να είναι θετικοί, αρνητικοί ή μηδέν.



c. **Χαρακτήρας**. Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και μία σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου μπορούν να περιέχουν οποιονδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο. Παραδείγματα χαρακτήρων είναι 'Κ', 'Κώστας', 'σήμερα είναι Τετάρτη', 'Τα πολλαπλάσια του 15 είναι'. Οι χαρακτήρες πρέπει υποχρεωτικά να βρίσκονται μέσα σε απλά εισαγωγικά, ' '. Τα δεδομένα αυτού του τύπου, επειδή περιέχουν τόσο αλφαβητικούς όσο και αριθμητικούς χαρακτήρες, ονομάζονται συχνά **αλφαριθμητικά**.



d. **Λογικός**. Αυτός ο τύπος δέχεται μόνο δύο τιμές: ΑΛΗΘΗΣ και ΨΕΥΔΗΣ. Οι τιμές αντιπροσωπεύουν αληθείς ή ψευδείς συνθήκες. (σελ 128 Βιβλίο Μαθητή)



020 ★★☆☆

Πότε ένα όνομα ενός αλγορίθμου, ενός προγράμματος ή μίας μεταβλητής, θεωρείται έγκυρο;



Ένα όνομα αλγορίθμου ή μεταβλητής θεωρούνται έγκυρα όταν:

- **Δεν αποτελούν δεσμευμένη λέξη**
- Υποχρεωτικά αρχίζουν με γράμμα (ελληνικό ή λατινικό, πεζό ή κεφαλαίο)
- Περιλαμβάνουν μόνο τον χαρακτήρα '_' (**κάτω παύλα**) και τα **ψηφία 0 – 9**. (σελ 129 Βιβλίο Μαθητή).



Παραδείγματα ονομάτων που είναι αποδεκτά από τη ΓΛΩΣΣΑ είναι: Α, Όνομα, Τιμή, **Τυπική_Απόκλιση**, **Α100**, ΦΠΑ, μέγιστο, Υπολογισμός_Ταχύτητας. Παραδείγματα ονομάτων που δεν είναι αποδεκτά είναι: **100Α**, **Μέση Τιμή**, **Κόστος$**.



021 ★★☆☆

Τι ονομάζουμε εντολή στην ψευδογλώσσα; Ποια είδη εντολών υπάρχουν;



Εντολή ονομάζουμε κάθε μια λέξη της ψευδογλώσσας που προσδιορίζει μια σαφή και συγκεκριμένη ενέργεια.

Υπάρχουν:



- Οι **δηλωτικές εντολές**: οι οποίες δηλώνουν και **δεν εκτελούν** μια ενέργεια



- Οι **εκτελεστέες**: οι οποίες **εκτελούν μια συγκεκριμένη λειτουργία**.



022 ★☆☆☆

Ποιες βασικές αλγοριθμικές δομές χρησιμοποιούνται στην κατασκευή ενός αλγορίθμου;

Οι αλγοριθμικές δομές που χρησιμοποιούνται σε έναν αλγόριθμο είναι:



- Η δομή **ακολουθίας**



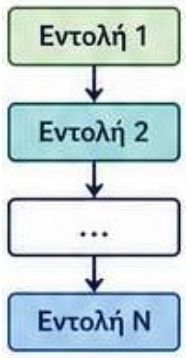
- Η δομή **επιλογής**



- Η δομή **επανάληψης**.

023 ★★☆☆
Ποια είναι τα χαρακτηριστικά της δομής ακολουθίας;

Η ακολουθιακή δομή εντολών (σειριακών βημάτων) χρησιμοποιείται πρακτικά για την αντιμετώπιση απλών προβλημάτων, όπου είναι **δεδομένη η σειρά εκτέλεσης** ενός συνόλου ενεργειών. Ένα απλό παράδειγμα από την καθημερινή ζωή είναι η ακολουθία οδηγιών μίας συνταγής μαγειρικής με στόχο την κατασκευή ενός φαγητού. Τα βήματα και οι ποσότητες που πρέπει να ακολουθηθούν είναι συγκεκριμένα και οι οδηγίες απόλυτα καθορισμένες και σαφείς. Το παράδειγμα που ακολουθεί παρουσιάζει ένα απλό πρόβλημα που επιλύεται με σειριακή εκτέλεση εντολών. Ο αλγόριθμος που χρησιμοποιεί μόνο τη δομή ακολουθίας αποτελείται μόνο από εντολές **εισόδου, εξόδου και εκχώρησης τιμής**. Επίσης εκτελούνται υποχρεωτικά όλες οι εντολές μια φορά και δεν παραλείπεται καμία. (σελ 36 Βιβλίο Μαθητή)



024 ★☆☆☆
Τι ονομάζουμε δεσμευμένες λέξεις;



Μερικές λέξεις χρησιμοποιούνται από την ίδια τη ΓΛΩΣΣΑ για συγκεκριμένους λόγους, όπως οι λέξεις **ΠΡΟΓΡΑΜΜΑ, ΑΚΕΡΑΙΕΣ, ΠΡΑΓΜΑΤΙΚΕΣ, ΑΝ** κ.λπ. Αυτές οι λέξεις δεν μπορούν να χρησιμοποιηθούν ως ονόματα. Οι λέξεις αυτές αποκαλούνται **δεσμευμένες**. (σελ 129 Βιβλίο Μαθητή)

025 ★★☆☆
Ποιοι είναι οι αριθμητικοί τελεστές που υποστηρίζει η ΓΛΩΣΣΑ;

Οι αριθμητικοί τελεστές είναι:

Αριθμητικός Τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός

Αριθμητικός Τελεστής	Πράξη
/	Διαίρεση
MOD	Υπόλοιπο ακέραιας διαίρεσης
DIV	Πηλίκο ακέραιας Διαίρεσης
^	**Ύψωση σε δύναμη**

Ο τελεστής **div** χρησιμοποιείται για τονυπολογισμό του πηλίκου μιας διαίρεσης ακεραίων αριθμών, ενώ ο τελεστής **mod** για το υπόλοιπο.
Π.χ. 7 div 2 = 3 και 7 mod 2 = 1 (σελ 131 Βιβλίο Μαθητή)

026 ★★★☆
Ποιες μαθηματικές συναρτήσεις υποστηρίζει η ΓΛΩΣΣΑ;

Οι συναρτήσεις που υποστηρίζει η ΓΛΩΣΣΑ (και είναι ήδη ενσωματωμένες) είναι:

ΗΜ(Χ)	Υπολογισμός ημιτόνου
ΣΥΝ(Χ)	Υπολογισμός συνημιτόνου
ΕΦ(Χ)	Υπολογισμός εφαπτομένης
Τ_Ρ(Χ)	Υπολογισμός τετραγωνικής ρίζας
ΛΟΓ(Χ)	Υπολογισμός φυσικού λογαρίθμου
Ε(Χ)	Υπολογισμός του e^x
Α_Μ(Χ)	Ακέραιο μέρος του Χ
Α_Τ(Χ)	Απόλυτη τιμή του Χ

(σελ 131 Βιβλίο Μαθητή)

027 ★☆☆☆
Ποια είναι η ιεραρχία των Αριθμητικών Τελεστών;

Οι πράξεις που παρουσιάζονται σε μια έκφραση εκτελούνται σύμφωνα με την επόμενη ιεραρχία

a. **Ύψωση σε δύναμη**
b. **Πολλαπλασιασμός και διαίρεση**
c. **Πρόσθεση και αφαίρεση**

(σελ 131 Βιβλίο Μαθητή)

028 ★★★☆
Ποιοι είναι οι συγκριτικοί τελεστές που υποστηρίζει η ΓΛΩΣΣΑ;

Οι συγκριτικοί τελεστές είναι:

Συγκριτικός Τελεστής	Ελεγχόμενη Σχέση
=	**Ισότητα**
<>	**Ανισότητα**
>	Μεγαλύτερο από
>=	**Μεγαλύτερο ή ίσο**
<	Μικρότερο από
<=	**Μικρότερο ή ίσο**

Παρατηρήσεις:

- Οι συγκρίσεις γίνονται σε δεδομένα **αριθμητικά, αλφαριθμητικά και λογικά**.
- Η σύγκριση μεταξύ δύο αριθμών γίνεται με προφανή τρόπο. Στην περίπτωση των πραγματικών αριθμών θεωρούμε ότι οι αριθμοί μπορούν να έχουν άπειρο αριθμό ψηφίων.
- Η σύγκριση ατομικών χαρακτήρων στηρίζεται στην **αλφαβητική σειρά**, για παράδειγμα το 'α' θεωρείται μικρότερο από το 'β'.
- Η σύγκριση αλφαριθμητικών δεδομένων βασίζεται στη σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρις ότου βρεθεί κάποια διαφορά, για παράδειγμα η λέξη 'κακός' θεωρείται μικρότερη από τη λέξη 'καλός' αφού το γράμμα κ προηγείται του γράμματος λ.
- Η **σύγκριση λογικών** έχει έννοια μόνο στην περίπτωση του ίσου (=) και του διάφορου (<>), αφού οι τιμές που μπορούν να έχουν είναι ΑΛΗΘΗΣ και ΨΕΥΔΗΣ. (σελ 139 Βιβλίο Μαθητή)

029 ★★★☆
Ποιοι είναι οι Λογικοί τελεστές που υποστηρίζει η ΓΛΩΣΣΑ;

Οι λογικοί τελεστές που υποστηρίζει η ΓΛΩΣΣΑ είναι:

1. **ΟΧΙ**
2. **ΚΑΙ**
3. **Ή**

Παρατηρήσεις:

- Η λογική πράξη ή είναι αληθής όταν **οποιαδήποτε από τις δύο προτάσεις** είναι αληθής.
- Η λογική πράξη και είναι αληθής όταν και οι δύο προτάσεις είναι αληθείς.
- Η λογική πράξη όχι είναι αληθής όταν η πρόταση που την ακολουθεί είναι ψευδής. Δηλαδή αντιστρέφει την λογική τιμή της πρότασης.

Ο επόμενος πίνακας δίνει τις τιμές των τριών αυτών λογικών πράξεων για όλους τους συνδυασμούς τιμών.

Πρόταση Α	Πρόταση Β	Α ή Β	Α και Β	όχι Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Αληθής	Ψευδής	Ψευδής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

(σελ 43 Βιβλίο Μαθητή)



030 ★★★☆
Ποια είναι η ιεραρχία των τελεστών σε μία έκφραση;

Η ιεραρχία με την οποία εκτελούνται οι πράξεις σε μία έκφραση είναι:



1. Πρώτα εκτελούνται όλοι οι **αριθμητικοί τελεστές**. Μέσα στην ομάδα των αριθμητικών τελεστών έχουμε πρώτα τη **δύναμη**, μετά τον **πολλαπλασιασμό, τη διαίρεση, το div και το mod** (ότι συναντάμε πρώτα από αριστερά προς δεξιά) και τέλος την **πρόσθεση και την αφαίρεση** (ότι συναντάμε πρώτα από αριστερά προς δεξιά).



2. Έπειτα εκτελούνται οι **συγκριτικοί τελεστές** (ότι συναντάμε πρώτα από αριστερά προς δεξιά)



3. Τέλος εκτελούνται οι **λογικοί τελεστές**. Μέσα στην ομάδα των λογικών τελεστών έχουμε σε προτεραιότητα το **ΟΧΙ**, έπειτα το **ΚΑΙ** ενώ τελευταίος είναι λογικός τελεστής **Ή**.



Φυσικά η προτεραιότητα αναιρείται εφόσον έχουμε παρενθέσεις.



031 ★★★★

Να περιγράψετε τη λειτουργία και να αναφέρετε τα βασικά χαρακτηριστικά
Της εντολής εκχώρησης
Της εντολής εισόδου
Της εντολής εξόδου.

Τα βασικά χαρακτηριστικά και η λειτουργία των εντολών περιγράφονται παρακάτω:

a. Μια **εντολή εκχώρησης** σε καμία περίπτωση δεν πρέπει να εκλαμβάνεται ως εξίσωση. Στην εξίσωση το αριστερό μέλος ισούται με το δεξιό, ενώ στην εντολή εκχώρησης η τιμή του δεξιού μέλους εκχωρείται, μεταβιβάζεται, αποδίδεται στη μεταβλητή του αριστερού μέλους. Για το λόγο αυτό ως τελεστής εκχώρησης χρησιμοποιείται το σύμβολο ← προκειμένου να διαφοροποιείται από το ίσον (=). Κατά την εκτέλεση της εντολής **πρώτα υπολογίζεται η έκφραση** που υπάρχει στα δεξιά της εντολής , ενώ το αποτέλεσμα που προκύπτει, **εκχωρείται στη μεταβλητή που βρίσκεται αριστερά** της εντολής.



b. Τα δεδομένα εισάγονται κατά τη διάρκεια της εκτέλεσης του προγράμματος από μία μονάδα εισόδου, για παράδειγμα το πληκτρολόγιο. Η ΓΛΩΣΣΑ υποστηρίζει για την εισαγωγή δεδομένων από το πληκτρολόγιο την εντολή **ΔΙΑΒΑΣΕ**. Η εντολή ΔΙΑΒΑΣΕ ακολουθείται πάντοτε από ένα ή περισσότερα ονόματα μεταβλητών. Αν υπάρχουν περισσότερες από μία μεταβλητές, τότε αυτές **χωρίζονται με κόμμα** (,). Κατά την εκτέλεση του προγράμματος η εντολή ΔΙΑΒΑΣΕ διακόπτει την εκτέλεσή του και το πρόγραμμα περιμένει την εισαγωγή από το πληκτρολόγιο τιμών, που θα εκχωρηθούν στις μεταβλητές. Μετά την ολοκλήρωση της εντολής η εκτέλεση του προγράμματος συνεχίζεται με την επόμενη εντολή.



c. Τα αποτελέσματα γράφονται σε μία μονάδα εξόδου, για παράδειγμα την οθόνη. Η ΓΛΩΣΣΑ υποστηρίζει για την εμφάνιση των αποτελεσμάτων την εντολή **ΓΡΑΨΕ**. Η εντολή ΓΡΑΨΕ έχει ως αποτέλεσμα την **εμφάνιση τιμών** στη μονάδα εξόδου. Συσκευή εξόδου μπορεί να είναι η οθόνη του υπολογιστή. Η λίστα των στοιχείων μπορεί να περιέχει **σταθερές τιμές και ονόματα μεταβλητών**. Κατά την εκτέλεση του προγράμματος η εντολή ΓΡΑΨΕ προκαλεί την εμφάνιση στην οθόνη των σταθερών τιμών. Όταν κάποιο όνομα μεταβλητής περιέχεται στη λίστα, τότε αρχικά ανακτάται η τιμή της και στη συνέχεια η τιμή αυτή εμφανίζεται στην οθόνη. Η χρήση της εντολής ΓΡΑΨΕ είναι κυρίως η **εμφάνιση μηνυμάτων** από τον υπολογιστή, καθώς και αποτελεσμάτων που περιέχονται στις μεταβλητές. (σελ 133 Βιβλίο Μαθητή)



032 ★★☆☆
Τί είναι τα σχόλια και ποια η λειτουργία του συμβόλου &;

Τα **σχόλια** είναι επεξηγηματικές φράσεις για τα διάφορα τμήματα κώδικα. Αν ο πρώτος χαρακτήρας είναι το **θαυμαστικό (!)**, σημαίνει ότι αυτή η γραμμή περιέχει σχόλια και <u>όχι εκτελέσιμες εντολές</u>. Σχόλια μπορούν να τοποθετηθούν οπουδήποτε μέσα στο πρόγραμμα, από την αρχική εντολή «ΠΡΟΓΡΑΜΜΑ όνομα» μέχρι και την τελική εντολή «ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ».

Παραδείγματα σχολίων

```
! Αυτό είναι σχόλιο
! Υπολογισμός αθροίσματος
! x ← x + 1
```

Κάθε εντολή γράφεται σε **ξεχωριστή γραμμή**. Αν μία εντολή πρέπει να συνεχιστεί και στην επόμενη γραμμή, τότε <u>ο πρώτος χαρακτήρας αυτής της γραμμής</u> πρέπει να είναι ο χαρακτήρας &. (σελ 134 Βιβλίο Μαθητή)

Παράδειγμα

```
ΓΡΑΨΕ 'Το άθροισμα είναι:',
& άθροισμα
```



ΚΕΦΑΛΑΙΑ 2ο 7ο 8ο • ΕΝΟΤΗΤΑ 3η

Επιλογή και επανάληψη

033 ★★★★
Ποια είδη δομών επιλογής υποστηρίζει η ΓΛΩΣΣΑ;

Η ΓΛΩΣΣΑ υποστηρίζει 3 είδη δομών επιλογής:

Απλή δομή επιλογής	ΑΝ συνθήκη ΤΟΤΕ εντολή-1 εντολή-2 ... εντολή-ν ΤΕΛΟΣ_ΑΝ	
Σύνθετη δομή επιλογής	ΑΝ συνθήκη ΤΟΤΕ εντολή-1 εντολή-2 ... εντολή-ν ΑΛΛΙΩΣ εντολή-1 εντολή-2 ... εντολή-ν ΤΕΛΟΣ_ΑΝ	
Πολλαπλή Δομή Επιλογής	ΑΝ συνθήκη -1 ΤΟΤΕ εντολή-1 εντολή-2 ... εντολή-ν **ΑΛΛΙΩΣ_ΑΝ** συνθήκη-2 ΤΟΤΕ εντολή-1 εντολή-2 ... εντολή-ν ..... ΑΛΛΙΩΣ εντολή-1 εντολή-2 ... εντολή-ν ΤΕΛΟΣ_ΑΝ	Αν οι εναλλακτικές περιπτώσεις επιλογής είναι πολλές, μπορεί να χρησιμοποιηθεί η εντολή **ΕΠΙΛΕΞΕ** (ως πολλαπλή δομή επιλογής), η γενική μορφή της οποίας είναι: **ΕΠΙΛΕΞΕ** έκφραση ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_1 εντολές_1 ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_2 εντολές_2 ..... ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ εντολές_αλλιώς **ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ***

* Στην εντολή αυτή οι **λίστες τιμών** που συνοδεύουν κάθε περίπτωση μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές ή περιοχή τιμών από-έως. Η χρήση της εντολής

ΕΠΙΛΕΞΕ λόγω της συμπαγούς δομής της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματισμό. (σελ 140 – 145 Βιβλίο Μαθητή)

034 ★★☆☆
Τι ονομάζουμε εμφωλευμένα Αν;

Εμφωλευμένα ΑΝ ονομάζονται δύο ή περισσότερες εντολές της μορφής ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ που περιέχονται η μία μέσα στην άλλη.

(Η χρήση εμφωλευμένων εντολών ΑΝ οδηγεί συνήθως σε πολύπλοκες δομές που αυξάνουν την **πιθανότητα του λάθους** καθώς και τη **δυσκολία κατανόησης** του προγράμματος) (σελ 142 Βιβλίο Μαθητή)

035 ★★★★
Ποια είδη δομών επανάληψης περιλαμβάνει η ΓΛΩΣΣΑ;

Η ΓΛΩΣΣΑ υποστηρίζει 3 είδη δομών επανάληψης:



Όνομα	Σύνταξη	Λειτουργία
Όσο	**ΟΣΟ συνθήκη ΕΠΑΝΑΛΑΒΕ** εντολή-1 εντολή-2 ... εντολή-ν **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	Ελέγχεται η συνθήκη και αν είναι Αληθής, εκτελούνται οι εντολές που βρίσκονται ανάμεσα στις ΟΣΟ_ΕΠΑΝΑΛΑΒΕ και **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**. Στη συνέχεια ελέγχεται πάλι η συνθήκη και αν ισχύει, εκτελούνται πάλι οι ίδιες εντολές. Όταν η λογική έκφραση γίνει Ψευδής, τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά το **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**. Εφόσον μετά από κάθε επανάληψη ελέγχεται εκ νέου η συνθήκη, πρέπει υποχρεωτικά μέσα στο βρόχο να υπάρχει μία εντολή, η οποία να μεταβάλλει την τιμή της μεταβλητής που ελέγχεται με τη συνθήκη. Σε αντίθετη περίπτωση η επανάληψη δε θα τερματίζεται και θα εκτελείται συνεχώς.

Όνομα	Σύνταξη	Λειτουργία
Μέχρις_ότου	**ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** εντολή-1 εντολή-2 ... εντολή-ν **ΜΕΧΡΙΣ_ΟΤΟΥ** λογική-έκφραση	Εκτελούνται οι εντολές μεταξύ των **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** και **ΜΕΧΡΙΣ_ΟΤΟΥ**. Στη συνέχεια ελέγχεται η λογική έκφραση και αν δεν ισχύει (είναι ψευδής), τότε οι εντολές που βρίσκονται ανάμεσα στις ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ και ΜΕΧΡΙΣ_ΟΤΟΥ, εκτελούνται πάλι. Ελέγχεται ξανά η λογική έκφραση και αν δεν ισχύει, επαναλαμβάνεται η εκτέλεση των ίδιων εντολών. Όταν η λογική έκφραση γίνει Αληθής, τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά από την ΜΕΧΡΙΣ_ΟΤΟΥ.
Για...από...μέχρι	ΓΙΑ μεταβλητή ΑΠΟ τιμή1 ΜΕΧΡΙ τιμή2 **ΜΕ ΒΗΜΑ** τιμή3 εντολή-1 εντολή-2 ... εντολή-ν **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	Οι εντολές του βρόχου εκτελούνται για όλες τις τιμές της μεταβλητής από την αρχική τιμή μέχρι την τελική τιμή, αυξανόμενες με την τιμή του βήματος. Αν το βήμα είναι ίσο με 1, τότε **παραλείπεται**.

(σελ 146 – 150 Βιβλίο Μαθητή)

036 ★☆☆☆

Τι ονομάζουμε Βρόχο;

Βρόχος αποκαλείται ένα τμήμα προγράμματος το οποίο εκτελείται επαναληπτικά.

(σελ 45 Βιβλίο Μαθητή)

037 ★☆☆☆

Ποιος βρόχος ονομάζεται «Ατέρμων Βρόχος»;

«**Ατέρμων Βρόχος**» ονομάζεται ο βρόχος που δεν τερματίζει ποτέ.

(σελ 45 Βιβλίο Μαθητή)

038 ★★☆☆
Πού και πώς ελέγχεται η συνθήκη μέσα σε μία δομή επανάληψης;

Σε μια επαναληπτική δομή η συνθήκη που ελέγχεται καθορίζει την έξοδο από την επανάληψη.

1. Στην **ΟΣΟ** η συνθήκη ελέγχεται στην αρχή της επανάληψης. Η επανάληψη εκτελείται όσο η συνθήκη είναι **ΑΛΗΘΗΣ**.
2. Στη **ΜΕΧΡΙΣ_ΟΤΟΥ** η συνθήκη ελέγχεται στο τέλος της επανάληψης. Η επανάληψη εκτελείται όσο η συνθήκη είναι **ΨΕΥΔΗΣ**.
3. Στην εντολή **ΓΙΑ..ΑΠΟ...ΜΕΧΡΙ** η επανάληψη εκτελείται για **προκαθορισμένο αριθμό επαναλήψεων**.

(σελ 148 Βιβλίο Μαθητή)

039 ★★☆☆
Τι ονομάζουμε «Τιμή Φρουρό»;

Η χρήση τιμών για τον **τερματισμό μίας επαναληπτικής διαδικασίας**, όπως στο παράδειγμα η αυθαίρετη επιλογή του 0, είναι συνήθης στον προγραμματισμό. Η τιμή αυτή ορίζεται από τον προγραμματιστή και αποτελεί μια σύμβαση για το τέλος του προγράμματος. Η τιμή αυτή είναι τέτοια, ώστε να μην είναι λογικά σωστή για το πρόβλημα, για παράδειγμα η τιμή 0 αποκλείεται από τις μετρήσεις σύμφωνα με την εκφώνηση του παραδείγματος. Η τιμή αυτή συχνά αποκαλείται "**τιμή φρουρός**".

(σελ 147 Βιβλίο Μαθητή)

040 ★★☆☆
Ποια είναι η γενικότερη δομή επανάληψης; Γιατί είναι πολύ σημαντική;

Η γενικότερη δομή επανάληψης υλοποιείται στη ΓΛΩΣΣΑ με την εντολή **ΟΣΟ... ΕΠΑΝΑΛΑΒΕ**. Σε αυτή, η συνθήκη που ελέγχει την επανάληψη βρίσκεται **στην αρχή της επανάληψης** και ο βρόχος επαναλαμβάνεται συνεχώς, όσο η συνθήκη αυτή ισχύει. Με τη δομή αυτή μπορούν να εκφραστούν όλες οι επαναλήψεις και γι' αυτό η εντολή ΟΣΟ... ΕΠΑΝΑΛΑΒΕ είναι η σημαντικότερη από όλες τις εντολές επανάληψης. Χαρακτηριστικό της επανάληψης αυτής είναι ότι ο αριθμός των επαναλήψεων δεν είναι γνωστός, ούτε μπορεί να υπολογιστεί πριν από την εκτέλεση του προγράμματος.

(σελ 147 Βιβλίο Μαθητή)

041 ★★★☆
Πότε χρησιμοποιείται η «ΜΕΧΡΣ_ΟΤΟΥ»;

Πολύ συχνά η ίδια επαναληπτική διαδικασία μπορεί να γραφεί εξίσου σωστά χρησιμοποιώντας είτε τη δομή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ είτε τη δομή **ΜΕΧΡΙΣ_ΟΤΟΥ** και είναι προσωπική επιλογή του προγραμματιστή ποια από τις δυο θα χρησιμοποιήσει. Υπάρχουν όμως περιπτώσεις όπου η χρήση της εντολής ΜΕΧΡΙΣ_ΟΤΟΥ οδηγεί σε απλούστερα και πιο ευκολονόητα προγράμματα. Γενικά σε περιπτώσεις όπου η επανάληψη θα συμβεί υποχρεωτικά μία φορά, είναι προτιμότερη η χρήση της ΜΕΧΡΙΣ_ΟΤΟΥ.

Χαρακτηριστική περίπτωση όπου προτιμάται η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ είναι:

- στον **έλεγχο αποδεκτών τιμών** (**έλεγχος εγκυρότητας**)

- στην επιλογή από **προκαθορισμένες απαντήσεις ή μενού**.
- Στην ερώτηση του χρήστη για τον **τερματισμό ή τη συνέχιση** της επανάληψης.

(σελ 148 Βιβλίο Μαθητή)

042 ★★☆☆

Πότε χρησιμοποιείται η «ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ»;

Η εντολή επανάληψης **ΓΙΑ** χρησιμοποιείται όταν γνωρίζουμε εκ των προτέρων τον αριθμό των επαναλήψεων που πρέπει να εκτελεστούν (είναι προκαθορισμένος).

Παρατήρηση: Κάθε επανάληψη που εκτελείται με μία εντολή ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ, μπορεί να υλοποιηθεί και με τη χρήση των βασικών εντολών επανάληψης **ΟΣΟ…ΕΠΑΝΑΛΑΒΕ** και **ΜΕΧΡΙΣ_ΟΤΟΥ**.

(σελ 150 Βιβλίο Μαθητή)

043 ★★☆☆

Πως ορίζεται ο βρόχος στην εντολή ΟΣΟ, πως στην ΜΕΧΡΙΣ_ΟΤΟΥ και πως στην εντολή ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ;



Στην εντολή **ΟΣΟ** ο βρόχος ορίζεται από την εντολή ΟΣΟ μέχρι και το **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**. Στην εντολή **ΜΕΧΡΙΣ_ΟΤΟΥ** ο βρόχος ορίζεται από την εντολή **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** μέχρι και το ΜΕΧΡΙΣ_ΟΤΟΥ. Στην εντολή **ΓΙΑ**, ο βρόχος ορίζεται από την εντολή ΓΙΑ μέχρι και το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ.

(σελ 45 Βιβλίο Μαθητή)

044 ★★★☆

Ποιες είναι οι διαφορές μεταξύ ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ;

Οι διαφορές είναι:

	Όσο	Μέχρις_ότου
1.	Στην Όσο **πρώτα ελέγχεται η συνθήκη** και έπειτα εκτελούνται οι εντολές του βρόχου.	Στην Μέχρις_ότου **πρώτα εκτελούνται οι εντολές** εντός βρόχου και στο τέλος ελέγχεται η συνθήκη.
2.	Στην Όσο ο βρόχος επαναλαμβάνεται όσο η συνθήκη είναι **αληθής**.	Στην Μέχρις_ότου ο βρόχος επαναλαμβάνεται όσο η συνθήκη είναι **ψευδής**.
3.	Η Όσο **μπορεί να μην εκτελεστεί καμία φορά** εφόσον η συνθήκη είναι εξαρχής **ψευδής**.	Η Μέχρις_ότου **εκτελείται οπωσδήποτε μια φορά**.
4.	Στην Όσο η συνθήκη ελέγχεται στην αρχή της επανάληψης.	Στη Μέχρις_ότου η συνθήκη ελέγχεται στο τέλος της επανάληψης.

Παρατήρηση:

- Το πλήθος των επαναλήψεων της δομής ΟΣΟ ισούται με τον αριθμό των φορών που θα εξεταστεί η συνθήκη, μείον μια (δηλαδή αν η συνθήκη εξετάζεται Ν φορές, το πλήθος επαναλήψεων είναι Ν-1).

- Το πλήθος των επαναλήψεων της δομής **ΜΕΧΡΙΣ_ΟΤΟΥ** ισούται με τον αριθμό των φορών που θα εξεταστεί η συνθήκη (δηλαδή αν η συνθήκη εξετάζεται Ν φορές, το πλήθος επαναλήψεων είναι Ν).

(σελ 148 Βιβλίο Μαθητή)

045 ★★☆☆
Ποιοι είναι οι κανόνες χρήσης των εμφωλευμένων βρόχων;

Στη χρήση των εμφωλευμένων βρόχων ισχύουν οι παρακάτω κανόνες:

1. Ο **εσωτερικός βρόχος** πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό.



2. Ο βρόχος που **ξεκινάει τελευταίος**, πρέπει να **ολοκληρώνεται πρώτος**.
3. Η είσοδος σε κάθε βρόχο υποχρεωτικά γίνεται **από την αρχή του**.
4. Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.

(σελ 150 Βιβλίο Μαθητή)

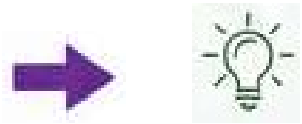
046 ★★★☆
Τι ορίζουμε ως «ολίσθηση προς αριστερά» και τι ως «ολίσθηση προς τα δεξιά»;



Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με **δυαδική μορφή**, δηλαδή 0 και 1 πχ ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό **00010001** του δυαδικού συστήματος.
Αν μετακινήσουμε τα ψηφία αυτά κατά μία θέση προς τα αριστερά, δηλαδή αν προσθέσουμε ένα 0 στο τέλος του αριθμού και αγνοήσουμε το αρχικό 0, τότε προκύπτει ο αριθμός **00100010** του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 34 του δεκαδικού συστήματος. Με παρόμοιο τρόπο, αν μετακινήσουμε τα ψηφία κατά μία θέση δεξιά, δηλαδή αποκόψουμε το τελευταίο ψηφίο 1 και θεωρήσουμε ένα ακόμη αρχικό 0, τότε προκύπτει ο αριθμός **00001000** του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος.

$00010001_2 = 17_{10}$ ↓ $00100010_2 = 34_{10}$

$00010001_2 = 17_{10}$ ↓ $00001000_2 = 8_{10}$



Άρα η **ολίσθηση προς τα αριστερά** ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η **ολίσθηση προς τα δεξιά** ισοδυναμεί με ακέραια διαίρεση διά δύο.

(σελ 48 Βιβλίο Μαθητή)

047 ★★☆☆
Γιατί χρησιμοποιείται ο «Πολλαπλασιασμός αλά Ρωσικά» στους υπολογιστές;



Η μέθοδος αυτή χρησιμοποιείται πρακτικά στους υπολογιστές, γιατί υλοποιείται πολύ πιο απλά απ’ ό,τι ο γνωστός μας χειρωνακτικός τρόπος πολλαπλασιασμού. Πιο συγκεκριμένα, απαιτεί **πολλαπλασιασμό επί δύο, διαίρεση διά δύο και πρόσθεση**. Σε επίπεδο, λοιπόν, κυκλωμάτων υπολογιστή ο πολλαπλασιασμός επί δύο και η διαίρεση διά δύο μπορούν να υλοποιηθούν ταχύτατα με μία απλή εντολή ολίσθησης (shift) (αριστερά και δεξιά αντίστοιχα). Αυτός είναι ο λόγος που ο πολλαπλασιασμός αλά ρωσικά είναι προτιμότερος απ’ ό,τι ο χειρωνακτικός τρόπος πολλαπλασιασμού δύο ακεραίων.

(σελ 50 Βιβλίο Μαθητή)

048 ★★★★
Να γράψετε τον αλγόριθμο του Πολλαπλασιασμού αλά Ρωσικά σε φυσική γλώσσα κατά βήματα;



Αλγόριθμος: Πολλαπλασιασμός δύο θετικών ακεραίων (αλά ρωσικά)	
Είσοδος:	Δύο ακέραιοι Μ1 και Μ2, όπου **M1, M2 ≥ 1**
Έξοδος:	Το γινόμενο P=M1*M2
Βήμα 1	Θέσε **P=0**
Βήμα 2	Αν **M2>0**, τότε πήγαινε στο Βήμα 3, αλλιώς πήγαινε στο Βήμα 7
Βήμα 3	Αν ο Μ2 είναι περιττός, τότε θέσε **P=P+M1**
Βήμα 4	Θέσε **M1=M1*2**
Βήμα 5	Θέσε **M2=M2/2** (θεώρησε μόνο το ακέραιο μέρος)
Βήμα 6	Πήγαινε στο Βήμα 2
Βήμα 7	Τύπωσε τον P

(σελ 51 Βιβλίο Μαθητή)



ΚΕΦΑΛΑΙΟ 4ο • ΕΝΟΤΗΤΑ 2η

Ανάλυση και σχεδίαση

ΚΕΦΑΛΑΙΟ 4ο

Ενότητα 2η – Βιβλίο «ΠΛΗΡΟΦΟΡΙΚΗ»



049 ★☆☆☆

Κάθε πρόβλημα έχει μια μόνο λύση;



Πολλές φορές ένα πρόβλημα επιδέχεται **αρκετές λύσεις** οι οποίες μπορούν να προέλθουν από **διαφορετικές τεχνικές και μεθόδους**. Έτσι πρέπει να γίνεται μια καλή ανάλυση του προβλήματος ώστε να προτείνεται συγκεκριμένη μεθοδολογία βημάτων ώστε να προκύψουν έξυπνες και αποδοτικές λύσεις.

(σελ 77 Βιβλίο Μαθητή)



050 ★★☆☆

Ποια βήματα περιλαμβάνει η ανάλυση ενός προβλήματος σε ένα σύγχρονο προγραμματιστικό περιβάλλον;



Η ανάλυση περιλαμβάνει:

1. την **καταγραφή της υπάρχουσας πληροφορίας** για το πρόβλημα,
2. την **αναγνώριση των ιδιαιτεροτήτων** του προβλήματος,
3. την **αποτύπωση των συνθηκών και προϋποθέσεων** υλοποίησής του
4. την **πρόταση επίλυσης** με χρήση κάποιας μεθόδου,
5. την **τελική επίλυση** με χρήση **υπολογιστικών συστημάτων**.

(σελ 77 Βιβλίο Μαθητή)

051 ★★★☆

Σε ποιες ερωτήσεις πρέπει να δοθούν απαντήσεις κατά την ανάλυση ενός προβλήματος;

1. Ποια είναι τα **δεδομένα και το μέγεθος** του προβλήματος.
2. Ποιες είναι οι **συνθήκες που πρέπει να πληρούνται** για την επίλυση του προβλήματος.
3. Ποια είναι η **πλέον αποδοτική μέθοδος** επίλυσής τους (**σχεδίαση αλγορίθμου**).
4. **Πώς θα καταγραφεί η λύση** σε ένα πρόβλημα (π.χ. σε ψευδογλώσσα).
5. Ποιος είναι ο **τρόπος υλοποίησης** στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. **επιλογή γλώσσας προγραμματισμού**).

(σελ 77 Βιβλίο Μαθητή)

052 ★☆☆☆

Γιατί είναι απαραίτητη η ανάλυση του προβλήματος ;

Η ανάλυση κάθε προβλήματος είναι απαραίτητη, έτσι ώστε να αναζητηθεί η **πλέον κατάλληλη μέθοδος** που να παρέχει τη ζητούμενη λύση, όσο γίνεται ταχύτερα και με το λιγότερο δυνατό κόστος σε υπολογιστικούς πόρους.

(σελ 78 Βιβλίο Μαθητή)

053 ★☆☆☆

Ποια προβλήματα ονομάζονται «συγγενή»;

«**Συγγενή**» ονομάζονται τα προβλήματα που μπορούν να αναλυθούν με παρόμοιο τρόπο και να αντιμετωπισθούν με αντίστοιχες μεθόδους και τεχνικές.

(σελ 78 Βιβλίο Μαθητή)

054 ★★★☆

Γιατί παρουσιάζουν ενδιαφέρον οι μέθοδοι ανάλυσης και επίλυσης προβλημάτων;

Οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον για τους εξής λόγους:

1. παρέχουν ένα **γενικό πρότυπο** κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
2. μπορούν να αναπαρασταθούν με **κοινές δομές δεδομένων και ελέγχου** (που υποστηρίζονται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού),
3. παρέχουν τη δυνατότητα καταγραφής των χρονικών και "χωρικών" απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

(σελ 78-79 Βιβλίο Μαθητή)

055 ★★☆☆
Τι είναι η μέθοδος «Διαίρει και Βασίλευε»;

Η μέθοδος "**Διαίρει και Βασίλευε**" (divide and conquer) είναι μια μέθοδος σχεδίασης αλγορίθμων στην οποία εντάσσονται οι τεχνικές που υποδιαιρούν ένα πρόβλημα σε **μικρότερα υπο-προβλήματα**, που έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα αλλά είναι μικρότερα σε μέγεθος. Με όμοιο τρόπο, τα υπο-προβλήματα αυτά μπορούν να διαιρεθούν σε ακόμη μικρότερα υποπροβλήματα κ.ο.κ. και τελικά καταλήγουμε στην επίλυση του αρχικού μας προβλήματος.



(σελ 80 Βιβλίο Μαθητή)

056 ★★☆☆
Τι είναι η προσέγγιση λύσης «Από πάνω προς τα Κάτω» (top-down method);



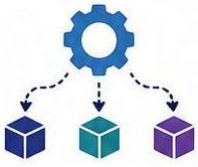
Η επίλυση ενός προβλήματος έγκειται στη **σταδιακή επίλυση** των όσο το δυνατόν μικρότερων υπο-προβλημάτων, ώστε τελικά να καταλήξουμε στη συνολική λύση του αρχικού ευρύτερου προβλήματος. Αυτή η προσέγγιση ονομάζεται **από επάνω προς τα κάτω** (top-down).

(σελ 80 Βιβλίο Μαθητή)

057 ★★★☆
Ποια είναι τα βήματα που εφαρμόζει η μέθοδος «διαίρει και βασίλευε»;

Η περιγραφή της μεθόδου σχεδίασης αλγορίθμων «Διαίρει και Βασίλευε» μπορεί να αποδοθεί με τα επόμενα βήματα:

1. Δίνεται για επίλυση **ένα στιγμιότυπο ενός προβλήματος**.
2. **Υποδιαιρείται** το στιγμιότυπο του προβλήματος σε υπο-στιγμιότυπα **του ίδιου προβλήματος**.



3. Δίνεται **ανεξάρτητη λύση** σε κάθε ένα υπο-στιγμιότυπο.
4. **Συνδυάζονται όλες οι μερικές λύσεις** που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.



Ένας κλασικός αλγόριθμος που ακολουθεί τη φιλοσοφία της μεθόδου Διαίρει και Βασίλευε είναι η **Δυαδική αναζήτηση** η οποία χρησιμοποιείται μόνο στην περίπτωση ταξινομημένου πίνακα.

(σελ 80 Βιβλίο Μαθητή)



ΚΕΦΑΛΑΙΑ 3ο 9ο

Δομές δεδομένων και πίνακες

ΚΕΦΑΛΑΙΟ 3ο – 9ο

058 ★☆☆☆
Τι μελετάει η «Θεωρία Πληροφοριών»



Η «**Θεωρία πληροφοριών**» έχει ως στόχο τη μελέτη της μέτρησης, της κωδικοποίησης και της μετάδοσης της πληροφορίας.

(σελ 55 Βιβλίο Μαθητή)

059 ★★★★
Από ποιες σκοπιές μελετάει η πληροφορική τα Δεδομένα;

Η Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές (**ΓΥΔΑ**):



1. **Γλωσσών προγραμματισμού**. Οι γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων **τύπων** (types) **μεταβλητών** (variables) για να περιγράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον υπολογιστή.



2. **Υλικού**. Το υλικό (hardware), δηλαδή η μηχανή, επιτρέπει στα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες **αναπαραστάσεις** (representations). Τέτοιες μορφές είναι η δυαδική, ο κώδικας ASCII, ο κώδικας EBCDIC, το συμπλήρωμα του 1 ή του 2 κ.λπ.



3. **Δομών δεδομένων**. Δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών. Για παράδειγμα, μία τέτοια δομή είναι η **εγγραφή** (record), που μπορεί να περιγράφει ένα είδος, ένα πρόσωπο κ.λπ. Η εγγραφή αποτελείται από τα **πεδία** (fields) που αποθηκεύουν χαρακτηριστικά (attributes) διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κ.λπ. Άλλη μορφή δομής δεδομένων είναι το **αρχείο** που αποτελείται από ένα σύνολο εγγραφών. Μία επιτρεπτή λειτουργία σε ένα αρχείο είναι η σειριακή προσπέλαση όλων των εγγραφών του.



4. **Ανάλυσης δεδομένων**. Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των **Βάσεων Δεδομένων** (Databases), της **Μοντελοποίησης Δεδομένων** (Data Modelling) και ττης **Αναπαράστασης Γνώσης** (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

(σελ 55 Βιβλίο Μαθητή)

060 ★☆☆☆
Τί είναι «Δομή Δεδομένων»;

Δομή Δεδομένων είναι ένα **σύνολο αποθηκευμένων δεδομένων** που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

(σελ 56 Βιβλίο Μαθητή)

061 ★★★★
Ποιες είναι οι βασικές λειτουργίες (πράξεις) επί των Δομών Δεδομένων;

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο **κόμβων** (nodes). Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες (**ΠΕΔΑΤΑΣΔ**):

1. **Προσπέλαση** (access), πρόσβαση σε έναν κόμβο με σκοπό να **εξετασθεί ή να τροποποιηθεί** το περιεχόμενό του.
2. **Εισαγωγή** (insertion), δηλαδή η **προσθήκη νέων κόμβων** σε μία υπάρχουσα δομή.
3. **Διαγραφή** (deletion), που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας **κόμβος αφαιρείται** από μία δομή.
4. **Αναζήτηση** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν **μια δεδομένη ιδιότητα**.
5. **Ταξινόμηση** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά **αύξουσα ή φθίνουσα σειρά**.
6. **Αντιγραφή** (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής **αντιγράφονται σε μία άλλη δομή**.
7. **Συγχώνευση** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται **σε μία ενιαία δομή**.
8. **Διαχωρισμός** (separation), που αποτελεί την **αντίστροφη πράξη της συγχώνευσης**.

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή. Συνηθέστατα παρατηρείται το φαινόμενο μία δομή δεδομένων να είναι αποδοτικότερη από μία άλλη δομή με κριτήριο κάποια λειτουργία, για παράδειγμα την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, για παράδειγμα την εισαγωγή.

(σελ 56 Βιβλίο Μαθητή)

062 ★★☆☆
Υπάρχει εξάρτηση μεταξύ αλγορίθμων και δομών δεδομένων;

Υπάρχει **μεγάλη εξάρτηση** μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται τη δομή. Μάλιστα, το πρόγραμμα πρέπει να θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα. Η παρατήρηση αυτή δικαιολογεί την εξίσωση που διατυπώθηκε το 1976 από τον Wirth (που σχεδίασε και υλοποίησε τη γλώσσα Pascal)

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

(σελ 57 Βιβλίο Μαθητή)

063 ★★★☆
Σε ποιες κατηγορίες διακρίνονται οι δομές δεδομένων; Ποια είναι τα χαρακτηριστικά – διαφορές τους;

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις **στατικές** (static) και τις **δυναμικές** (dynamic).

	Στατικές	Δυναμικές
1	Το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται **κατά τη στιγμή του προγραμματισμού** τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος.	Στηρίζονται στην τεχνική της λεγόμενης **δυναμικής παραχώρησης μνήμης** (dynamic memory allocation)
2	**Έχουν σταθερό μέγεθος.**	**Δεν έχουν σταθερό μέγεθος**, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα
3	Τα στοιχεία τους αποθηκεύονται σε **συνεχόμενες θέσεις μνήμης**.	Τα στοιχεία τους δεν αποθηκεύονται σε **συνεχόμενες θέσεις μνήμης**.

(σελ 57-58 Βιβλίο Μαθητή)

064 ★★☆☆
Τι ορίζουμε ως Πίνακα;

Πίνακας είναι ένα σύνολο αντικειμένων ίδιου τύπου, τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται **στοιχείο του πίνακα**. Η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο ή περισσότερους **δείκτες σε αγκύλες**. (σελ 58 και 156 Βιβλίο Μαθητή)

065 ★★★☆
Τι διαστάσεις μπορεί να έχει ένας πίνακας; Που δηλώνεται ένας πίνακας;



Ένας πίνακας μπορεί να είναι **μονοδιάστατος**, αλλά στη γενικότερη περίπτωση μπορεί να είναι **δισδιάστατος**, **τρισδιάστατος** και γενικά **ν-διάστατος** πίνακας. Όσον αφορά στους δισδιάστατους πίνακες σημειώνεται ότι, αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται **τετραγωνικός** (square) και γενικά συμβολίζεται ως πίνακας **NxN**. Μάλιστα μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι ένας μονοδιάστατος πίνακας, όπου κάθε θέση του περιέχει ένα νέο μονοδιάστατο πίνακα.

Ένας πίνακας δηλώνεται στο τμήμα δήλωσης των μεταβλητών. Η δήλωση γίνεται με το όνομα του πίνακα και μέσα σε αγκύλες τον μεγαλύτερο αριθμό στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας ώστε να δεσμευτούν όλες οι αντίστοιχες συνεχόμενες θέσεις μνήμης.

(σελ 58 Βιβλίο Μαθητή)

066 ★★☆☆

Πότε ένας πίνακας ονομάζεται Μονοδιάστατος και πότε Δισδιάστατος;

Ένας πίνακας που χρησιμοποιεί έναν μόνο δείκτη για να γίνεται αναφορά στα στοιχεία του, ονομάζεται **μονοδιάστατος πίνακας**. Για παράδειγμα Π[10] είναι ένας πίνακας με 10 στοιχεία.



Ένας πίνακας που χρησιμοποιεί δυο δείκτες για να γίνεται αναφορά στα στοιχεία του, ονομάζεται **δισδιάστατος πίνακας**. Για παράδειγμα Π[10,20] είναι ένα δισδιάστατος πίνακας με **10 γραμμές και 20 στήλες**.



067 ★★☆☆

Ποια η διαφορά μεταξύ ενός πίνακα και ενός στοιχείου του πίνακα;

Ο πίνακας είναι ένα **σύνολο από αντικείμενα του ίδιου τύπου** και στα οποία αναφερόμαστε με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα αυτά αποτελεί ένα **στοιχείο του πίνακα**. Αντίθετα κάθε στοιχείο ενός πίνακα είναι ένα κομμάτι του πίνακα αυτού, και προσδιορίζεται από το όνομα του πίνακα και ένα δείκτη. (σελ 58 Βιβλίο Μαθητή)



068 ★★★☆

Τι είδους δομή είναι ένας Πίνακας; Πότε και γιατί τον χρησιμοποιούμε;

Ο πίνακας είναι μια **Στατική Δομή Δεδομένων** και συνεπώς έχει όλα τα χαρακτηριστικά των στατικών δομών. Τα στοιχεία του αποθηκεύονται στην **κύρια μνήμη** του υπολογιστή, ενώ τα αποθηκευμένα δεδομένα διατηρούνται μέχρι το τέλος της εκτέλεσης του αλγορίθμου ή του προγράμματος.

Η απόφαση για τη χρήση ή όχι πίνακα για τη διαχείριση των δεδομένων είναι κυρίως θέμα εμπειρίας στον προγραμματισμό. Γενικά, αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης, τότε η χρήση πινάκων βοηθάει ή συχνά είναι απαραίτητη για την επίλυση του προβλήματος. Σε άλλη περίπτωση μπορεί να αποφεύγεται η χρήση τους.

(σελ 160 Βιβλίο Μαθητή)

069 ★★☆☆
Ποιες λειτουργίες των δομών δεδομένων εφαρμόζονται στους πίνακες;



Στους πίνακες **δεν μπορούν να εφαρμοστούν** και οι 8 λειτουργίες οι οποίες εφαρμόζονται πάνω σε μία δομή δεδομένων. Ειδικότερα δεν μπορούν να εφαρμοστούν οι λειτουργίες της **Εισαγωγής** (ενός νέου κόμβου)και της **Διαγραφής** (ενός ήδη υπάρχοντος κόμβου). Και αυτό γιατί ο Πίνακας είναι μια στατική δομή δεδομένων, με σταθερό μέγεθος.

070 ★★★☆
Ποια είναι τα μειονεκτήματα από τη χρήση των πινάκων;

1. **Οι πίνακες απαιτούν μνήμη.** Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση μεγάλων πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.
2. Οι πίνακες **περιορίζουν τις δυνατότητες του προγράμματος**. Στο προηγούμενο πρόγραμμα του υπολογισμού των στατιστικών μεγεθών, υπάρχει ανώτατο όριο στο πλήθος των αριθμών ίσο με 100. Αυτό γιατί οι πίνακες είναι στατικές δομές και το μέγεθός τους πρέπει να δηλώνεται στην αρχή του προγράμματος, ενώ παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση του προγράμματος.

(σελ 160 Βιβλίο Μαθητή)

071 ★★☆☆
Ποιες είναι οι Τυπικές Επεξεργασίες των πινάκων;

Τα προγράμματα τα οποία χρησιμοποιούν πίνακες πολύ συχνά απαιτούν συγκεκριμένες επεξεργασίες στα στοιχεία του πίνακα. Οι τυπικές αυτές επεξεργασίες είναι:

1. **Υπολογισμός αθροισμάτων** στοιχείων του πίνακα.
2. Εύρεση του **μέγιστου ή του ελάχιστου** στοιχείου.
3. **Ταξινόμηση** των στοιχείων του πίνακα.
4. **Αναζήτηση** ενός στοιχείου του πίνακα.
5. **Συγχώνευση** δύο πινάκων.

(σελ 166 Βιβλίο Μαθητή)

072 ★★★☆

Γιατί είναι σημαντική η λειτουργία της ταξινόμησης; αναφέρετε παραδείγματα από την καθημερινότητα.



Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία πολύ σημαντική λειτουργία που ονομάζεται **ταξινόμηση** (sorting) ή διάταξη (ordering). Η σειρά αυτή είναι η **αύξουσα τάξη** (ascending sequence) ή η **φθίνουσα σειρά** (descending sequence) της τιμής των μεγεθών προς ταξινόμηση. Σκοπός της ταξινόμησης είναι να διευκολυνθεί στη συνέχεια η αναζήτηση των στοιχείων του ταξινομημένου πίνακα. Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, **λεξικά, τηλεφωνικούς καταλόγους**. Η ταξινόμηση δεν εφαρμόζεται σε δεδομένα Λογικού τύπου (Αληθής, Ψευδής).

073 ★★★★

Ποιος είναι ο ορισμός της Ταξινόμησης;

Δοθέντων των στοιχείων a_1, a_2, ..., a_n η **ταξινόμηση** συνίσταται στη **μετάθεση** (permutation) της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά a_{k1}, a_{k2}, ..., a_{kn} έτσι ώστε, δοθείσης μίας **συνάρτησης διάταξης** (ordering function), f, να ισχύει:

$f(a_{k1}) \leq f(a_{k2}) \leq ... \leq f(a_{kn})$

(σελ 65 Βιβλίο Μαθητή)

074 ★★☆☆

Ποια είναι η πιο απλή μέθοδος ταξινόμησης; Πως λειτουργεί αυτή;

Η μέθοδος της **ταξινόμησης ευθείας ανταλλαγής** (straight exchange sort) βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων, μέχρις ότου διαταχθούν όλα τα στοιχεία. Η μέθοδος είναι γνωστή ως **ταξινόμηση φυσαλίδας** (bubblesort). Αυτή η μέθοδος ταξινόμησης είναι η πιο απλή από όλες τις μεθόδους ταξινόμησης, αλλά ταυτόχρονα και η λιγότερο αποδοτική, καθώς είναι η πιο αργή από όλες τις γνωστές μεθόδους ταξινόμησης.

(σελ 66 Βιβλίο Μαθητή)

075 ★☆☆☆

Με ποια κριτήρια επιλέγεται ο καλύτερος αλγόριθμος ταξινόμησης σε ένα πίνακα;

Τα κριτήρια για την επιλογή του καλύτερου αλγορίθμου ταξινόμησης είναι:

1. Η **αρχική διάταξη** των στοιχείων του πίνακα, δηλαδή αν ο πίνακας είναι εν μέρη ταξινομημένος ή είναι πλήρως αταξινόμητος.
2. Το **πλήθος των στοιχείων** του.

076 ★★☆☆

Ποιους άλλους αλγορίθμους ταξινόμησης γνωρίζετε και ποια τα χαρακτηριστικά τους;

1. **Ταξινόμηση με Επιλογή**: Βασίζεται στην επιλογή του μικρότερου ή του μεγαλύτερου στοιχείου από αυτά που δεν έχουν ταξινομηθεί και στην τοποθέτησή του στην αρχή του πίνακα.
2. **Ταξινόμηση Ευθείας Εισαγωγής ή Παρεμβολής**
3. **Γρήγορη Ταξινόμηση – Quick Sort** (ο πιο γρήγορος αλγόριθμος ταξινόμησης).

(σελ 67 Βιβλίο Μαθητή)

077 ★★☆☆

Τι γνωρίζετε για τις Δομές Δεδομένων Δευτερεύουσας Μνήμης;

Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το μέγεθος της **κύριας μνήμης δεν επαρκεί** για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη **δευτερεύουσα μνήμη**, δηλαδή κυρίως στο σκληρό δίσκο. Οι ειδικές αυτές δομές ονομάζονται αρχεία (files).

(σελ 66 Βιβλίο Μαθητή)

078 ★★☆☆

Ποια διαφορά υπάρχει ανάμεσα στις δομές Κύριας και Δευτερεύουσας Μνήμης;



Η **κύρια μνήμη** είναι η **RAM** ενώ η **δευτερεύουσα** είναι ο **σκληρός δίσκος**. Στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή. Έτσι, τα δεδομένα των αρχείων διατηρούνται ακόμη και μετά τον τερματισμό ενός προγράμματος, κάτι που δεν συμβαίνει στην περίπτωση των δομών της κύριας μνήμης, όπως είναι οι πίνακες, όπου τα δεδομένα χάνονται όταν τελειώσει το πρόγραμμα.

(σελ 66 Βιβλίο Μαθητή)

079 ★★★★

Τι γνωρίζετε για τις έννοιες «εγγραφή», «πεδία», «πρωτεύον κλειδί» και «δευτερεύον κλειδί»;



Τα στοιχεία ενός αρχείου ονομάζονται **εγγραφές** (records), όπου κάθε εγγραφή αποτελείται από ένα ή περισσότερα **πεδία** (fields), που ταυτοποιούν την εγγραφή, και από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής. Για παράδειγμα, έστω η εγγραφή ενός μαθητή με πεδία: Αριθμός Μητρώου, Ονοματεπώνυμο, Έτος Γέννησης, Τάξη, Τμήμα. Το πεδίο Αριθμός Μητρώου ταυτοποιεί την εγγραφή και ονομάζεται **πρωτεύον κλειδί** (primary key) ή απλά κλειδί. Το πεδίο Ονοματεπώνυμο επίσης ταυτοποιεί την εγγραφή και γι' αυτό αποκαλείται **δευτερεύον κλειδί** (secondary keys), αν υπάρχει πρωτεύον κλειδί. Το πρόβλημα της αναζήτησης (searching) μίας εγγραφής με βάση την τιμή του πρωτεύοντος ή ενός δευτερεύοντος κλειδιού σε αρχεία είναι ιδιαίτερα ενδιαφέρον, αν ληφθεί υπ' όψη η μεγάλη ποικιλία των χαρακτηριστικών τόσο της δομής (για παράδειγμα, στατική ή δυναμική, τρόπος οργάνωσης, μέσο αποθήκευσης κ.λπ.), του τύπου των δεδομένων (για παράδειγμα, ακέραιοι, κείμενο, χαρτογραφικά δεδομένα, χρονοσειρές κ.λπ.), όσο και της αναζήτησης (δηλαδή, με βάση το πρωτεύον ή το δευτερεύον κλειδί κ.λπ.).

Παράδειγμα εγγραφής μαθητή

Αριθμός Μητρώου	Ονοματεπώνυμο	Έτος Γέννησης	Τάξη	Τμήμα
12345	Γεωργίου Νίκος	2006	Α'	1

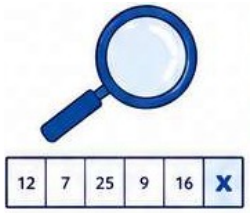
Πρωτεύον κλειδί — Δευτερεύον κλειδί — Άλλα πεδία (χαρακτηριστικά)

(σελ 66 Βιβλίο Μαθητή)

080 ★★☆☆

Τι γνωρίζετε για την Αναζήτηση ενός στοιχείου σε πίνακα; Τί τύπου δεδομένα αναζητάμε;

Με τη λειτουργία της **αναζήτησης** προσπελαύνονται τα στοιχεία των πινάκων και εξετάζεται αν υπάρχει μέσα σε αυτά μια τιμή που μας ενδιαφέρει.



Η αναζήτηση εφαρμόζεται σε πίνακες οι οποίοι περιλαμβάνουν **αριθμητικά δεδομένα** (ακέραιους ή πραγματικούς αριθμούς) αλλά και **αλφαριθμητικά δεδομένα** (χαρακτήρες).

(σελ 64 Βιβλίο Μαθητή)

081 ★★☆☆

Ποιες είναι οι πιο γνωστές μέθοδοι αναζήτησης και ποιες είναι οι διαφορές τους;

Οι πιο γνωστές μέθοδοι αναζήτησης είναι:

1. Η **Σειριακή Αναζήτηση**: Είναι η **πιο απλή μέθοδος αναζήτησης** και η λιγότερο αποτελεσματική. Χρησιμοποιείται υποχρεωτικά σε ταξινομημένους πίνακες, αλλά μπορεί να χρησιμοποιηθεί και σε πίνακες οι οποίοι δεν είναι ταξινομημένοι.
2. Η **Δυαδική Αναζήτηση**: Είναι πιο αποδοτική από τη σειριακή αναζήτηση και χρησιμοποιείται μόνο σε ταξινομημένους πίνακες.

(σελ 64 Βιβλίο Μαθητή)

082 ★★☆☆

Πότε δικαιολογείται η χρήση της Σειριακής Αναζήτησης;

Δικαιολογείται όταν:

1. Ο πίνακας είναι **μη ταξινομημένος**
2. Ο πίνακας είναι **μικρού μεγέθους** (αριθμός στοιχείων <= 20)
3. Η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια.

(σελ 64 Βιβλίο Μαθητή)

083 ★☆☆☆

Ποιος πίνακας ονομάζεται Δισδιάστατος;

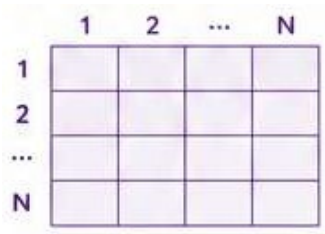
Ο πίνακας που χρησιμοποιεί δυο δείκτες για την αναφορά των στοιχείων του, λέγεται **Δισδιάστατος**.



(σελ 58 Βιβλίο Μαθητή)

084 ★☆☆☆

Πότε ένας δισδιάστατος πίνακας λέγεται Τετραγωνικός;



Ένας δισδιάστατος πίνακας λέγεται **Τετραγωνικός** όταν ο αριθμός των γραμμών είναι ίδιος με τον αριθμό των στηλών του. Συμβολίζεται ως **NxN**.

(σελ 58 Βιβλίο Μαθητή)

ΚΕΦΑΛΑΙΟ 6ο

Αρχές προγραμματισμού

ΚΕΦΑΛΑΙΟ 6ο

085 ★☆☆☆
Ποια στάδια περιλαμβάνει η επίλυση ενός προβλήματος με υπολογιστή;

Περιλαμβάνονται:

1. Ο **ακριβής προσδιορισμός** του προβλήματος.
2. Η **ανάπτυξη του κατάλληλου αλγορίθμου**.
3. Η **διατύπωση του αλγορίθμου** σε μορφή κατανοητή από τον υπολογιστή.

(σελ 105 Βιβλίο Μαθητή)

086 ★★☆☆
Τί ονομάζουμε Πρόγραμμα;

Είναι ένα **σύνολο εντολών** που πρέπει να εκτελεστούν από τον υπολογιστή. Το πρόγραμμα, το οποίο γράφεται σε κάποια **γλώσσα προγραμματισμού**, δεν είναι απλά η υλοποίηση του αλγορίθμου, αλλά βασικό στοιχείο του είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων ενεργεί.

(σελ 105 Βιβλίο Μαθητή)

087 ★★☆☆
Τι ονομάζουμε προγραμματισμό;

Προγραμματισμός είναι η **διατύπωση των αλγορίθμων** σε τέτοια μορφή, ώστε να μπορούν να υλοποιηθούν από τον υπολογιστή. Ο προγραμματισμός είναι αυτός που δίνει την εντύπωση ότι οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν τα πολύπλοκα προβλήματα.

(σελ 105 Βιβλίο Μαθητή)

088 ★★★☆
Ποιες στοιχειώδης ενέργειες – πράξεις μπορεί να εκτελέσει ένας υπολογιστής;

Ο υπολογιστής, ως γνωστόν, είναι μία μηχανή που καταλαβαίνει μόνο **δύο καταστάσεις**, οι οποίες αντιπροσωπεύονται με δύο αριθμούς, το **μηδέν και το ένα**, τα ψηφία του δυαδικού συστήματος. Το μόνο πράγμα που κάνει ο υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολουθίες αυτών των δύο ψηφίων, αλλά αυτές τις ενέργειες τις εκτελεί με ασύλληπτη ταχύτητα. Ο υπολογιστής μπορεί απλά **να αποθηκεύει στη μνήμη** τις ακολουθίες των δυαδικών ψηφίων, **να τις ανακτά**, να κάνει **στοιχειώδεις αριθμητικές πράξεις** με αυτές και **να τις συγκρίνει**.

(σελ 105 Βιβλίο Μαθητή)

089 ★☆☆☆
Για ποιο λόγο αναπτύχθηκαν οι γλώσσες προγραμματισμού;

Οι **γλώσσες προγραμματισμού** αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου με τον υπολογιστή.

(σελ 106 Βιβλίο Μαθητή)

090 ★★★☆
Ποια η διαφορά μεταξύ Φυσικών και Τεχνητών Γλωσσών;

Οι γλώσσες προγραμματισμού, που είναι **τεχνητές γλώσσες**, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις **φυσικές γλώσσες**.

Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η **δυνατότητα εξέλιξής τους**. Οι φυσικές γλώσσες **εξελίσσονται συνεχώς**, νέες λέξεις δημιουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν με την πάροδο του χρόνου και αυτό γιατί η γλώσσα χρησιμοποιείται για την επικοινωνία μεταξύ ανθρώπων, που εξελίσσονται και αλλάζουν ανάλογα με τις εποχές και τον κοινωνικό περίγυρο. Αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από **στασιμότητα**, αφού κατασκευάζονται συνειδητά για ένα συγκεκριμένο σκοπό. Ωστόσο συχνά οι γλώσσες προγραμματισμού βελτιώνονται και μεταβάλλονται από τους δημιουργούς τους, με σκοπό να διορθωθούν αδυναμίες ή να καλύψουν μεγαλύτερο εύρος εφαρμογών ή τέλος να ακολουθήσουν τις νέες εξελίξεις.

(σελ 115 - 116 Βιβλίο Μαθητή)

091 ★★★★
Ποια είναι τα στοιχεία που προσδιορίζουν μία γλώσσα; Τί καλείται Αλφάβητο, τί Λεξιλόγιο, τί Γραμματική και τί Συντακτικό;

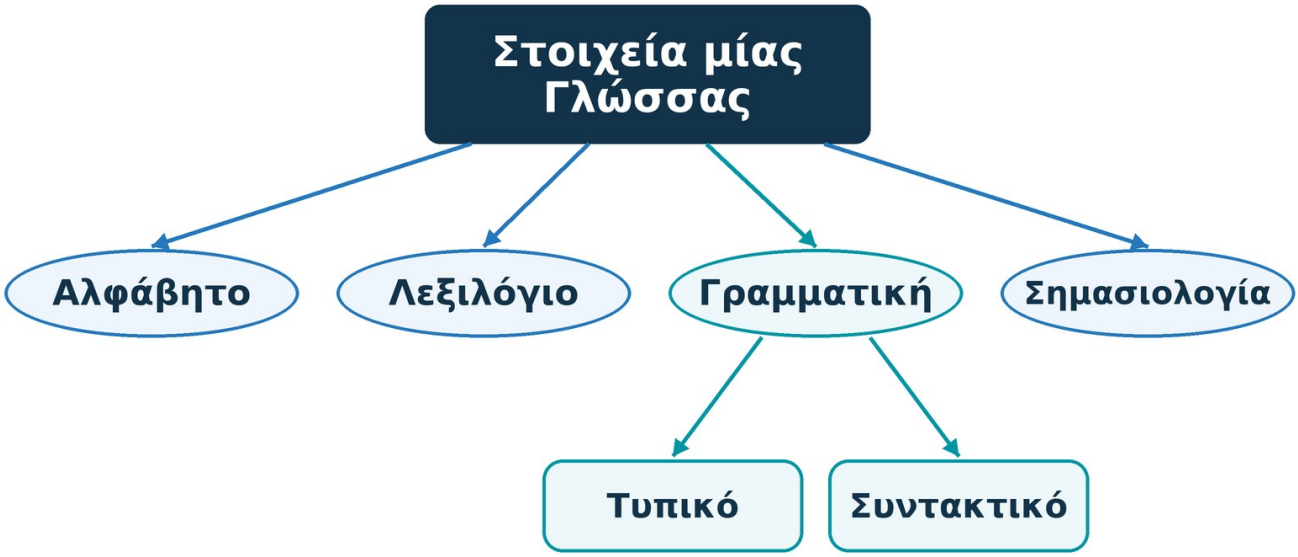
Τα στοιχεία που προσδιορίζουν μια γλώσσα είναι: το αλφάβητο, το λεξιλόγιο, η γραμματική και η σημασιολογία μιας γλώσσας

1. **Αλφάβητο** μίας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα.
2. Το **λεξιλόγιο** αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, τις **λέξεις που είναι δεκτές** από τη γλώσσα.
3. Η **Γραμματική** αποτελείται από το **τυπικό** και το **συντακτικό**.

a. **Τυπικό** είναι το σύνολο των κανόνων που ορίζει τις **μορφές** με τις οποίες μία λέξη είναι αποδεκτή.

b. **Συντακτικό** είναι το σύνολο των κανόνων που καθορίζει τη νομιμότητα της **διάταξης** και της **σύνδεσης των λέξεων** της γλώσσας για τη δημιουργία προτάσεων.

4. Η **σημασιολογία (Semantics)** είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μία γλώσσα.



(σελ 115 Βιβλίο Μαθητή)

092 ★★★☆
Τι είναι ιεραρχική σχεδίαση και ποιος ο σκοπός της;

Η τεχνική της **ιεραρχικής σχεδίασης** και επίλυσης ή η διαδικασία σχεδίασης "από **Επάνω προς τα κάτω**" όπως συχνά ονομάζεται (**top-down program design**) περιλαμβάνει τον καθορισμό των βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο, και στη συνέχεια τη διάσπαση των λειτουργιών αυτών σε όλο και **μικρότερες λειτουργίες**, μέχρι το τελευταίο επίπεδο που οι λειτουργίες είναι πολύ άπλες, ώστε να επιλυθούν εύκολα.

Σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση λοιπόν του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

(σελ 116 Βιβλίο Μαθητή)

093 ★★☆☆
Τι είναι τμηματικός προγραμματισμός;

Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον **τμηματικό προγραμματισμό**. Μετά την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί **ανεξάρτητη ενότητα (module)**, που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.

(σελ 116 Βιβλίο Μαθητή)

094 ★☆☆☆
Ποια τα πλεονεκτήματα του τμηματικού προγραμματισμού;

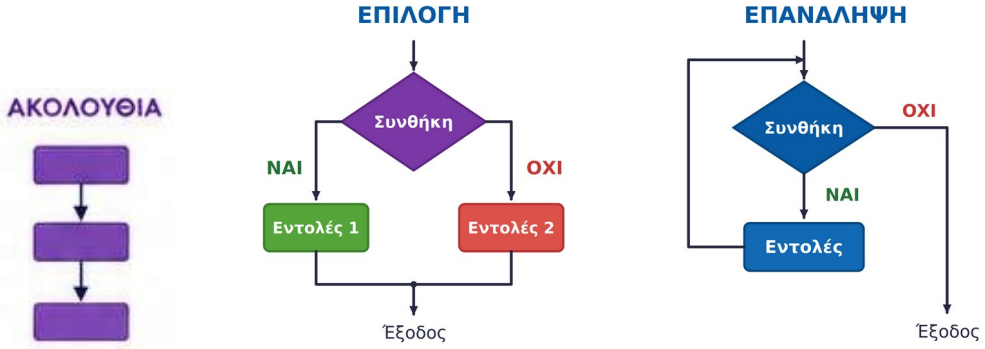
Ο τμηματικός προγραμματισμός

1. διευκολύνει τη **δημιουργία του προγράμματος**,
2. **μειώνει τα λάθη**
3. επιτρέπει την ευκολότερη **παρακολούθηση, κατανόηση και συντήρηση** του προγράμματος από τρίτους.

(σελ 117 Βιβλίο Μαθητή)

095 ★★★★
Τι είναι δομημένος προγραμματισμός; (Ποιες είναι οι αρχές του;)

Ο δομημένος προγραμματισμός στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών, **τη δομή της ακολουθίας**, **τη δομή της επιλογής** και **τη δομή της επανάληψης**. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο μία είσοδο και μόνο μία έξοδο.



Ο δομημένος προγραμματισμός αναπτύχθηκε από την ανάγκη να υπάρχει μία **κοινή μεθοδολογία** στην ανάπτυξη των προγραμμάτων και τη **μείωση των εντολών GOTO** που χρησιμοποιούνται στο πρόγραμμα. Η εντολή GOTO έχει ως αποτέλεσμα την **αλλαγή της ροής του προγράμματος**, της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη. Όλες οι σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν το δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν τη χρήση του GOTO περιττή. Για λόγους όμως συμβατότητας με τις παλιότερες εκδόσεις τους καθώς και για λόγους συντήρησης παλιών προγραμμάτων, μερικές τη διατηρούν στο ρεπερτόριο των εντολών τους.

(σελ 118 - 119 Βιβλίο Μαθητή)

096 ★★☆☆
Ποιος είναι ο σκοπός του δομημένου προγραμματισμού;

Ο δομημένος προγραμματισμός δεν είναι απλώς ένα είδος προγραμματισμού, είναι μία **μεθοδολογία σύνταξης προγραμμάτων** που έχει σκοπό:

1. να βοηθήσει τον προγραμματιστή στην **ανάπτυξη σύνθετων προγραμμάτων**,
2. να **μειώσει τα λάθη**,
3. να εξασφαλίσει την **εύκολη κατανόηση** των προγραμμάτων και
4. να διευκολύνει τις **διορθώσεις και τις αλλαγές** σε αυτά.

(σελ 119 Βιβλίο Μαθητή)

097 ★★★☆
Ποια τα πλεονεκτήματα του δομημένου προγραμματισμού;

Τα πλεονεκτήματα του δομημένου προγραμματισμού είναι:

1. Δημιουργία **απλούστερων προγραμμάτων**
2. **Άμεση μεταφορά** των αλγορίθμων σε προγράμματα.
3. Διευκόλυνση **ανάλυσης του προγράμματος σε τμήματα**.
4. **Περιορισμός των λαθών** κατά την ανάπτυξη του προγράμματος.
5. Διευκόλυνση στην **ανάγνωση και κατανόηση** του προγράμματος από τρίτους.
6. Ευκολότερη **διόρθωση και συντήρηση**.

(σελ 119 Βιβλίο Μαθητή)

098 ★★☆☆
Τι γνωρίζετε για την Αντικειμενοστραφής σχεδίαση προγραμμάτων;

Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος **τα δεδομένα**, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση **τα αντικείμενα (objects)**. Τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα. Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την **ιεραρχική σχεδίαση**, τον **τμηματικό προγραμματισμό** και ακολουθεί τις αρχές του **δομημένου προγραμματισμού**.



(σελ 120 Βιβλίο Μαθητή)

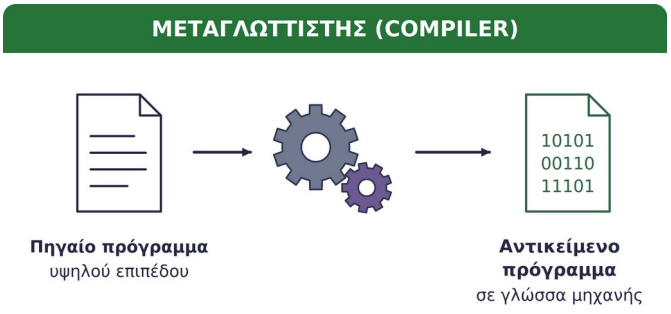
099 ★★★☆
Ποιος ο ρόλος και ποια τα χαρακτηριστικά του Μεταγλωττιστή και του Διερμηνευτή;

Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές **γλώσσας μηχανής**.



Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμάτων, οι **μεταγλωττιστές (compilers)** και οι **διερμηνευτές (interpreters)**.

Ο **μεταγλωττιστής** δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως **ανεξάρτητο από το αρχικό πρόγραμμα**. Αντίθετα ο **διερμηνευτής** διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για καθεμία εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.





(σελ 120 – 121 Βιβλίο Μαθητή)

100 ★★★☆
Ποια η διαφορά ανάμεσα στον Μεταγλωττιστή και τον Διερμηνευτή;

Η χρήση μεταγλωττιστή έχει το μειονέκτημα ότι, προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της **μεταγλώττισης και σύνδεσης**. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της **άμεσης εκτέλεσης** και συνεπώς και της **άμεσης διόρθωσης**. Όμως η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής. Πάντως τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με **μεικτές υλοποιήσεις**, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

(σελ 122 Βιβλίο Μαθητή)

101 ★★★★

Περιγράψτε τα στάδια μεταγλώττισης και σύνδεσης ενός προγράμματος από το αρχικό πρόγραμμα ως το εκτελέσιμο. Ποιος ο ρόλος του συνδέτη - φορτωτή;

Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής. Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμάτων, οι **μεταγλωττιστές (compilers)** και οι **διερμηνευτές (interpreters)**.



Το αρχικό πρόγραμμα λέγεται **πηγαίο πρόγραμμα (source)**, ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται **αντικείμενο πρόγραμμα (object)**.

Ο μεταγλωττιστής δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε **γλώσσα μηχανής**.



Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά συνήθως δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις **βιβλιοθήκες (libraries)** της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται **συνδέτης – φορτωτής (linker- loader)**. Το αποτέλεσμα του συνδέτη είναι η παραγωγή του **εκτελέσιμου προγράμματος (executable)**, το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται **μεταγλώττιση και σύνδεση**.



Αντίθετα ο **διερμηνευτής** διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για καθεμία εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

(σελ 120 - 121 Βιβλίο Μαθητή)

102 ★★☆☆

Τι ονομάζουμε συντάκτη



Για την αρχική σύνταξη των προγραμμάτων και τη διόρθωσή τους στη συνέχεια χρησιμοποιείται ένα ειδικό πρόγραμμα που ονομάζεται **συντάκτης (editor)**. Ο συντάκτης είναι ουσιαστικά ένας μικρός **επεξεργαστής κειμένου**, με δυνατότητες όμως που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων.

(σελ 122 Βιβλίο Μαθητή)

103 ★☆☆☆

Ποια προγράμματα και εργαλεία πρέπει να περιέχει ένα σύγχρονο προγραμματιστικό περιβάλλον;





Για τη δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον τρία προγράμματα: **ο συντάκτης**, **ο μεταγλωττιστής** και **ο συνδέτης**. Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.

(σελ 122 Βιβλίο Μαθητή)

104 ★★★☆

Ποια είναι τα χαρακτηριστικά του διερμηνευτή και του μεταγλωττιστή; (Διαφορές)








Η χρήση μεταγλωττιστή έχει το μειονέκτημα ότι, προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της **μεταγλώττισης και σύνδεσης**. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της **άμεσης εκτέλεσης** και συνεπώς και της **άμεσης διόρθωσης**. Όμως η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής. Πάντως τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με **μεικτές υλοποιήσεις**, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

(σελ 122 Βιβλίο Μαθητή)

105 ★★★★

Ποια είδη λαθών υπάρχουν σε ένα πρόγραμμα; Ποια λάθη εντοπίζουν ο μεταγλωττιστής και ο διερμηνευτής;

Μπορούμε να διακρίνουμε τις εξής κατηγορίες λαθών:



- **Συντακτικά λάθη**: Κάποιες φορές ένα πρόγραμμα δεν μπορεί να εκτελεστεί, επειδή **κατά τη μετάφραση** εντοπίζονται συντακτικά λάθη. Π.χ. δε γράψαμε σωστά μια δεσμευμένη λέξη, παραλείψαμε μια δεσμευμένη λέξη ή παραλείψαμε να δηλώσουμε μια μεταβλητή.



- Λάθη που οδηγούν σε **αντικανονικό τερματισμό** του προγράμματος: Ένα πρόγραμμα μπορεί να τερματίσει αντικανονικά λόγω διαφόρων λαθών. Για παράδειγμα, αν επιχειρήσουμε να **διαιρέσουμε με το μηδέν** ή αν κατά την ανάγνωση ενός ακεραίου αριθμού εισαχθεί ένα γράμμα.



- **Λογικά λάθη** που παράγουν **λανθασμένα αποτελέσματα**: Ακόμη κι αν το πρόγραμμά μας δεν περιέχει συντακτικά λάθη και μπορεί να εκτελεστεί, πρέπει οπωσδήποτε να ελεγχθεί, ώστε να διαπιστώσουμε αν κατά την εκτέλεσή του εμφανίζονται λογικά λάθη. Τα λογικά λάθη έχουν ως συνέπεια το πρόγραμμα σε κάποιες περιπτώσεις να εξάγει λανθασμένα αποτελέσματα. Για να εντοπίσουμε τα λογικά λάθη μπορούμε να κάνουμε δοκιμαστικές εκτελέσεις του προγράμματός μας και να ελέγξουμε αν για συγκεκριμένες τιμές εισόδου, το πρόγραμμά μας εξάγει σωστά αποτελέσματα.





Ο μεταγλωττιστής ή ο διερμηνευτής ανιχνεύει μόνο τα συντακτικά λάθη και εμφανίζει κατάλληλα **διαγνωστικά μηνύματα**. Το στάδιο που ακολουθεί είναι η διόρθωση των λαθών. Το διορθωμένο πρόγραμμα επαναϋποβάλλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαλειφθούν πλήρως όλα τα λάθη.

(σελ 115 – 117 – 119 Συμπληρωματικό Υλικό, σελ 121 – 122 Βιβλίο Μαθητή)

ΚΕΦΑΛΑΙΟ 10ο

Υποπρογράμματα

ΚΕΦΑΛΑΙΟ 10ο

106 ★☆☆☆
Τι ονομάζεται τμηματικός προγραμματισμός;

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.

(σελ 171 Βιβλίο Μαθητή)

107 ★★★☆
Ποια τεχνική σχεδίασης προγραμμάτων υλοποιεί ο τμηματικός προγραμματισμός;

Ο καλύτερος τρόπος για να αντιμετωπισθούν σύνθετα προβλήματα και να γραφούν τα αντίστοιχα προγράμματα, είναι η **ιεραρχική προσέγγιση**, η ανάπτυξη του προγράμματος **από επάνω προς τα κάτω (top-down)**. Κάθε πρόβλημα διαιρείται σε μικρότερα επιμέρους προβλήματα και κάθε ένα από αυτά τα προγράμματα διαιρείται σε ακόμα απλούστερα και μικρότερα. Στο τέλος τα επιμέρους υπο-προβλήματα είναι αρκετά απλά, ώστε οι αντίστοιχοι αλγόριθμοι και τα αντίστοιχα τμήματα προγράμματος να μπορούν να σχεδιασθούν και να γραφούν εύκολα. Ο τελικός αλγόριθμος του προβλήματος ανάγεται σε **πολλούς απλούστερους επιμέρους αλγορίθμους** και το τελικό πρόγραμμα σε πολλά απλούστερα τμήματα προγράμματος.



(σελ 171 Βιβλίο Μαθητή)

108 ★☆☆☆
Τι ονομάζουμε υποπρόγραμμα;

Υποπρόγραμμα (subprogram) είναι ένα τμήμα προγράμματος το οποίο επιτελεί ένα **αυτόνομο έργο** και έχει γραφτεί χωριστά από το υπόλοιπο πρόγραμμα.

(σελ 172 Βιβλίο Μαθητή)

109 ★★★☆
Ποιες είναι οι ιδιότητες των υποπρογραμμάτων



1. Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο.

Στην πραγματικότητα κάθε υποπρόγραμμα ενεργοποιείται με την είσοδο σε αυτό που γίνεται πάντοτε **από την αρχή του**, εκτελεί ορισμένες ενέργειες, και απενεργοποιείται με την έξοδο από αυτό που γίνεται πάντοτε **από το τέλος του**.

2. Κάθε υποπρόγραμμα πρέπει να είναι **ανεξάρτητο από τα άλλα**.

Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί **αυτόνομα** χωρίς να επηρεαστούν άλλα υποπρογράμματα. Στην πράξη βέβαια η απόλυτη ανεξαρτησία είναι δύσκολο να επιτευχθεί.

3. Κάθε υποπρόγραμμα πρέπει **να μην είναι πολύ μεγάλο**.

Η έννοια του μεγάλου προγράμματος είναι υποκειμενική, αλλά πρέπει κάθε υποπρόγραμμα να είναι τόσο, ώστε να είναι εύκολα κατανοητό για να μπορεί να ελέγχεται. Γενικά κάθε υποπρόγραμμα πρέπει να εκτελεί μόνο μία λειτουργία. Αν εκτελεί περισσότερες λειτουργίες, τότε συνήθως μπορεί και πρέπει να διασπαστεί σε ακόμη μικρότερα υποπρογράμματα.

(σελ 173 Βιβλίο Μαθητή)

110 ★★★★
Ποια είναι τα πλεονεκτήματα του Τμηματικού Προγραμματισμού;



1. Διευκολύνει την **ανάπτυξη του αλγορίθμου** και του αντιστοίχου προγράμματος.

Επιτρέπει την εξέταση και την επίλυση απλών προβλημάτων και όχι στην αντιμετώπιση του συνολικού προβλήματος. Με τη **σταδιακή επίλυση των υποπροβλημάτων** και τη δημιουργία των αντιστοίχων υποπρογραμμάτων τελικά επιλύεται το συνολικό πρόβλημα.



2. Διευκολύνει την **κατανόηση και διόρθωση** του προγράμματος.

Ο χωρισμός του προγράμματος σε μακρότερα αυτοτελή τμήματα επιτρέπει τη **γρήγορη διόρθωση** ενός συγκεκριμένου τμήματος του **χωρίς οι αλλαγές αυτές να επηρεάσουν** όλο το υπόλοιπο πρόγραμμα. Επίσης διευκολύνει οποιονδήποτε χρειαστεί να διαβάσει και να κατανοήσει τον τρόπο που λειτουργεί το πρόγραμμα. Όπως έχει πολλές φορές τονιστεί αυτό είναι πολύ σημαντικό χαρακτηριστικό του σωστού προγραμματισμού, αφού ένα μεγάλο πρόγραμμα χρειάζεται να συντηρηθεί από διαφορετικούς προγραμματιστές.



3. Απαιτεί **λιγότερο χρόνο και προσπάθεια** στη συγγραφή του προγράμματος.

Πολύ συχνά χρειάζεται η ίδια λειτουργία σε διαφορετικά σημεία ενός προγράμματος. Από τη στιγμή που ένα υποπρόγραμμα έχει γραφεί, μπορεί το ίδιο να καλείται από πολλά σημεία του προγράμματος. Έτσι μειώνονται το μέγεθος του προγράμματος, ο χρόνος που απαιτείται για τη συγγραφή του και οι πιθανότητες λάθους, ενώ ταυτόχρονα το πρόγραμμα γίνεται πιο εύληπτο και κατανοητό.



4. **Επεκτείνει τις δυνατότητες** των γλωσσών προγραμματισμού.

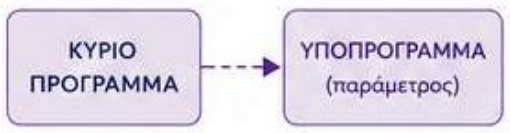
Ένα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί πολύ εύκολα **και σε άλλα προγράμματα**. Από τη στιγμή που έχει δημιουργηθεί, η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων που παρέχει η γλώσσα προγραμματισμού. Αν λοιπόν χρειάζεται συχνά κάποια λειτουργία που δεν υποστηρίζεται απευθείας από τη γλώσσα, τότε μπορεί να γραφεί το αντίστοιχο υποπρόγραμμα. Η συγγραφή πολλών υποπρογραμμάτων και η δημιουργία **βιβλιοθηκών** με αυτά, ουσιαστικά επεκτείνουν την ίδια τη γλώσσα προγραμματισμού.

(σελ 173-174 Βιβλίο Μαθητή)

111 ★★☆☆

Τι ονομάζουμε παράμετρο; Σε τι διαφέρουν οι παράμετροι από τις μεταβλητές;

Μία **παράμετρος** είναι μία **μεταβλητή** που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο. Οι παράμετροι είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μία ουσιώδη διαφορά, χρησιμοποιούνται για να περνούν τιμές στα υποπρογράμματα.



(σελ 174-175 Βιβλίο Μαθητή)

112 ★★☆☆

Ποια είδη υποπρογραμμάτων υπάρχουν στη ΓΛΩΣΣΑ και σε ποιο σημείο γράφονται αυτά;

Η ΓΛΩΣΣΑ υποστηρίζει δυο είδη υποπρογραμμάτων, τις **Συναρτήσεις** και τις **Διαδικασίες**. Τα υποπρογράμματα γράφονται μετά το Κυρίως Πρόγραμμα και συγκεκριμένα αμέσως μετά της εντολή «**ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ**». Η σειρά των υποπρογραμμάτων (όταν έχουμε περισσότερα από ένα) δεν παίζει ρόλο.

(σελ 175 Βιβλίο Μαθητή)

113 ★★☆☆

Τι ονομάζουμε συνάρτηση και τι διαδικασία;

Η **συνάρτηση** είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομά της (όπως οι μαθηματικές συναρτήσεις).

Η **διαδικασία** είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος.

(σελ 175 Βιβλίο Μαθητή)

114 ★★★☆

Τι ονομάζουμε τυπικές και τι πραγματικές παράμετροι; Πως αλλιώς ονομάζονται οι τυπικές και οι πραγματικές παράμετροι στη ΓΛΩΣΣΑ; Τι καθορίζει η λίστα των τυπικών και τι η λίστα των πραγματικών παραμέτρων;

- **Τυπικές παράμετροι** είναι εκείνες που χρησιμοποιούνται στη δήλωση του υποπρογράμματος. **Πραγματικές** είναι εκείνες που χρησιμοποιούνται κατά την κλήση του υποπρογράμματος.



- Οι τυπικές **παράμετροι** λέγονται αλλιώς και **ορίσματα** ενώ οι πραγματικές παράμετροι λέγονται πιο απλά και παράμετροι.
- Η λίστα των **τυπικών παραμέτρων (formal parameter list)** καθορίζει τις παραμέτρους στη δήλωση του υποπρογράμματος. Η λίστα των **πραγματικών παραμέτρων (actual parameter list)** καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος.

(σελ 181 Βιβλίο Μαθητή)

115 ★★☆☆

Πόσες παραμέτρους μπορεί να έχει μια συνάρτηση και πόσες μια διαδικασία;

Μια συνάρτηση μπορεί να έχει **τουλάχιστον μια παράμετρο**. Είναι όλες παράμετροι εισόδου.



Οι διαδικασίες μπορεί να έχουν **καμία, μία ή περισσότερες παραμέτρους**. Όταν υπάρχουν παράμετροι άλλες χρησιμοποιούνται για να μεταβιβάσουν τιμές στη διαδικασία, άλλες για να επιστρέψουν τιμές από αυτή και άλλες για να λειτουργήσουν και **ως είσοδος και ως έξοδος** τιμών.



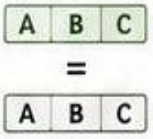
(σελ 181 Βιβλίο Μαθητή)

116 ★★★☆

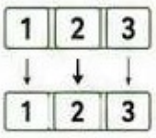
Ποιοι είναι οι κανόνες της λίστας παραμέτρων στα υποπρογράμματα;

Οι λίστες των παραμέτρων πρέπει να ακολουθούν τους εξής κανόνες:

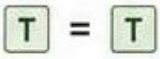
1. Ο **αριθμός** των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.



2. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην **αντίστοιχη θέση**. Για παράδειγμα, η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κ.ο.κ.



3. Η τυπική παράμετρος και η αντίστοιχή της πραγματική πρέπει να είναι του ιδίου τύπου.



(σελ 182 Βιβλίο Μαθητή)

117 ★★★☆

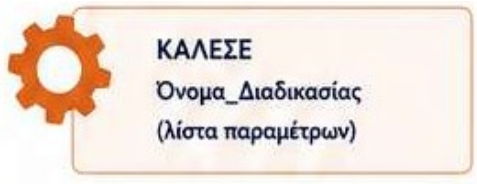
Με ποιόν τρόπο καλείται μια Συνάρτηση;



Κάθε συνάρτηση εκτελείται, όπως ακριβώς εκτελούνται οι ενσωματωμένες συναρτήσεις της γλώσσας. Απλώς αναφέρεται **το όνομά της σε μια έκφραση** ή σε μία εντολή και επιστρέφεται η τιμή της. Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται στη συνάρτηση κατά την κλήση. Οι συναρτήσεις μπορούν να επιστρέφουν τιμές όλων των τύπων δεδομένων που υποστηρίζει η γλώσσα. Μια συνάρτηση λοιπόν μπορεί να είναι **ΠΡΑΓΜΑΤΙΚΗ, ΑΚΕΡΑΙΑ, ΧΑΡΑΚΤΗΡΑΣ, ΛΟΓΙΚΗ**. Στις εντολές του σώματος της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μία εντολή εκχώρησης τιμής στο όνομα της συνάρτησης

118 ★★★☆
Με ποιόν τρόπο καλείται μια Διαδικασία;

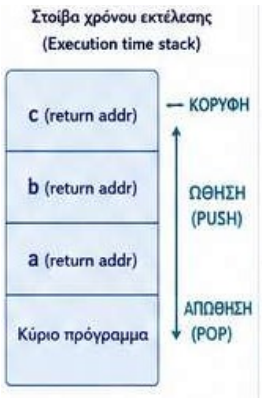
Κάθε διαδικασία εκτελείται όταν καλείται από το κύριο πρόγραμμα ή άλλη διαδικασία. Η κλήση σε διαδικασία πραγματοποιείται με την εντολή **ΚΑΛΕΣΕ**, που ακολουθείται από το όνομα της διαδικασίας συνοδευόμενο μέσα σε παρενθέσεις με τη **λίστα παραμέτρων**. Το όνομα της διαδικασίας είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται προς τη διαδικασία κατά την κλήση ή/και επιστρέφονται στο κύριο πρόγραμμα μετά το τέλος της διαδικασίας. Στο σώμα της διαδικασίας μπορούν να υπάρχουν **οποιεσδήποτε εντολές** της γλώσσας.

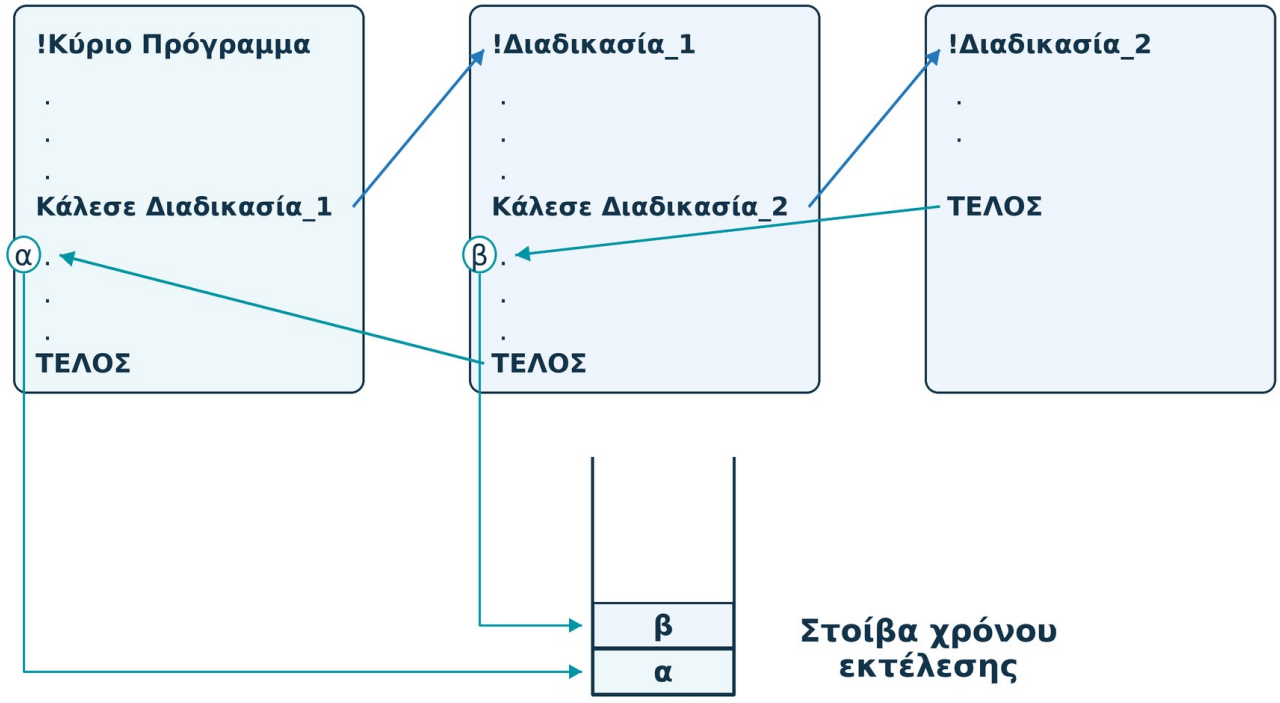


(σελ 178 Βιβλίο Μαθητή)

119 ★★★★
Τι γνωρίζετε για τη Στοίβα χρόνου εκτέλεσης;

Όταν μία διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η αμέσως επόμενη διεύθυνση του κύριου προγράμματος, που ονομάζεται **διεύθυνση επιστροφής (return address)**, αποθηκεύεται από το μεταφραστή σε μία στοίβα που ονομάζεται **στοίβα χρόνου εκτέλεσης (execution time stack)**. Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής **απωθείται από τη στοίβα** και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κύριο πρόγραμμα. Η τεχνική αυτή εφαρμόζεται και γενικότερα, δηλαδή οποτεδήποτε μία διαδικασία ή συνάρτηση καλεί μία διαδικασία ή συνάρτηση. Για παράδειγμα, έστω ότι μία διαδικασία a καλεί τη διαδικασία b, που με τη σειρά της καλεί τη διαδικασία c κλπ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στη στοίβα με σειρά c, b, a. Μετά την εκτέλεση κάθε διαδικασίας, η διεύθυνση επιστροφής απωθείται από τη στοίβα και ο έλεγχος μεταβιβάζεται στη διεύθυνση αυτή. Το παράδειγμα αυτό δείχνει μία από τις πολλές χρησιμότητες της LIFO ιδιότητας της στοίβας.





(σελ 182 Βιβλίο Μαθητή)

120 ★★★☆
Τι ονομάζουμε εμβέλεια μεταβλητών; τι εννοούμε όταν λέμε ότι οι μεταβλητές έχουν εμβέλεια τοπικά;

Το τμήμα του προγράμματος που ισχύουν οι μεταβλητές λέγεται **εμβέλεια (scope) μεταβλητών**.

Οι μεταβλητές στη ΓΛΩΣΣΑ είναι γνωστές στο αντίστοιχο πρόγραμμα ή υποπρόγραμμα που δηλώνονται και μόνο σε αυτό. Όλες οι μεταβλητές (και οι σταθερές) είναι **τοπικές** στο συγκεκριμένο τμήμα προγράμματος. Ο μόνος τρόπος για να περάσει μία τιμή από ένα υποπρόγραμμα σε ένα άλλο ή από το κυρίως πρόγραμμα σε ένα υποπρόγραμμα είναι **διαμέσου των παραμέτρων** κατά το στάδιο της κλήσης του υποπρογράμματος και μετά το τέλος της εκτέλεσης του υποπρογράμματος.

(σελ 183 Βιβλίο Μαθητή)

121 ★★★☆
Τι ονομάζουμε απεριόριστη εμβέλεια μεταβλητών; Τι προβλήματα αυτή δημιουργεί;

Σύμφωνα με αυτή την αρχή όλες οι μεταβλητές και όλες οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, άσχετα που δηλώθηκαν. Όλες οι μεταβλητές είναι **καθολικές** και μπορούν να χρησιμοποιηθούν σε οποιοδήποτε τμήμα προγράμματος ή υποπρογράμματος, ανεξάρτητα με το που δηλώθηκαν.

Η απεριόριστη εμβέλεια καταστρατηγεί την **αρχή της αυτονομίας των υποπρογραμμάτων**, δημιουργεί πολλά προβλήματα και τελικά είναι **αδύνατη για μεγάλα προγράμματα** με πολλά υποπρογράμματα, αφού ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.

(σελ 184 Βιβλίο Μαθητή)

122 ★★☆☆

Τι ονομάζουμε περιορισμένη εμβέλεια μεταβλητών; Ποια τα πλεονεκτήματά της;

Η **περιορισμένη εμβέλεια** υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα. Όλες οι μεταβλητές είναι **τοπικές**, ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Στη ΓΛΩΣΣΑ έχουμε περιορισμένη εμβέλεια.

Τα πλεονεκτήματα της περιορισμένης εμβέλειας είναι η **απόλυτη αυτονομία** όλων των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.

(σελ 184 Βιβλίο Μαθητή)

123 ★★☆☆

Τι ονομάζουμε μερικώς περιορισμένη εμβέλεια; Τι προσφέρει αυτού του είδους η εμβέλεια;



Σύμφωνα με αυτή την αρχή άλλες μεταβλητές είναι **τοπικές** και άλλες **καθολικές**. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές.

Η μερικώς περιορισμένη εμβέλεια προσφέρει μερικά πλεονεκτήματα στον **πεπειραμένο προγραμματιστή**, αλλά **για τον αρχάριο περιπλέκει** το πρόγραμμα δυσκολεύοντας την ανάπτυξή του.

(σελ 184 Βιβλίο Μαθητή)

ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ

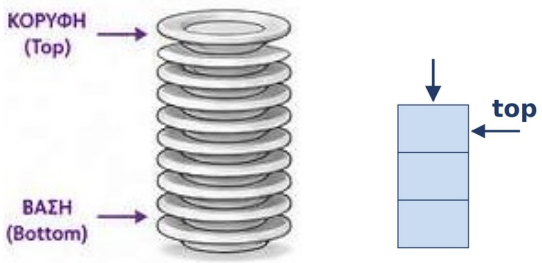
Στοίβα και ουρά

ΕΝΟΤΗΤΑ 1η Συμπληρωματικό Υλικό

124 ★★★★

Τι γνωρίζετε για τη δομή δεδομένων Στοίβα; Ποιες είναι οι κύριες λειτουργίες της; Τι πρέπει να προσέχουμε;

Στοίβα (stack), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που βρίσκονται στην **κορυφή της στοίβας** λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο **βάθος της στοίβας** λαμβάνονται τελευταία. Η παραπάνω μέθοδος ονομάζεται **Τελευταίο Μέσα, Πρώτο Έξω ή LIFO** (=Last In First Out). Μπορούμε να φανταστούμε την τοποθέτηση των στοιχείων μιας στοίβας σε κατακόρυφη σειρά. Χαρακτηριστικό παράδειγμα είναι μια στοίβα από πιάτα. Παίρνουμε προς χρήση το πιάτο που τοποθετήσαμε τελευταίο.



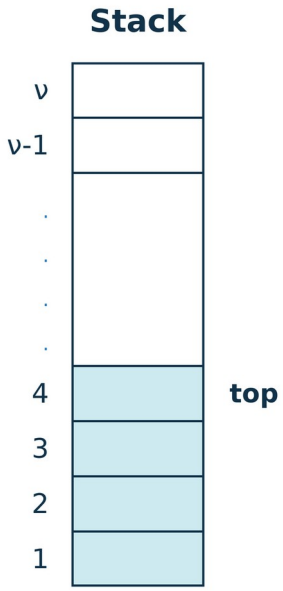
Οι κύριες λειτουργίες σε μια στοίβα είναι δύο:

1. Η **ώθηση (push)** στοιχείου στην κορυφή της στοίβας. Στη διαδικασία της ώθησης ελέγχουμε αν η στοίβα είναι γεμάτη. Στην περίπτωση που προσπαθήσουμε να «προσθέσουμε» ένα στοιχείο σε μια ήδη γεμάτη στοίβα, έχουμε **υπερχείλιση (overflow)** της στοίβας.



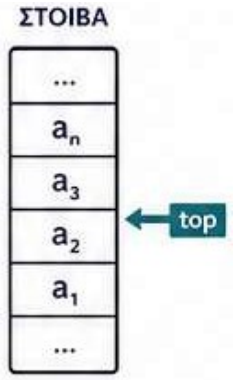
2. Η **απώθηση (pop)** στοιχείου από τη στοίβα. Στη διαδικασία της απώθησης ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα. Στην περίπτωση που προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία κενή στοίβα, έχουμε **υποχείλιση (underflow)** της στοίβας.

(σελ 13 Συμπληρωματικό Υλικό)



125 ★★★★
Πως υλοποιείται η δομή δεδομένων Στοίβα στη ΓΛΩΣΣΑ;

Χρησιμοποιούμε μια βοηθητική **μεταβλητή (top)**, που δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην **κορυφή της στοίβας**.



- Η **ώθηση** ενός νέου στοιχείου στη στοίβα (εισαγωγή στοιχείου στον πίνακα) γίνεται πάντα στην κορυφή της. Συγκεκριμένα, η μεταβλητή top αυξάνεται κατά ένα: top ← top+1 και στη συνέχεια γίνεται η ώθηση του στοιχείου.
- Η **απώθηση** ενός στοιχείου από τη στοίβα (εξαγωγή από τον πίνακα) γίνεται πάντα από την κορυφή της στοίβας. Συγκεκριμένα, εξάγεται το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η μεταβλητή top μειώνεται κατά ένα: top ← top-1

Σε μια **κενή στοίβα/πίνακα** θεωρούμε ότι η αρχική τιμή της μεταβλητής top είναι **μηδέν** (top ← 0). Η μεταβλητή top είναι η μεταβλητή που δείχνει τη θέση που τοποθετήθηκε το τελευταίο στοιχείο στη στοίβα/πίνακα (δηλ. δείχνει την κορυφή της στοίβας).

Κατά την ώθηση ενός στοιχείου στη στοίβα (εισαγωγή ενός στοιχείου στον πίνακα), **πρώτα αυξάνεται** η τιμή της μεταβλητής top κατά ένα, δηλ. top ← top+1, και στη συνέχεια γίνεται η

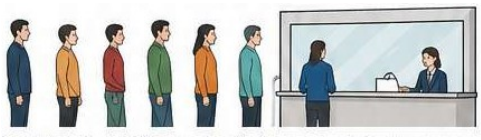
ώθηση του στοιχείου στην κορυφή της στοίβας. Κατά την απώθηση ενός στοιχείου από τη στοίβα (εξαγωγή στοιχείου από τον πίνακα) μειώνεται η τιμή της μεταβλητής top κατά ένα, δηλ. top ← top-1. Στην απώθηση **δε διαγράφεται το στοιχείο**, στην πραγματικότητα δε γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα. Απλώς ο δείκτης κορυφή δείχνει στην **προηγούμενη θέση**.

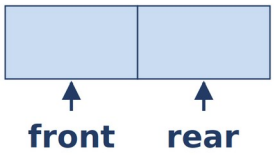
(σελ 14 Συμπληρωματικό Υλικό)

126 ★★★☆
Τι γνωρίζετε για τη δομή δεδομένων Ουρά; Ποιες είναι οι κύριες λειτουργίες της; Τι πρέπει να προσέχουμε;

Ουρά (Queue), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα.

Η παραπάνω μέθοδος ονομάζεται **Πρώτο Μέσα, Πρώτο Έξω ή FIFO** (=First In First Out). Μπορούμε να φανταστούμε την τοποθέτηση των στοιχείων μιας ουράς σε οριζόντια σειρά. Ο πελάτης μιας τράπεζας που μπαίνει πρώτος σε μια ουρά για να εξυπηρετηθεί, είναι αυτός που εξυπηρετείται και πρώτος, με την έξοδό του από την ουρά αναμονής.





Άλλα παραδείγματα είναι η ουρά στα λεωφορεία ή η ουρά στα ταμεία, όπου ο πρώτος που στέκεται στην ουρά εξυπηρετείται και πρώτος.

Οι κύριες λειτουργίες που εκτελούνται σε μια ουρά είναι δύο:

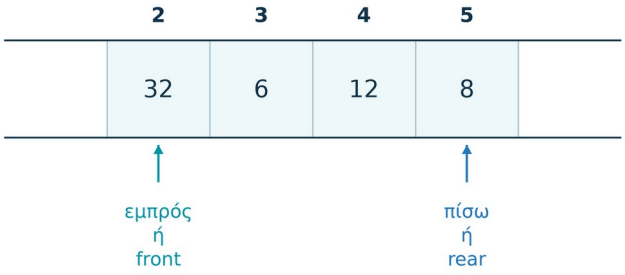
1. Η **εισαγωγή (enqueue)** στοιχείου στο πίσω άκρο της ουράς.
2. Η **εξαγωγή (dequeue)** στοιχείου από το εμπρός άκρο της ουράς.

Κατά την εισαγωγή στοιχείου στην ουρά πρέπει να εξετάζουμε αν υπάρχει διαθέσιμος χώρος και αυτή **να μην είναι γεμάτη** διότι θα συμβεί **υπερχείλιση ουράς**. Επίσης κατά την εξαγωγή στοιχείου από την ουρά θα πρέπει να εξετάζουμε αν υπάρχει τουλάχιστον ένα στοιχείο στην ουρά για να μπορεί να εξαχθεί και **να μην είναι κενή** καθώς σε διαφορετική περίπτωση θα συμβεί **υποχείλιση της ουράς**.

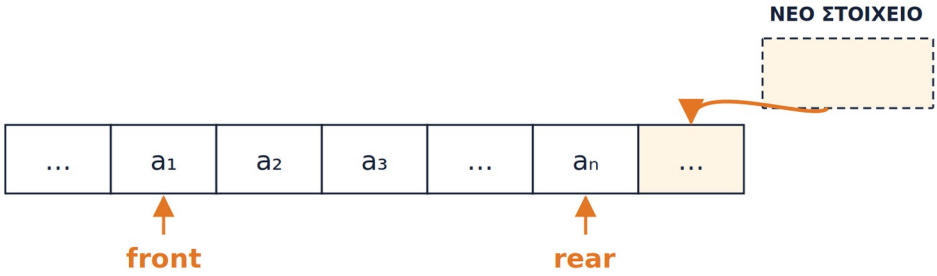
(σελ 23 Συμπληρωματικό Υλικό)

127 ★★★☆
Πως υλοποιείται η δομή δεδομένων Ουρά στη ΓΛΩΣΣΑ;

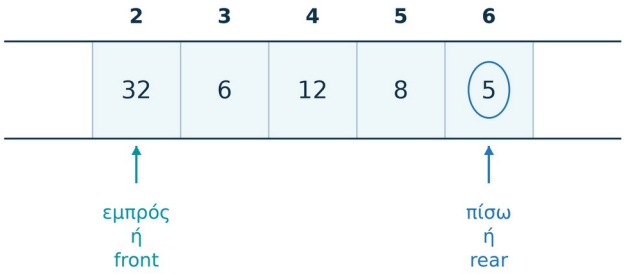
Χρησιμοποιούμε δύο μεταβλητές, την **front (ή εμπρός)** που δείχνει τη θέση του 1ου στοιχείου της ουράς και την **rear (ή πίσω)** που δείχνει τη θέση του τελευταίου στοιχείου. Ως **αρχικές τιμές** των μεταβλητών rear και front θεωρούμε το μηδέν.



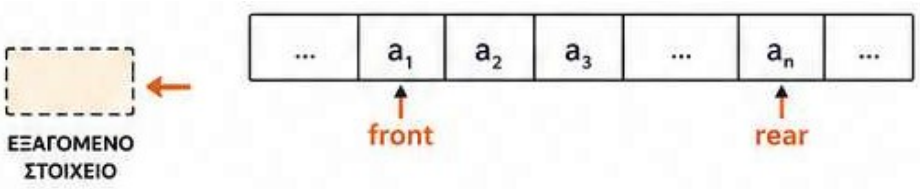
- Η εισαγωγή ενός νέου στοιχείου γίνεται από το **πίσω άκρο** της ουράς και η τιμή της μεταβλητής rear αλλάζει ως εξής: rear ← rear+1. Κατά την εισαγωγή, **πρώτα αυξάνουμε τον δείκτη rear** κατά ένα και μετά εισάγουμε το στοιχείο στον πίνακα.

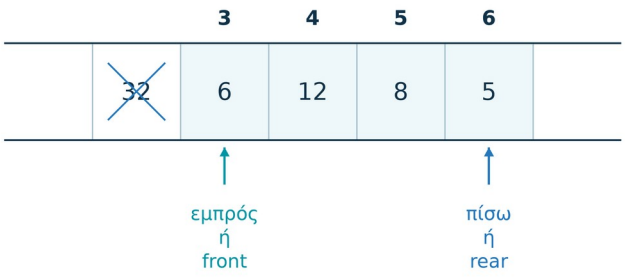


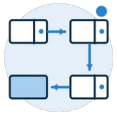
(σελ 24 Συμπληρωματικό Υλικό)



- Η εξαγωγή ενός στοιχείου γίνεται από το **εμπρός άκρο** της ουράς και η τιμή της μεταβλητής front αλλάζει ως εξής: front ← front +1. Κατά την εξαγωγή ενός στοιχείου, **αυξάνεται ο δείκτης front** κατά ένα (δείχνει στην επόμενη θέση του πίνακα) χωρίς στην πραγματικότητα να γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα (χωρίς να διαγράφεται κάποιο στοιχείο).





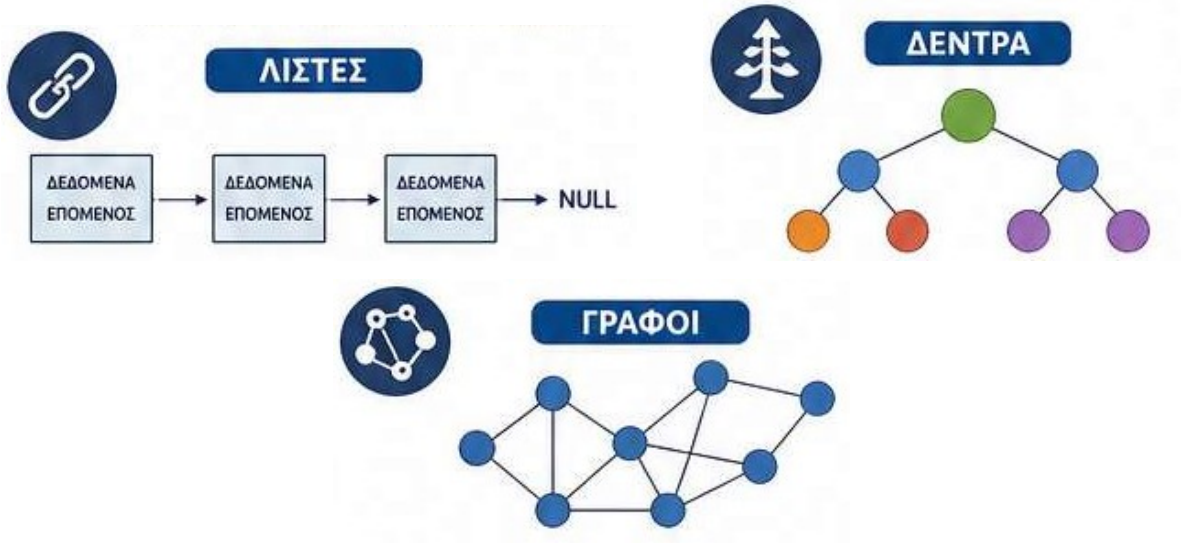


ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ

Συνδεδεμένες λίστες

128 ★☆☆☆
Ποιες δυναμικές δομές δεδομένων γνωρίζετε και ποιο είναι το χαρακτηριστικό τους γνώρισμα.

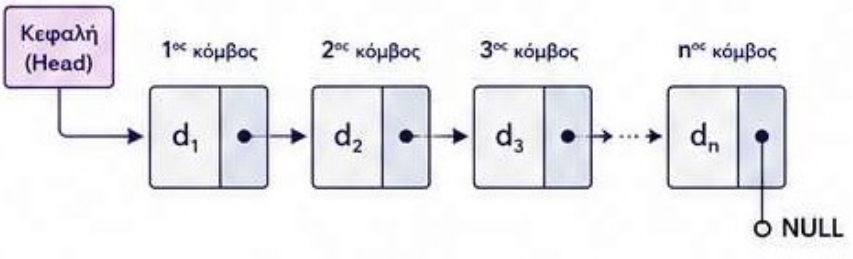
Δομές δεδομένων, στις οποίες οι κόμβοι δεν είναι απαραίτητο να κατέχουν συνεχόμενες θέσεις μνήμης είναι οι **λίστες**, τα **δέντρα** και οι **γράφοι**.



(σελ 37 Συμπληρωματικό Υλικό)

129 ★★★☆
Τι ονομάζουμε Συνδεδεμένη Λίστα

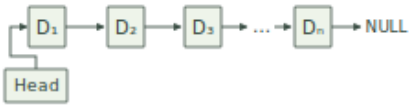
Μία (απλά) **συνδεδεμένη λίστα (linked list)** είναι ένα σύνολο κόμβων **διατεταγμένων γραμμικά** (ο ένας μετά τον άλλο). Κάθε κόμβος περιέχει εκτός από τα **δεδομένα** του και έναν **δείκτη** που δείχνει προς τον επόμενο κόμβο.

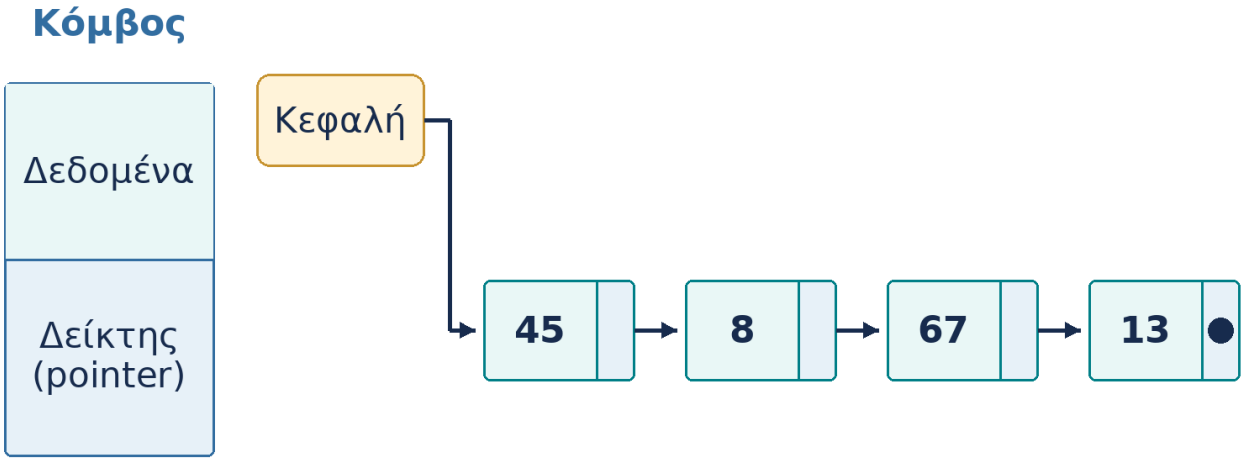


Ο δείκτης του τελευταίου κόμβου δε δείχνει σε κάποιον κόμβο (δείκτης στο κενό). Για να το

δηλώσουμε αυτό λέμε ότι το πεδίο δείκτη του **τελευταίου κόμβου** έχει την τιμή **NULL**.

Για να προσπελάσουμε τους κόμβους της λίστας χρειάζεται να γνωρίζουμε τη διεύθυνση (θέση στη μνήμη) του πρώτου κόμβου της λίστας. Η διεύθυνση αυτή αποθηκεύεται σε μία ειδική μεταβλητή που την ονομάζουμε συνήθως **Κεφαλή (Head)**.

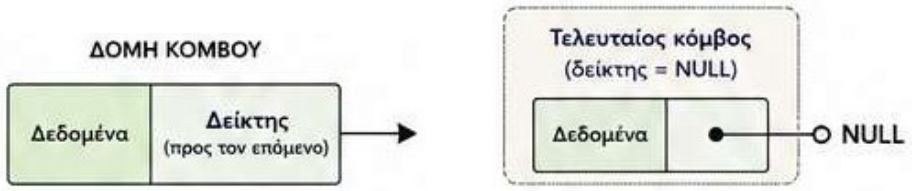




(σελ 37 Συμπληρωματικό Υλικό)

130 ★★★★
Τι πεδία μπορεί κανείς να διακρίνει σε έναν κόμβο μίας απλά συνδεδεμένης λίστας; Τι ονομάζεται δείκτης σε μια δομή δεδομένων;

Κάθε κόμβος αποτελείται από δύο κύρια τμήματα. Το πρώτο τμήμα περιέχει τα **δεδομένα** και το δεύτερο τμήμα φιλοξενεί τη **διεύθυνση του επόμενου κόμβου** με τον οποίο συνδέεται ή όπως αλλιώς θα λέγαμε στη γλώσσα των δομών δεδομένων, το δεύτερο τμήμα περιέχει έναν **δείκτη (pointer)** που δείχνει στον επόμενο κόμβο. Το πεδίο Δεδομένα μπορεί να περιέχει μία ή περισσότερες αλφαριθμητικές ή αριθμητικές πληροφορίες. Ο δείκτης του τελευταίου κόμβου της λίστας έχει ως τιμή το **NULL (κενό)** και τον αναπαριστούμε συμβολικά με το σύμβολο του συμπαγούς κύκλου (σχήμα)





Οι δείκτες των υπολοίπων κόμβων περιέχουν τη διεύθυνση του επόμενου κόμβου αντίστοιχα και τους απεικονίζουμε συμβολικά με ένα βέλος για να υποδηλώσουμε τη

σύνδεση μεταξύ του προηγούμενου και του επόμενου κόμβου. Κάθε λίστα συνοδεύεται από έναν δείκτη με το όνομα **«Κεφαλή» (head)**, που δείχνει στον πρώτο κόμβο της λίστας, δηλαδή περιέχει τη διεύθυνση του πρώτου κόμβου της λίστας.

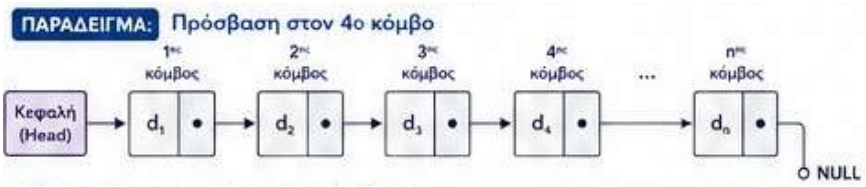
Ο **δείκτης (pointer)** είναι ένας ιδιαίτερος τύπος δεδομένων που προσφέρεται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού. Ο δείκτης δε λαμβάνει αριθμητικές τιμές όπως ακέραιες, πραγματικές κ.ά., αλλά οι τιμές του είναι διευθύνσεις στην κύρια μνήμη και χρησιμοποιείται ακριβώς για τη σύνδεση των διαφόρων στοιχείων μιας δομής, που είναι αποθηκευμένα σε μη συνεχόμενες θέσεις μνήμης.

(σελ 38-39 Συμπληρωματικό Υλικό)

131 ★★★☆
Πως μπορούμε να προσπελάσουμε τους κόμβους μιας συνδεδεμένης λίστας;

Οι κόμβοι μιας (απλά) συνδεδεμένης λίστας είναι διατεταγμένοι σε μια συγκεκριμένη σειρά, χωρίς αυτό να σημαίνει ότι αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη. Αντίθετα, είναι διασκορπισμένοι σε όλη τη μνήμη και η σύνδεση μεταξύ τους γίνεται **μέσω των δεικτών**. Έχουμε άμεση πρόσβαση μόνο στον πρώτο κόμβο της λίστας. Επομένως, για να εντοπίσουμε κάποιον από τους ενδιάμεσους κόμβους, πρέπει να ξεκινήσουμε από τον πρώτο κόμβο της λίστας και **να ακολουθήσουμε τους δείκτες με τη σειρά**, μέχρι να φτάσουμε στον επιθυμητό κόμβο.

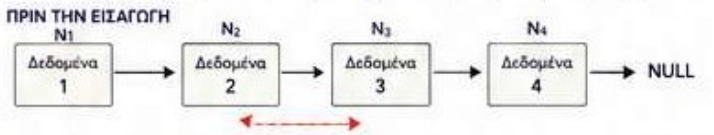
Για παράδειγμα αν θελήσουμε να έχουμε πρόσβαση στον τέταρτο κόμβο μιας λίστας, για να επεξεργαστούμε τα δεδομένα που περιέχει, θα πρέπει να ξεκινήσουμε από τον πρώτο κόμβο της λίστας, η διεύθυνση του οποίου περιέχεται στον δείκτη Κεφαλή. Ξεκινώντας από τον πρώτο κόμβο της λίστας μπορούμε να έχουμε πρόσβαση στη διεύθυνση μνήμης του δεύτερου κόμβου. Από τον δεύτερο κόμβο μπορούμε να έχουμε πρόσβαση στη διεύθυνση μνήμης του τρίτου κόμβου. Και συνεχίζουμε έτσι μέχρι να φτάσουμε στον τέταρτο κόμβο. Αυτός είναι ο μόνος τρόπος για να διασχίσουμε μία συνδεδεμένη λίστα. Για το λόγο αυτό και οι Λίστες χαρακτηρίζονται επίσης και **Γραμμικές Δυναμικές Δομές Δεδομένων**.



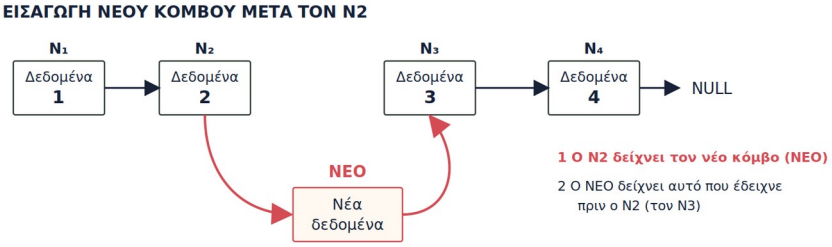
(σελ 39 Συμπληρωματικό Υλικό)

132 ★★☆☆
Πώς προσθέτω νέο κόμβο σε μια συνδεδεμένη λίστα;

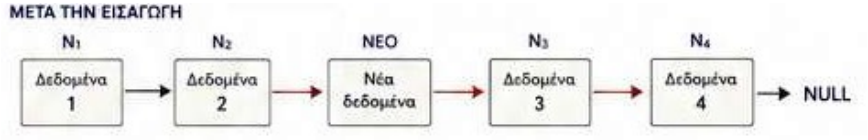
Οι απαιτούμενες ενέργειες για την εισαγωγή (παρεμβολή) του νέου κόμβου είναι:

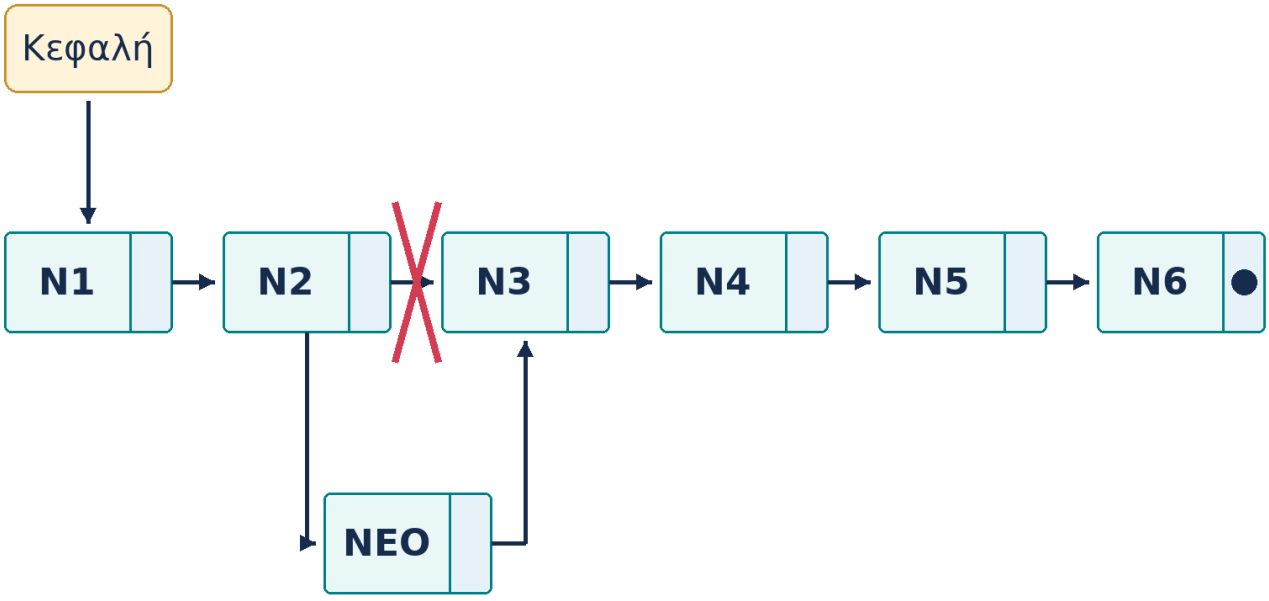


1. ο **δείκτης του κόμβου** μετά τον οποίο θα γίνει η παρεμβολή (N2), <u>να δείχνει τον νέο κόμβο</u> (NEO)



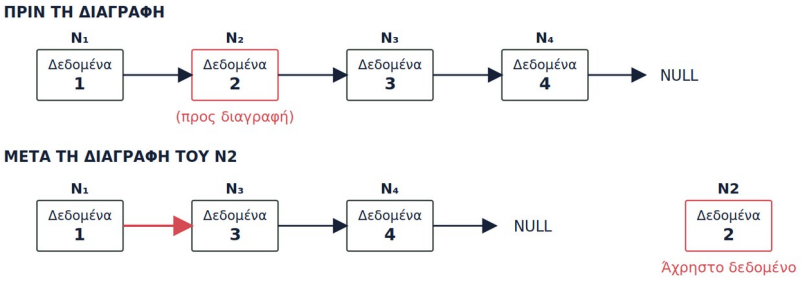
2. ο **δείκτης του νέου κόμβου** (NEO), <u>να δείχνει αυτό που έδειχνε ο προηγούμενος</u> (στο N3) (δηλαδή, να πάρει την τιμή που είχε πριν την εισαγωγή ο δείκτης του N2). Με αυτόν τον τρόπο, οι κόμβοι της λίστας διατηρούν τη **λογική τους σειρά**, αλλά οι φυσικές θέσεις στη μνήμη μπορεί να είναι τελείως διαφορετικές.

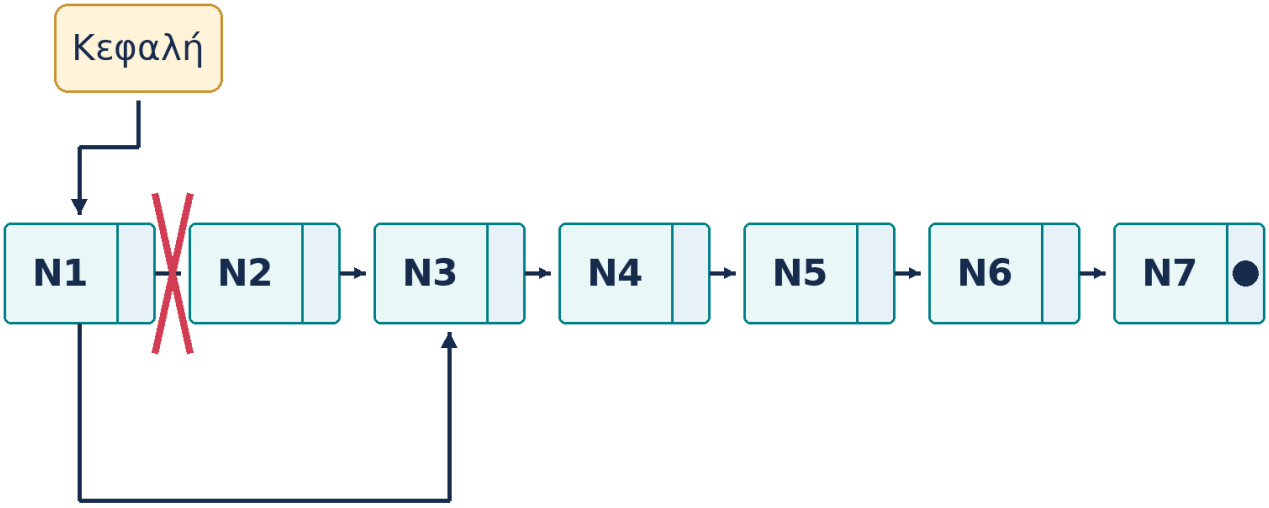




133 ★☆☆☆
Πώς διαγράφω κόμβο από μια συνδεδεμένη λίστα;

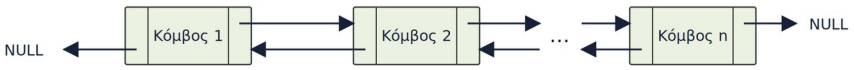
Για τη διαγραφή ενός κόμβου αρκεί να αλλάξει τιμή ο **δείκτης του προηγούμενου κόμβου** και να δείχνει πλέον στον επόμενο αυτού που διαγράφεται. Ο κόμβος που διαγράφηκε (ο δεύτερος) αποτελεί «άχρηστο δεδομένο» και ο **χώρος μνήμης** που καταλάμβανε, παραχωρείται για άλλη χρήση.

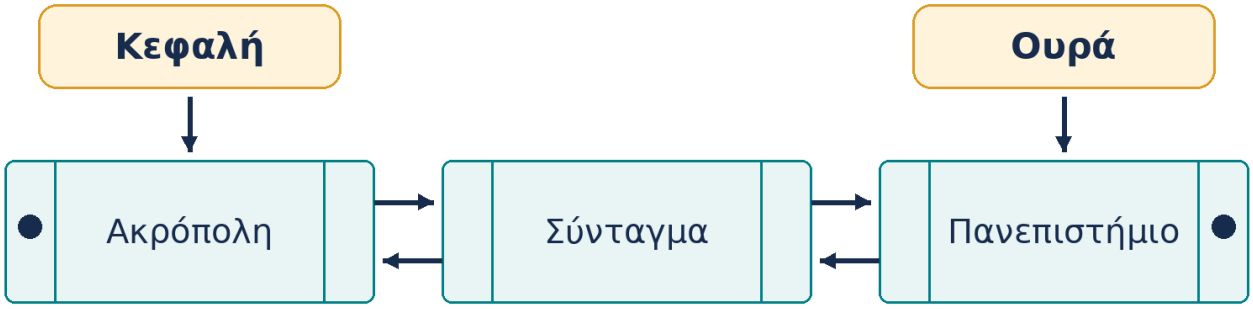




134 ★★☆☆
Ποιες λίστες ονομάζουμε Διπλά συνδεδεμένες; Τι προβλήματα μπορούν να δημιουργηθούν;

Διπλά συνδεδεμένες είναι οι λίστες στις οποίες μπορούμε να κινηθούμε και προς τις δυο κατευθύνσεις. Χρειαζόμαστε **δεύτερο δείκτη** για την υλοποίησή τους (σχήμα).







Οι διπλά συνδεδεμένες λίστες αυξάνουν την **πολυπλοκότητα** στη διαχείριση των κόμβων καθώς απαιτείται **επιπλέον χώρος στη μνήμη** για τον επιπλέον δείκτη.

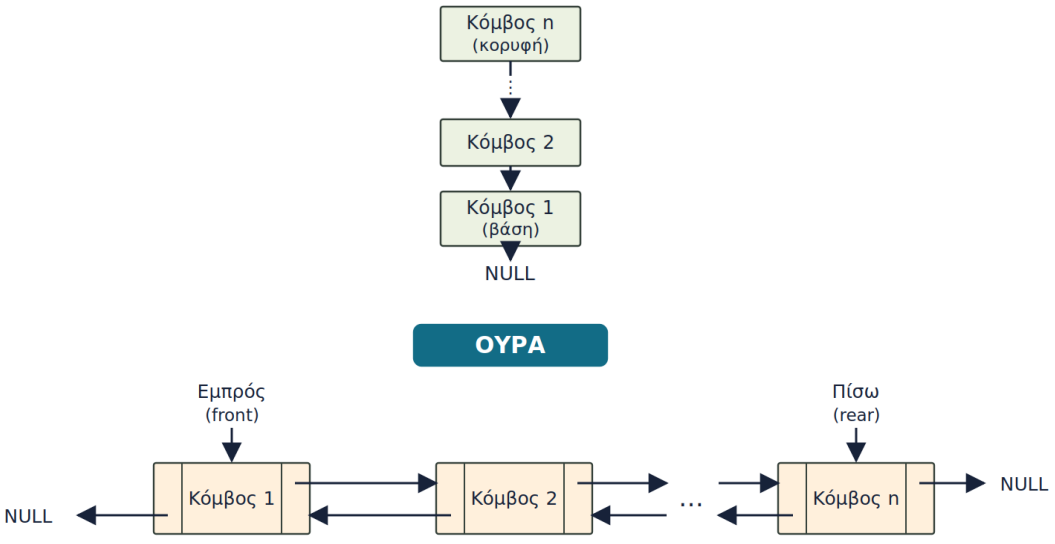
(σελ 41 Συμπληρωματικό Υλικό)



135 ★☆☆☆
Που αξιοποιούνται οι Συνδεδεμένες Λίστες;



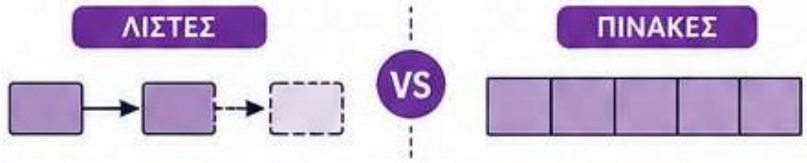

Οι συνδεδεμένες λίστες αξιοποιούνται για την υλοποίηση της **στοίβας και της ουράς**, λόγω της δυνατότητάς αυξομείωσης του μεγέθους τους. Συγκεκριμένα οι **Απλά συνδεδεμένες λίστες** χρησιμοποιούνται στη διαχείριση μιας Στοίβας ενώ οι **Διπλά συνδεδεμένες Λίστες** χρησιμοποιούνται για τη διαχείριση μιας Ουράς.



(σελ 41 Συμπληρωματικό Υλικό)

136 ★★★☆
Διαφορές μεταξύ λιστών και πινάκων

Οι βασικές διαφορές μεταξύ των λιστών και των πινάκων είναι:



1. 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' έναν κόμβο μιας λίστας πρέπει να περάσουμε από όλους τους προηγούμενους ξεκινώντας από τον πρώτο.
2. Ο πίνακας έχει **σταθερό μέγεθος**, το οποίο δηλώνεται εξαρχής κατά την υλοποίηση. Αυτό γίνεται, διότι ο πίνακας είναι **στατική δομή** δεδομένων σε αντίθεση με τη λίστα που είναι **δυναμική δομή** και το μέγεθός της μπορεί να μεταβάλλεται καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.



3. Οι κόμβοι της λίστας αποθηκεύονται σε **μη συνεχόμενες θέσεις μνήμης** σε αντιδιαστολή με τους πίνακες, όπου τα στοιχεία αποθηκεύονται **σε συνεχόμενες θέσεις μνήμης**.

(σελ 42 Συμπληρωματικό Υλικό)

137 ★★★★
Αναφέρετε τα πλεονεκτήματα και τα μειονεκτήματα από τη χρήση της δομής Λίστας έναντι της δομής Πίνακα.

Στα πλεονεκτήματα των λιστών (έναντι των πινάκων) συγκαταλέγονται τα εξής:

1. Το **δυναμικό τους μέγεθος**.
2. Η **ευκολία εισαγωγής και διαγραφής** από οποιοδήποτε μέρος της λίστας.
3. Η **μη αναγκαιότητα δήλωσης του μεγέθους** τους.

Στα μειονεκτήματα των λιστών (έναντι των πινάκων) περιλαμβάνονται τα εξής:

1. Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται. Είναι αδύνατο να φτάσετε στον n-οστό κόμβο μιας απλά συνδεδεμένης λίστας χωρίς πρώτα να περάσετε από όλους τους κόμβους διαδοχικά μέχρι τον συγκεκριμένο κόμβο ξεκινώντας από τον πρώτο κόμβο. Εναλλακτικά, στην περίπτωση της διπλά συνδεμένης λίστας μπορείτε να ξεκινήσετε και από τον τελευταίο κόμβο. Επομένως, δεν μπορούμε να πραγματοποιήσουμε με αποτελεσματικό τρόπο **δυαδική αναζήτηση** σε συνδεδεμένες λίστες.

2. Οι συνδεδεμένες λίστες έχουν πολύ **μεγαλύτερη επιβάρυνση** από τους πίνακες, αφού οι συνδεδεμένοι κόμβοι της λίστας είναι δυναμικά κατανεμημένοι (οι οποίοι είναι λιγότερο αποτελεσματικοί στη χρήση της μνήμης) και κάθε κόμβος στη λίστα πρέπει, επιπλέον, να αποθηκεύσει έναν **πρόσθετο δείκτη** που θα δείχνει στον επόμενο κόμβο. Στην περίπτωση των διπλά συνδεδεμένων λιστών χρειαζόμαστε επιπλέον έναν **δεύτερο δείκτη** που θα δείχνει στον προηγούμενο κόμβο.

(σελ 42 Συμπληρωματικό Υλικό)

138 ★★☆☆
Ποιες είναι οι βασικές πράξεις των συνδεδεμένων λιστών;

Οι βασικές πράξεις των συνδεδεμένων λιστών είναι οι παρακάτω:

1. **Εισαγωγή κόμβου** στη λίστα (εισαγωγή κόμβου στην αρχή, στο τέλος της λίστας ή ενδιάμεσα).
2. **Διαγραφή κόμβου** από τη λίστα (διαγραφή από την αρχή, το τέλος της λίστας ή ενδιάμεσα).
3. **Έλεγχος** για το αν η λίστα είναι κενή.
4. **Αναζήτηση κόμβου** για την εύρεση συγκεκριμένου στοιχείου.
5. **Διάσχιση της λίστας** και **προσπέλαση** των στοιχείων της (π.χ. εκτύπωση των δεδομένων που περιέχονται σε όλους τους κόμβους της λίστας).

(σελ 43 Συμπληρωματικό Υλικό)

139 ★★☆☆
Ποιες διαφορές μπορούμε να εντοπίσουμε ανάμεσα στις Λίστες, στις Ουρές και στις Στοίβες;

Οι διαφορές είναι:

1. Στη **θέση εισαγωγής** των στοιχείων: σε μια στοίβα μπορούν να εισαχθούν στοιχεία **μόνο στην κορυφή της**, σε μια ουρά μπορούν να εισαχθούν στοιχεία **μόνο στο πίσω άκρο της** ενώ στη Λίστα μπορούν να εισαχθούν στοιχεία σε οποιοδήποτε σημείο της.



2. Στη **θέση εξαγωγής** των στοιχείων: σε μια στοίβα ένα στοιχείο μπορεί να εξέλθει **μόνο από την κορυφή**, σε μια ουρά ένα στοιχείο μπορεί να εξέλθει **μόνο από την αρχή της**, ενώ σε μια Λίστα ένα στοιχείο μπορεί να εξέλθει από οποιοδήποτε σημείο.



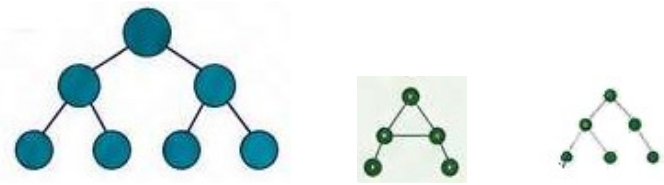
(σελ 42 Συμπληρωματικό Υλικό – εξάγεται από τα συμφραζόμενα)

ΕΝΟΤΗΤΑ 1η • ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΥΛΙΚΟ

Δέντρα και γράφοι

140 ★☆☆☆
Τι ονομάζουμε Δέντρο (στην πληροφορική);

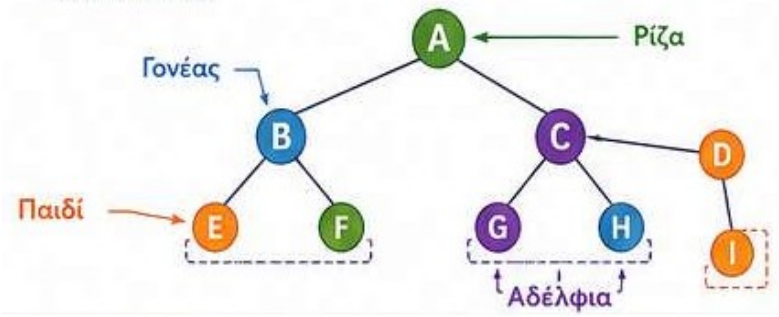
Ένα **δένδρο** είναι μια δομή δεδομένων η οποία αποτελείται από **κόμβους**, οι οποίοι συνδέονται μεταξύ τους με **ακμές**.



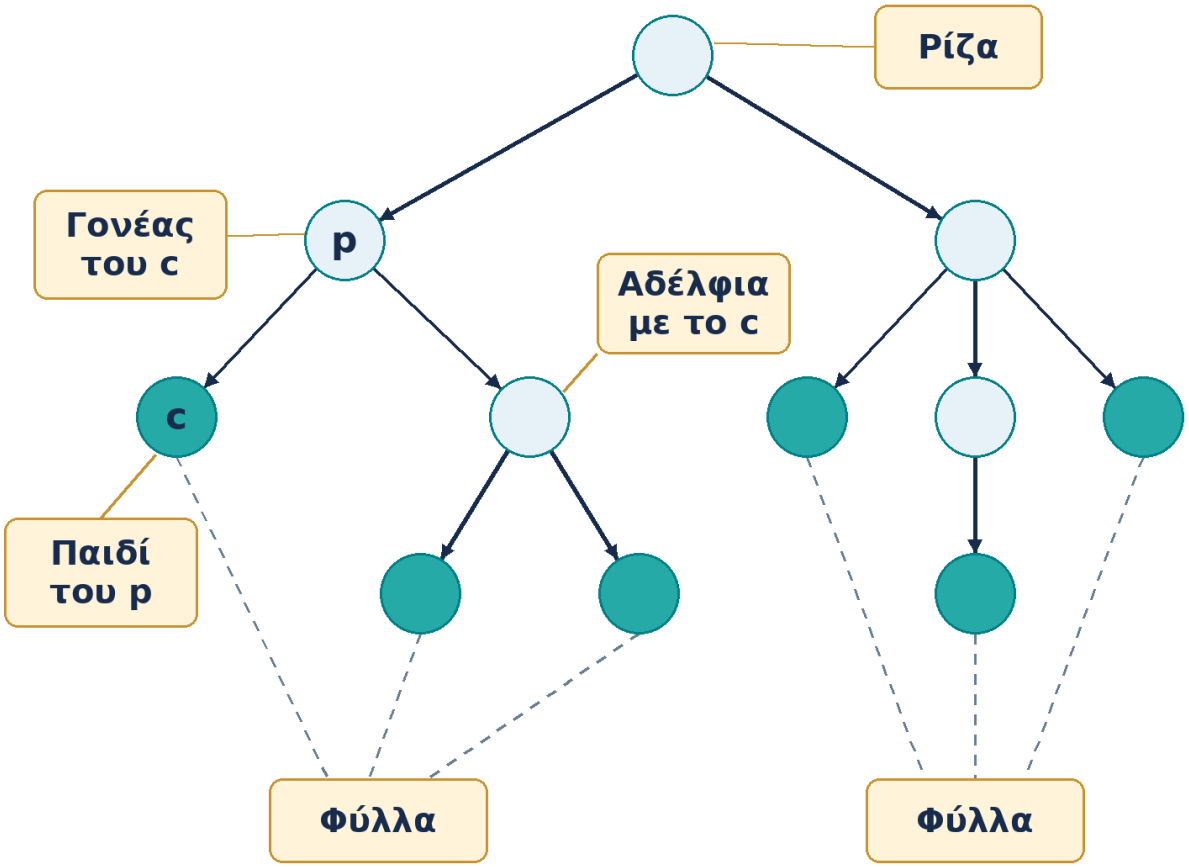
(σελ 43 Συμπληρωματικό Υλικό)

141 ★★☆☆
Τι ονομάζουμε «γονέα», τι «παιδί», τι «ρίζα», τι «αδέλφια» και τι «φύλλα» στα Δέντρα;

1. **Γονέας**: όταν υπάρχει σύνδεση μεταξύ δυο κόμβων με μια ακμή, γονέας λέγεται ο κόμβος από τον οποίο **ξεκινάει η ακμή**.



2. **Παιδί**: όταν υπάρχει σύνδεση μεταξύ δυο κόμβων με μια ακμή, παιδί λέγεται ο κόμβος στον οποίο **καταλήγει η ακμή**.
3. **Ρίζα**: λέγεται ο κόμβος του δέντρου, που δεν έχει γονέα.
4. **Αδέλφια**: λέγονται οι κόμβοι ενός δέντρου, οι οποίοι έχουν **τον ίδιο γονέα**.
5. **Φύλλα**: λέγονται οι κόμβοι ενός δέντρου, οι οποίοι δεν έχουν παιδιά.



(σελ 44 Συμπληρωματικό Υλικό)

142 ★★★☆
Τι ονομάζουμε «Κενό Δέντρο»; Υπάρχει Δέντρο με έναν μόνο κόμβο; Τι είναι τα «υποδέντρα»;

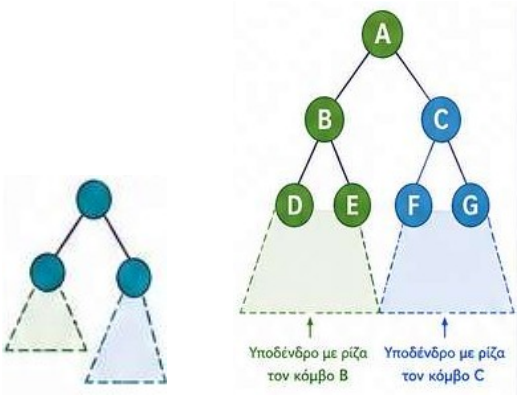
«Κενό» είναι το δένδρο που δεν έχει **ούτε κόμβους, ούτε ακμές** και είναι το μόνο δένδρο χωρίς ρίζα.

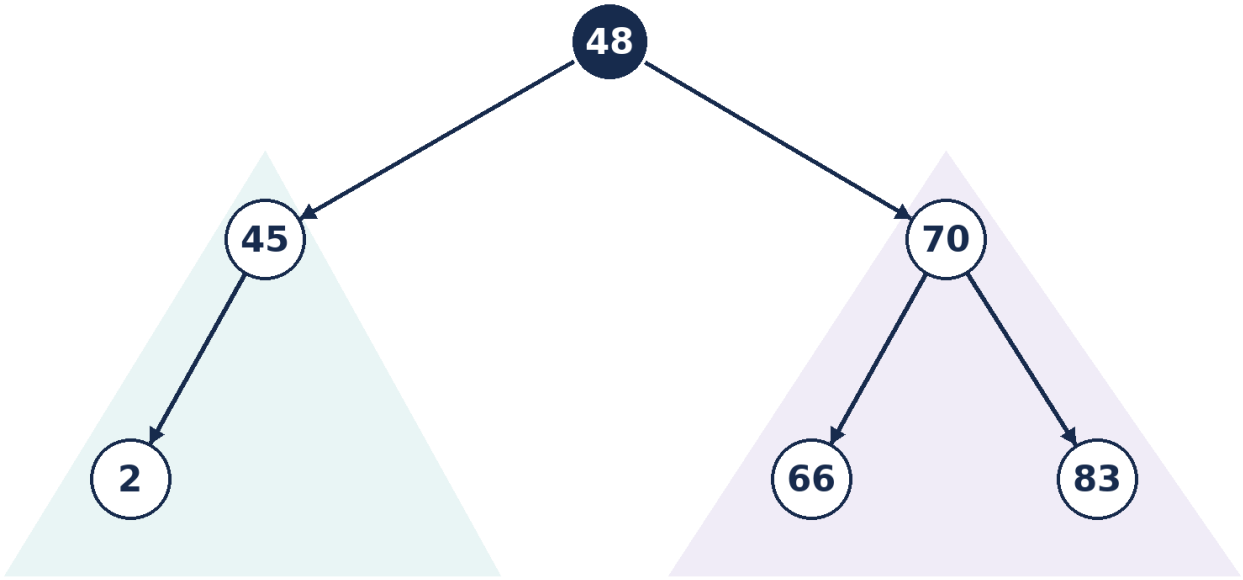


Μπορούμε να έχουμε ένα απλό δένδρο, το οποίο να απαρτίζεται από **έναν μόνο κόμβο**. Αυτός ο κόμβος είναι και **ρίζα** του απλού αυτού δένδρου, διότι δεν έχει γονέα και **φύλλο**, και διότι δεν έχει παιδιά.



Μέσα σε ένα δένδρο μπορούμε να εντοπίσουμε και άλλα μικρότερα δένδρα, που ονομάζονται **υποδένδρα**. Κάθε κόμβος ενός δένδρου μπορεί να θεωρηθεί ως ρίζα ενός υποδένδρου, δηλαδή ενός άλλου μικρότερου δένδρου, που ξεκινάει από τον κόμβο αυτόν.

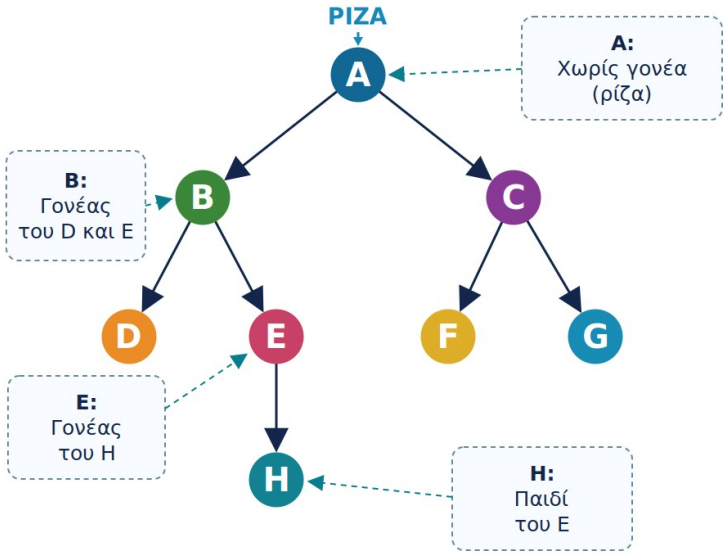




(σελ 45 – 46 Συμπληρωματικό Υλικό)

143 ★★★☆
Ποιοι κανόνες ισχύουν στη δομή δεδομένων Δέντρο;

Ένα δένδρο (tree) είναι μία δομή που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο ακμών μεταξύ των κόμβων με βάση τους εξής κανόνες:



1. Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται **ρίζα**. Αυτός είναι ένας κόμβος **χωρίς γονέα**.
2. Για κάθε κόμβο c, **εκτός από τη ρίζα**, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p.
3. Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.

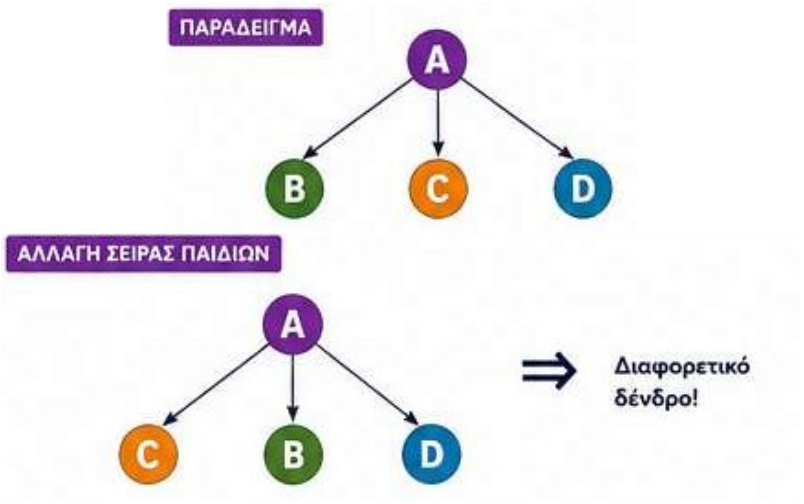
Δένδρο θεωρούμε και το **κενό δένδρο**, δηλαδή το δένδρο που δεν έχει ούτε κόμβους, ούτε ακμές. Το κενό δένδρο είναι το μόνο δένδρο **χωρίς ρίζα**.



(σελ 45 Συμπληρωματικό Υλικό)

144 ★☆☆☆
Τι ονομάζουμε «Διατεταγμένο Δέντρο»;

Όταν για κάθε κόμβο υπάρχει μία γραμμική σχέση μεταξύ των παιδιών του κόμβου αυτού και έχει σημασία η διάταξή τους, αναφερόμαστε σε ένα **διατεταγμένο δένδρο**.



(σελ 46 Συμπληρωματικό Υλικό)

145 ★★★★
Σε ποιους τομείς της επιστήμης των υπολογιστών χρησιμοποιούνται τα δέντρα; Γιατί είναι τόσο ισχυρές δομές;

Τα δέντρα είναι ισχυρές δομές δεδομένων διότι:

1. Είναι πολύ εύκολο **να προσθέσετε, να αφαιρέσετε ή να αναζητήσετε** ένα στοιχείο σε ένα δένδρο.
2. Η δομή των δέντρων **μεταφέρει πληροφορίες**.

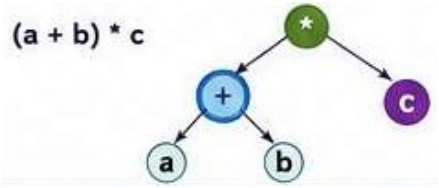


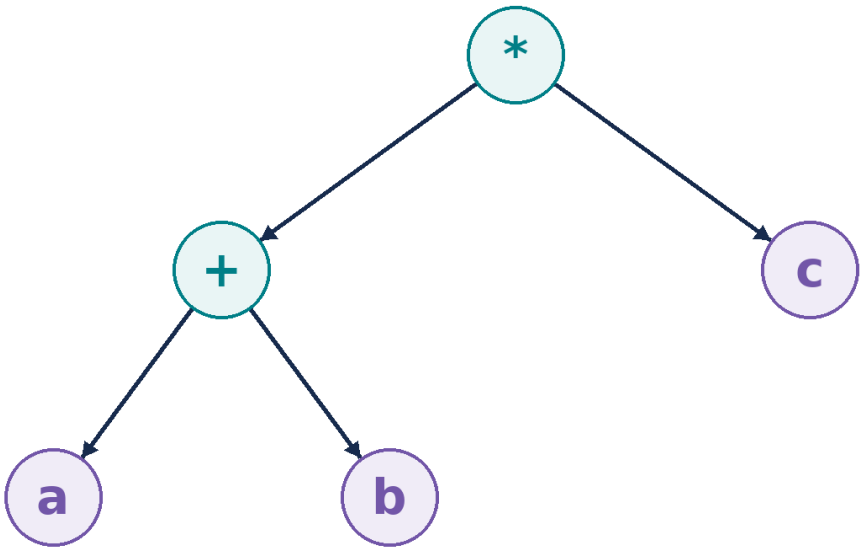
Δέντρα χρησιμοποιούνται σε τομείς όπως η **συμπίεση εικόνων**, η **ταξινόμηση**, η **αυτόματη συμπλήρωση λέξεων** σε συσκευές κινητών τηλεφώνων, η **μεταγλώττιση** ενός προγράμματος και η **λήψη αποφάσεων**. Για παράδειγμα, ένα απλό πρόβλημα λήψης απόφασης μπορεί να είναι αυτό της καθαριότητας του φοιτητικού σπιτιού σας. Κάποια στιγμή θα κληθείτε να πάρετε αποφάσεις λαμβάνοντας υπόψη κάποιους παράγοντες.



Επίσης όταν παίζετε παιχνίδια στον υπολογιστή, όπως είναι το σκάκι, η τρίλιζα, το τάβλι και πολλά άλλα, ο υπολογιστής χρησιμοποιεί ένα ειδικό δένδρο, που ονομάζεται **δένδρο παιχνιδιού (game tree)**, το οποίο μοντελοποιεί όλες τις πιθανές κινήσεις των παικτών για να σας νικήσει.

Τέλος τα δένδρα χρησιμοποιούνται επίσης για την **αναπαράσταση** και κατ' επέκταση τον **υπολογισμό αριθμητικών εκφράσεων** όπως στην εικόνα όπου και υπολογίζεται η παράσταση (a+b)*c.

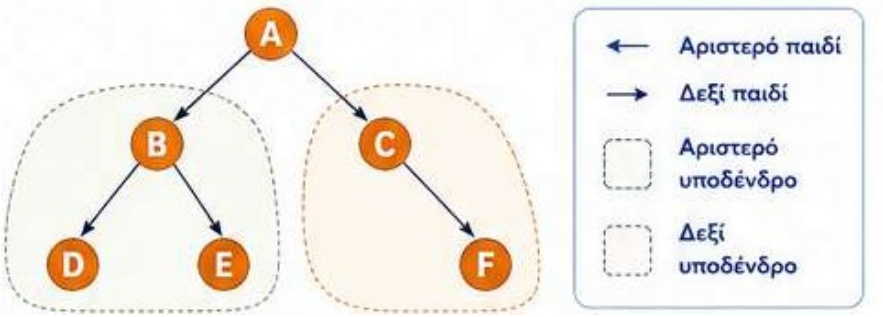


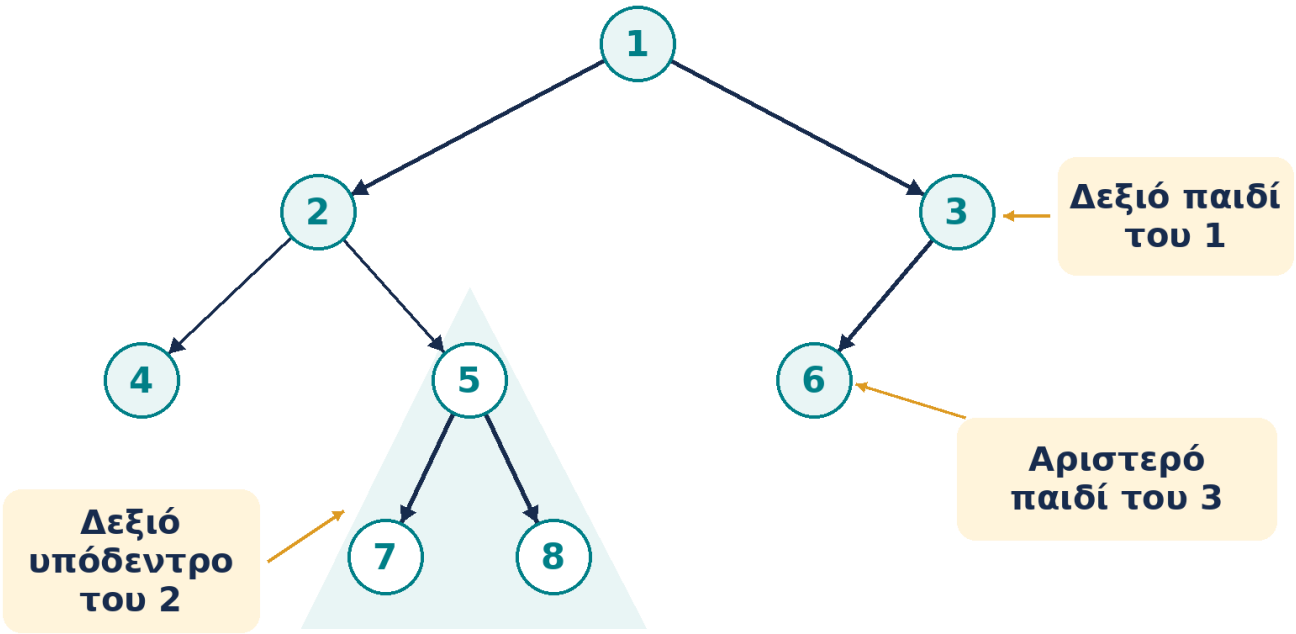


(σελ 47 – 49 Συμπληρωματικό Υλικό)

146 ★★☆☆
Τι ονομάζουμε «Δυαδικό Δέντρο»;

Ένα **δυαδικό δένδρο (binary tree)** είναι ένα διατεταγμένο δένδρο, στο οποίο κάθε κόμβος έχει το πολύ δύο παιδιά, το αριστερό και το δεξί παιδί. Μπορούμε, συνεπώς, να μιλάμε για **αριστερό και δεξιό υποδένδρο** ενός κόμβου. Προφανώς, αν ανταλλάξουμε το αριστερό με το δεξιό υποδένδρο ενός κόμβου παίρνουμε ένα διαφορετικό δένδρο.

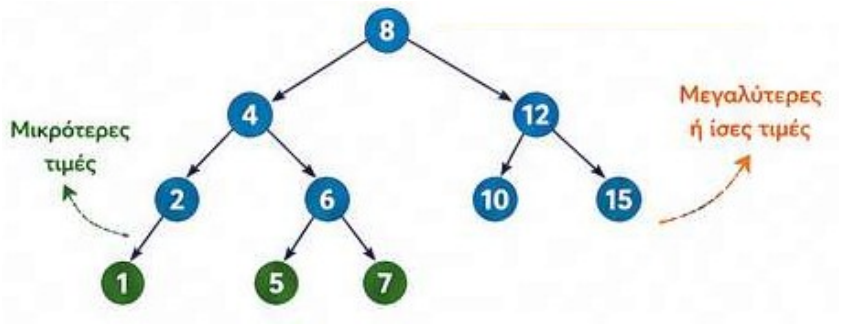


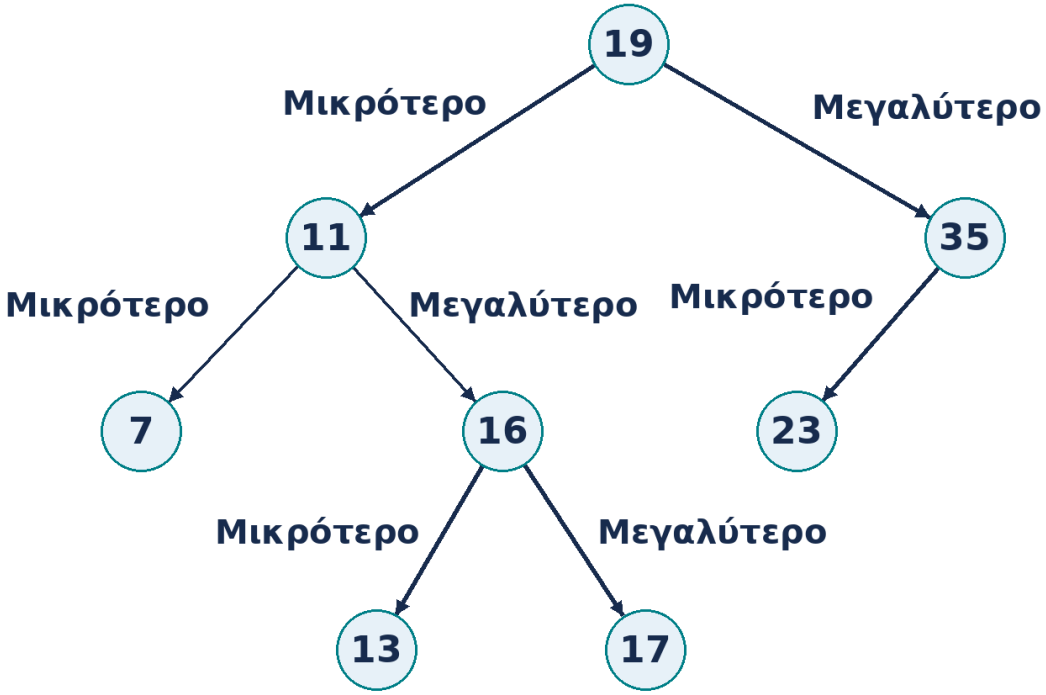


(σελ 50 Συμπληρωματικό Υλικό)

147 ★★☆☆
Τι ονομάζουμε «Δυαδικό Δέντρο Αναζήτησης»;

Ένα **δυαδικό δένδρο αναζήτησης (binary search tree)** είναι ένα δυαδικό δένδρο, όπου για κάθε κόμβο u, όλοι οι κόμβοι του **αριστερού υποδένδρου** έχουν τιμές μικρότερες της τιμής του κόμβου u και όλοι οι κόμβοι του **δεξιού υποδένδρου** έχουν τιμές μεγαλύτερες (ή ίσες) της τιμής του κόμβου u.





(σελ 51 Συμπληρωματικό Υλικό)

148 ★★★★

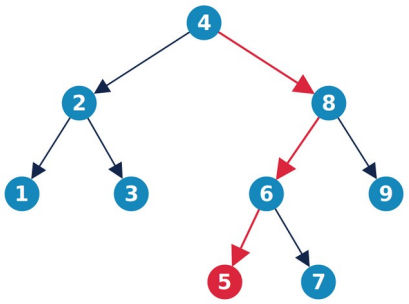
Ποιο είναι το πλεονέκτημα των Δυαδικών Δέντρων Αναζήτησης; Που χρησιμοποιούνται αυτά; Γιατί έχει σημασία η δομή ενός δυαδικού δέντρου αναζήτησης στην εύρεση στοιχείων;

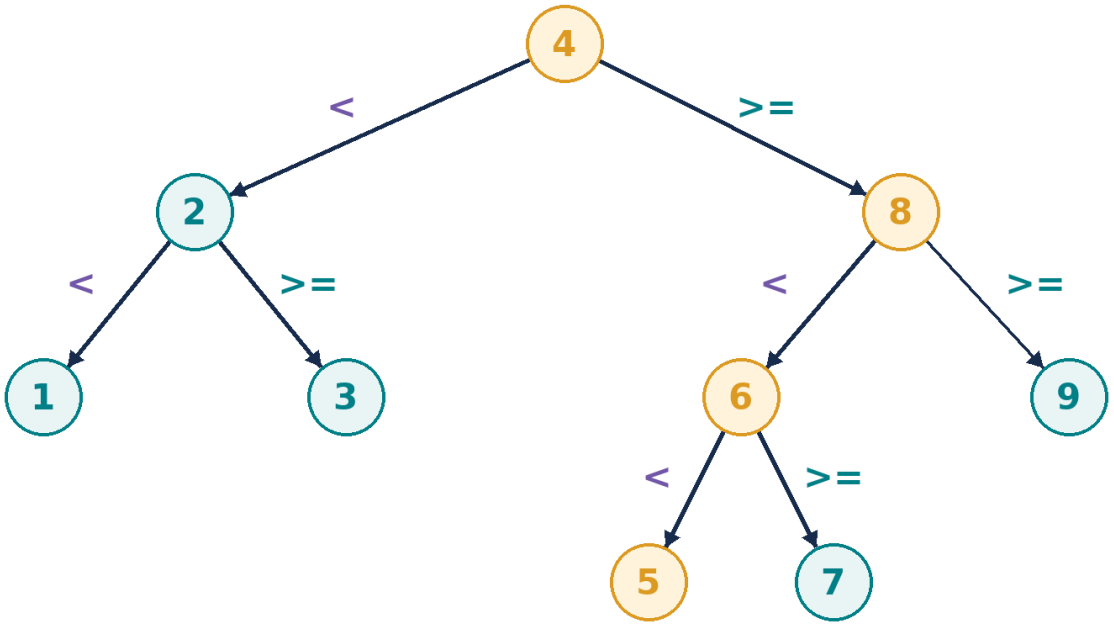


Η αναζήτηση για μια συγκεκριμένη τιμή γίνεται **ταχύτερα** χάρη στον τρόπο αποθήκευσης των τιμών.

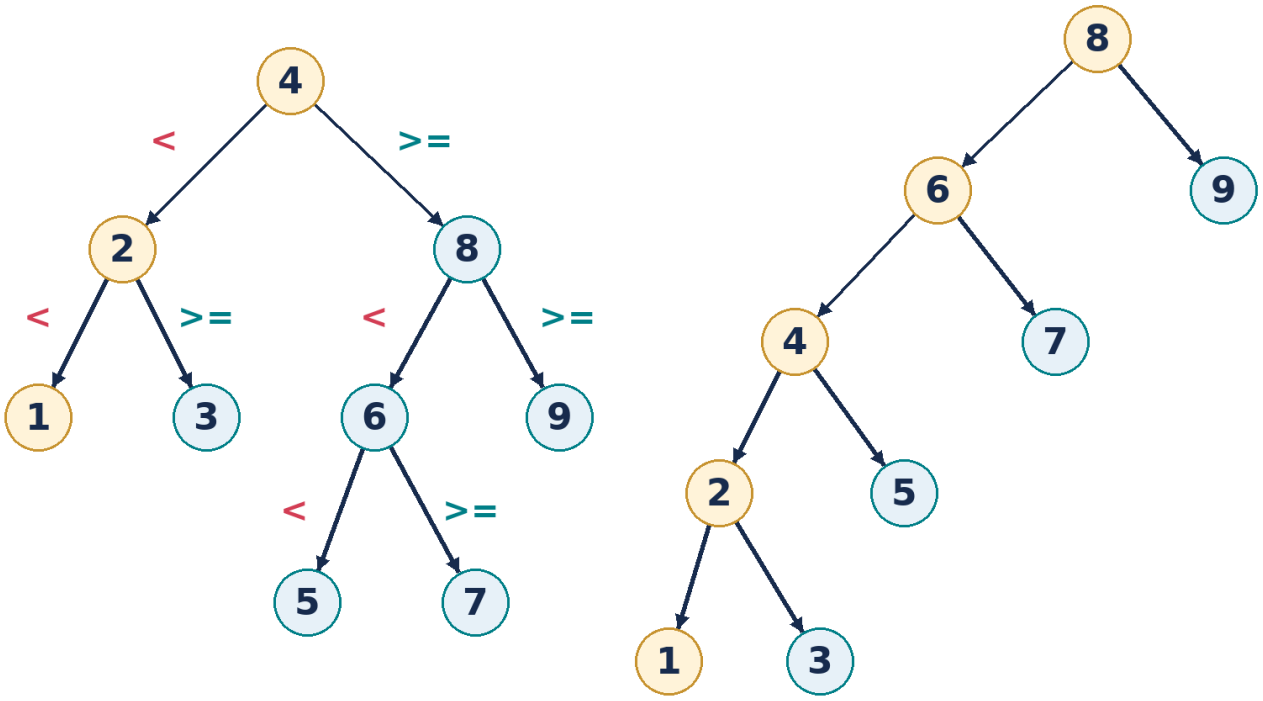


Ας δούμε τον αλγόριθμο της αναζήτησης σε ένα δυαδικό δένδρο αναζήτησης μέσα από την εύρεση του αριθμού 5. Αρχικά, συγκρίνετε τη ρίζα του δένδρου (δηλαδή το 4) με το υπό αναζήτηση στοιχείο (δηλαδή το 5). Ο αριθμός που αναζητάτε είναι μεγαλύτερος από τη ρίζα. Συνεπώς συνεχίζετε με το δεξιό υποδένδρο και αγνοείτε το αριστερό υποδένδρο. Στη συνέχεια, συγκρίνετε το 8 με το 5. Επειδή το 8 είναι μεγαλύτερο από το 5, συνεχίζετε με το αριστερό υποδένδρο. Κατόπιν, συναντάτε το 6 που είναι και αυτό μεγαλύτερο από το 5. Μεταβαίνετε, λοιπόν, στο αριστερό υποδένδρο του 6, όπου βρίσκετε το στοιχείο 5 που αναζητάτε. Στην περίπτωση που δεν υπάρχει το στοιχείο στο δένδρο (για παράδειγμα αν ψάχναμε το στοιχείο 12), η αναζήτηση καταλήγει σε ένα **κενό υποδένδρο** και εκεί τερματίζει η όλη διαδικασία. Με βάση τον αλγόριθμο αυτό, μειώνουμε δραματικά τον χρόνο για να βρούμε το στοιχείο που ψάχνουμε και αυτό διότι **περιορίζουμε αισθητά τους κόμβους** τους οποίους επισκεπτόμαστε. Κάθε φορά αφήνουμε στην άκρη ένα υποδένδρο και συνεχίζουμε με το άλλο.

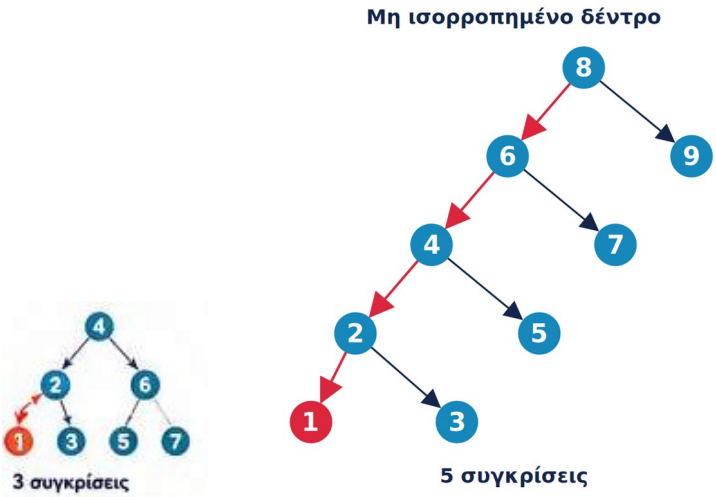




Στα παρακάτω δυο δέντρα έστω ότι αναζητούμε το στοιχείο 1.



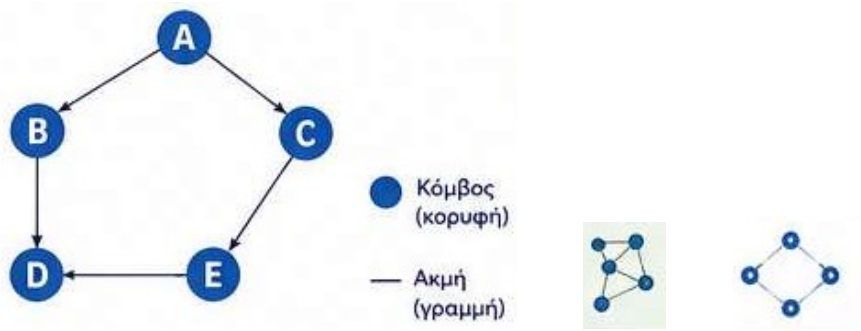
Παρατηρούμε ότι στο αριστερό δέντρο θα γίνουν **3 συγκρίσεις** μέχρι να φτάσουμε στο επιθυμητό. Ενώ στο δεξί θα πραγματοποιηθούν **5 συγκρίσεις**. Συνεπώς το αριστερό δέντρο μπορούμε να πούμε ότι είναι <u>πιο ισορροπημένο</u>.



(σελ 52-53 Συμπληρωματικό Υλικό)

149 ★☆☆☆
Τι ονομάζουμε «Γράφο»;

Ένας **γράφος (graph)** είναι μία δομή που αποτελείται από ένα σύνολο **κόμβων (ή σημείων ή κορυφών)** και ένα σύνολο **γραμμών (ή ακμών ή τόξων)** που ενώνουν μερικούς ή όλους τους κόμβους.



(σελ 54 Συμπληρωματικό Υλικό)

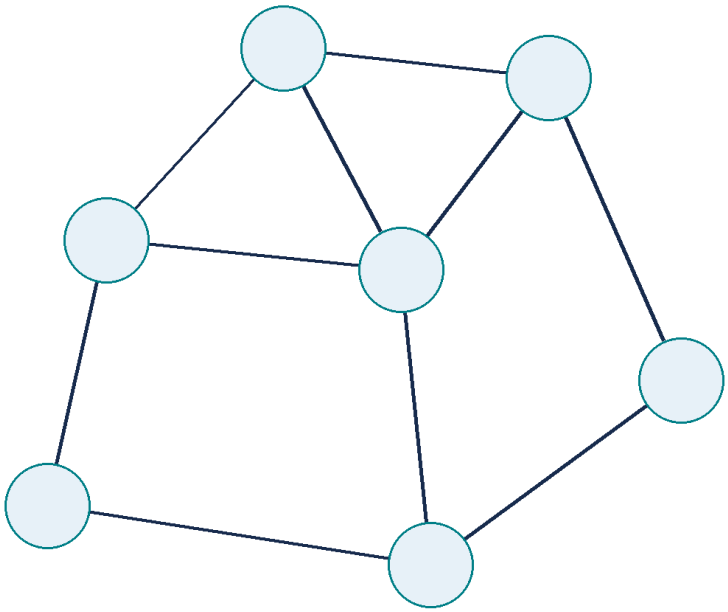
150 ★★★☆
Γιατί ο Γράφος είναι η πιο γενική δομή δεδομένων και γιατί ανήκει στις μη γραμμικές δομές δεδομένων; Υπάρχουν κανόνες στη διάταξη των κόμβων στους Γράφους;

Ο Γράφος αποτελεί την **πιο γενική δομή δεδομένων**, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν περιπτώσεις γράφων.





Οι Γράφοι είναι **μη γραμμικές δομές δεδομένων** καθώς τα δεδομένα τους δεν ακολουθούν μια συγκεκριμένη σειρά αλλά μπορούν να έχουν διάφορες διατάξεις αλλά ακόμα και από κάθε κόμβο να ξεκινούν ή να καταλήγουν περισσότεροι από ένας κόμβοι χωρίς να υπάρχει η έννοια της ρίζας.



Σε έναν γράφο δεν υπάρχουν πραγματικοί κανόνες για τον τρόπο που ένας κόμβος συνδέεται με έναν άλλο κόμβο. Οι ακμές μπορούν να συνδέσουν τους κόμβους με οποιονδήποτε τρόπο. Οι διαφορετικοί τύποι ακμών είναι πολύ σημαντικοί όταν πρόκειται για την αναγνώριση και τον καθορισμό του τύπου των γράφων.

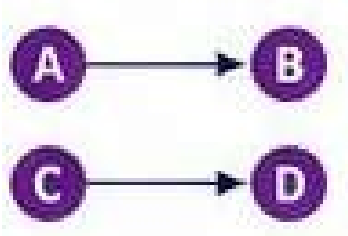
(σελ 54-55 Συμπληρωματικό Υλικό)

151 ★★★☆

Ποια είδη ακμών διακρίνουμε στους Γράφους; Ποια είδη γράφων έχουμε; Αναφέρετε παραδείγματα.

Οι γράφοι μπορούν να έχουν δύο τύπους ακμών:

1. ακμές που έχουν κατεύθυνση (**κατευθυνόμενες ακμές**) και



2. ακμές που δεν έχουν κατεύθυνση (**μη κατευθυνόμενες ακμές**). Σε μια κατευθυνόμενη ακμή, δύο κόμβοι συνδέονται με πολύ συγκεκριμένο τρόπο.





α: Κατευθυνόμενη ακμή

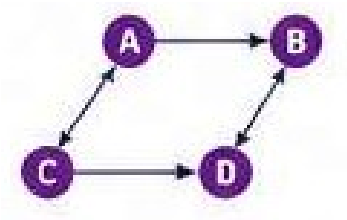


β: μη κατευθυνόμενη ακμή

Υπάρχουν δυο τύποι γράφων:

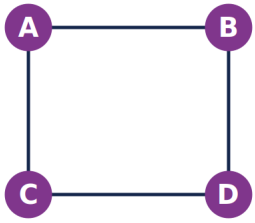



1. **Κατευθυνόμενος**: στον οποίο <u>όλες οι ακμές του έχουν κατεύθυνση</u>. Παραδείγματα είναι ο **παγκόσμιος ιστός (www)** και το **twitter** (γιατί δεν απαιτείται follow back)





2. **Μη κατευθυνόμενος**: στον οποίο <u>όλες οι ακμές του δεν έχουν κατεύθυνση</u>. Παράδειγμα είναι το **Facebook** (γιατί άπαξ και γίνει αποδοχή φιλίας τότε υπάρχει αμφίδρομη σχέση μεταξύ των χρηστών).

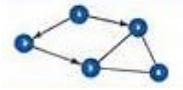


(σελ 55-56 Συμπληρωματικό Υλικό)



152 ★★★★
Ποιες είναι οι διαφορές ανάμεσα σε ένα Δένδρο και σε έναν Γράφο;

1. Ένα δένδρο μπορεί μόνο να ρέει **προς μία κατεύθυνση**, από τον κόμβο ρίζας σε κόμβους φύλλων ή κόμβους παιδιών. Οι γράφοι **δεν έχουν την έννοια ενός κόμβου «ρίζας»**.

2. Ένα δένδρο μπορεί να έχει **μόνο μονόδρομες συνδέσεις** ενώ ένας γράφος μπορεί να έχει **μονόδρομες αλλά και αμφίδρομες συνδέσεις**.




3. Στο δέντρο ένα παιδί μπορεί να έχει **έναν μόνο γονέα** ενώ στους γράφους μπορεί να έχει **περισσότερους γονείς**.




4. Ένα δένδρο <u>δεν μπορεί να έχει βρόχους ή κυκλικούς δεσμούς</u> ενώ ένας γράφος μπορεί.



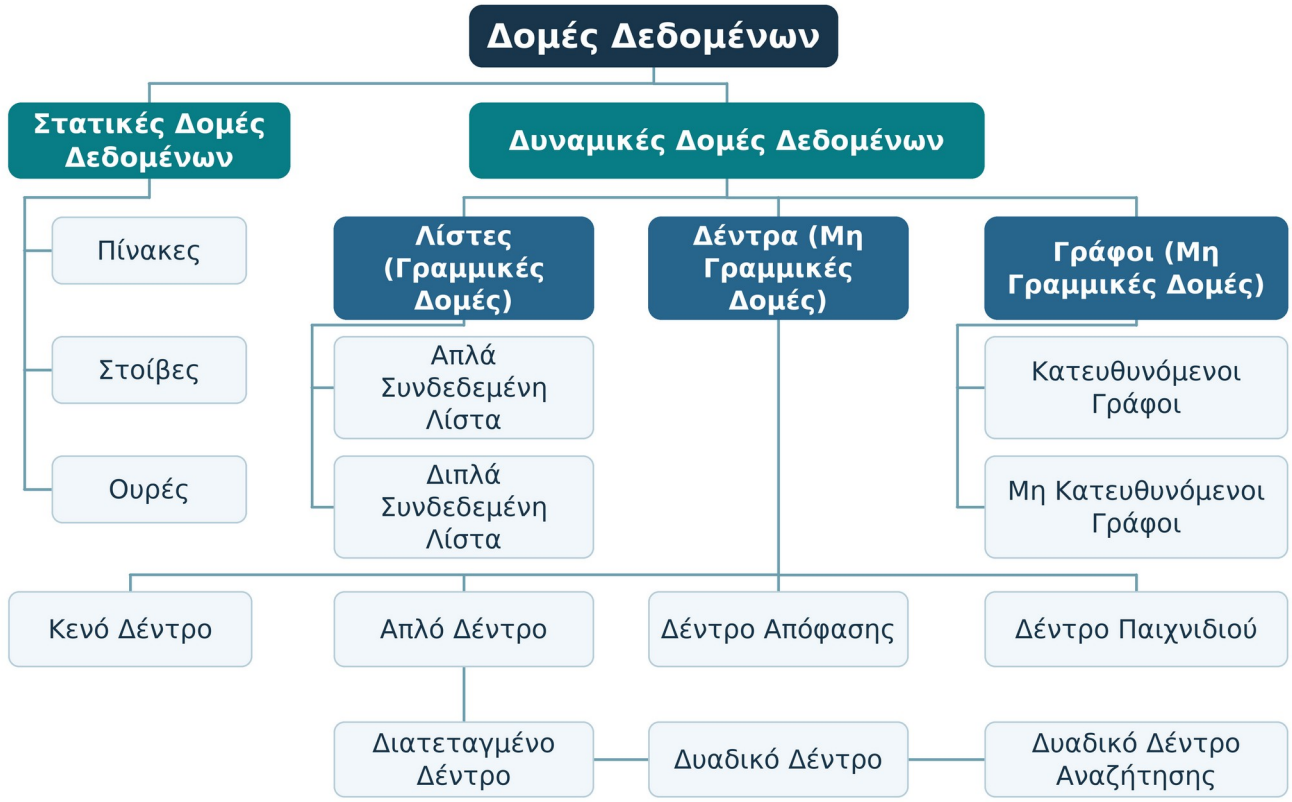


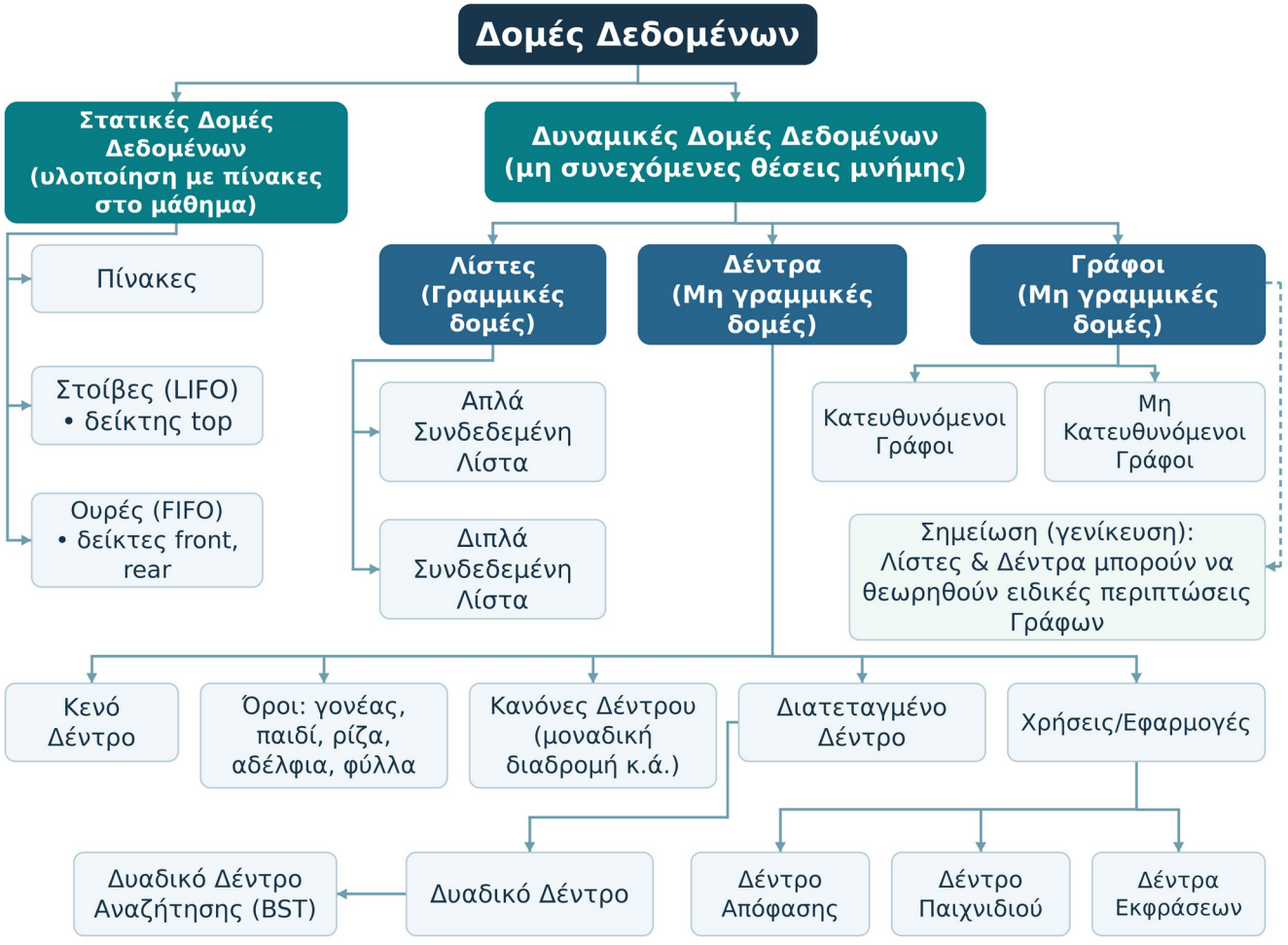

5. **Τα δένδρα έχουν κανόνες** ενώ **οι γράφοι δεν διέπονται από κάποιους κανόνες** στη σύνδεση των κόμβων τους.
6. Σε κάθε κόμβο ενός δέντρου καταλήγει πάντα **μόνο μια ακμή** (εκτός αν είναι ρίζα) ενώ σε κάθε κόμβο ενός γράφου καταλήγουν **μια ή περισσότερες ακμές**.




(σελ 54 Συμπληρωματικό Υλικό – δεν αποτελεί ξεκάθαρη ερώτηση σχολική βιβλίου)

Εννοιολογικό Διάγραμμα Δομών Δεδομένων Μαθήματος





(επίσης οι Λίστες και τα Δένδρα είναι υποκατηγορίες των Γράφων)

Προτεινόμενη διάταξη

1) Δομές Δεδομένων

1.1 Στατικές Δομές Δεδομένων

- **Πίνακες**
- **Στοίβες** (υλοποίηση με πίνακα, δείκτης top) *(124–125)*
- **Ουρές** (υλοποίηση με πίνακα, δείκτες front, rear) *(126–127)*

1.2 Δυναμικές Δομές Δεδομένων *(128)*

1.2.1 Λίστες (Γραμμικές Δυναμικές Δομές) *(129–139)*

- **Απλά Συνδεδεμένη Λίστα** *(129–131)*
- **Διπλά Συνδεδεμένη Λίστα** *(134)*
- Βασικές έννοιες/μέρη:
  - **Κόμβος**: Δεδομένα + Δείκτης *(130)*
  - **Κεφαλή (Head)** *(129–131)*
  - **NULL** *(129–130)*
- Βασικές πράξεις *(138)*:
  - Εισαγωγή κόμβου (αρχή/τέλος/ενδιάμεσα)
  - Διαγραφή κόμβου (αρχή/τέλος/ενδιάμεσα)
  - Έλεγχος κενής λίστας
  - Αναζήτηση
  - Διάσχιση/προσπέλαση
- Σύγκριση με πίνακες *(136–137)*
- Διαφορές Λίστας–Στοίβας–Ουράς *(139)*
- Χρήση λιστών *(135)*: υλοποίηση στοίβας/ουράς (στο υλικό σου αναφέρεται έτσι)

1.2.2 Δέντρα (Μη Γραμμικές Δυναμικές Δομές) *(140–148)*

- **Ορισμός Δέντρου** *(140)*
- **Βασικοί όροι** *(141)*:
  - γονέας, παιδί, ρίζα, αδέλφια, φύλλα
- **Ειδικές περιπτώσεις/έννοιες** *(142)*:
  - **Κενό Δέντρο**
  - **Δέντρο με έναν μόνο κόμβο**
  - **Υποδέντρα**
- **Κανόνες Δέντρου** *(143)*:
  - μοναδική ρίζα (εκτός από κενό)
  - μία ακμή προς κάθε κόμβο (εκτός ρίζας)
  - μοναδική διαδρομή από ρίζα σε κάθε κόμβο
- **Διατεταγμένο Δέντρο** *(144)*
  - (όπως στο υλικό σου) **η διάταξη των παιδιών έχει σημασία**
- **Δυαδικό Δέντρο** *(146)*
  - κάθε κόμβος: το πολύ 2 παιδιά (αριστερό/δεξί)
  - αριστερό/δεξί υποδένδρο
- **Δυαδικό Δέντρο Αναζήτησης (BST)** *(147–148)*
  - αριστερά < τιμή κόμβου, δεξιά > ή = (όπως το έχεις)
  - πλεονέκτημα: ταχύτερη αναζήτηση, εξαρτάται από ισορροπία *(148)*
- **Χρήσεις/εφαρμογές δέντρων** *(145)*:
  - συμπίεση εικόνων, ταξινόμηση, autocomplete, μεταγλώττιση, λήψη αποφάσεων
  - **Δέντρο Απόφασης**
  - **Δέντρο Παιχνιδιού (game tree)**
  - αναπαράσταση αριθμητικών εκφράσεων

1.2.3 Γράφοι (Μη Γραμμικές Δυναμικές Δομές) *(149–152)*

- **Ορισμός Γράφου** *(149)*
- **Γιατί είναι γενική δομή / μη γραμμική** *(150)*
- **Είδη ακμών** *(151)*:
  - κατευθυνόμενες / μη κατευθυνόμενες
- **Είδη γράφων** *(151)*:
  - **Κατευθυνόμενος γράφος** (π.χ. www, twitter)
  - **Μη κατευθυνόμενος γράφος** (π.χ. Facebook)
- **Διαφορές Δέντρου-Γράφου** *(152)*:
  - ρίζα, βρόχοι/κύκλοι, κανόνες, αριθμός γονέων, κλπ.

ΕΝΟΤΗΤΑ 4η • ΚΕΦΑΛΑΙΟ 6ο

Αντικειμενοστραφής προγραμματισμός

ΕΝΟΤΗΤΑ 4η Συμπληρωματικό Υλικό – ΚΕΦΑΛΑΙΟ 6ο Σχολικό Βιβλίο

153 ★★★☆

Τι ονομάζουμε Αντικειμενοστραφή Προγραμματισμό;




Αντικειμενοστραφής προγραμματισμός (object-oriented programming) ή αντικειμενοστραφής σχεδίαση είναι μια μεθοδολογία ανάπτυξης εφαρμογών η οποία στηρίζεται σε αυτόνομες προγραμματιστικές οντότητες με δική τους ταυτότητα και συμπεριφορά. Οι οντότητες αυτές καλούνται **αντικείμενα (objects)**, αντιστοιχούν σε φυσικές οντότητες ή έννοιες του φυσικού μας κόσμου, και δομούνται με βάση **δεδομένα (ιδιότητες)** που προσδιορίζουν την υπόστασή τους και **ενέργειες (κανόνες συμπεριφοράς)** που εφαρμόζονται πάνω στα δεδομένα.

(σελ 86 Συμπληρωματικό Υλικό)

154 ★★☆☆

Τι ονομάζονται Αντικείμενα, τι Ιδιότητες και τι Μέθοδοι στον Αντικειμενοστραφή προγραμματισμό;

Σε μια εφαρμογή, ένα **Αντικείμενο** είναι ο ομαδοποιημένος συνδυασμός δεδομένων και κώδικα, τα οποία έχουμε τη δυνατότητα να χειριστούμε ενιαία.



Τα δεδομένα αποτελούν τα **χαρακτηριστικά** ενός αντικειμένου και αναφέρονται ως **Ιδιότητες (properties)**.

Οι ενέργειες καθορίζουν τη **συμπεριφορά** του και στον αντικειμενοστραφή προγραμματισμό αυτές οι ενέργειες αναφέρονται και ως **Μέθοδοι (methods)**.

(σελ 86 Συμπληρωματικό Υλικό)

155 ★★★☆

Ποια είναι τα κύρια δομικά στοιχεία ενός προγράμματος στην Αντικειμενοστραφής σχεδίαση; Ποια πλεονεκτήματα προσφέρει αυτή η σχεδίαση; Ποιες τεχνικές χρησιμοποιεί ο Αντικειμενοστραφής προγραμματισμός;

Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος **τα δεδομένα**, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση **τα αντικείμενα (objects)**.




Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.



Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την **ιεραρχική σχεδίαση**, τον **τμηματικό προγραμματισμό** και ακολουθεί τις αρχές του **δομημένου προγραμματισμού**.

(σελ 120 Σχολικό Βιβλίο)

156 ★★★☆
Από πού πηγάζουν οι θεμελιώδεις αρχές του Αντικειμενοστραφούς Προγραμματισμού;

Οι θεμελιώδεις αρχές του αντικειμενοστραφούς προγραμματισμού πηγάζουν από τον **καθημερινό μας φυσικό κόσμο**, καθώς είναι πολύ κοντά στον τρόπο που σκεφτόμαστε για να επιλύσουμε προβλήματα της καθημερινότητάς μας. Στηρίζονται στο γεγονός ότι για να μπορέσει κάποιος να κατανοήσει άγνωστες σε αυτόν έννοιες, θα πρέπει να καθοδηγηθεί μέσω της **προσομοίωσης** των άγνωστων αυτών εννοιών αντιστοιχίζοντας αυτές σε πρακτικές γνώσεις και εικόνες από το περιβάλλον του, τις οποίες γνωρίζει και μπορεί πολύ εύκολα να χειριστεί.



(σελ 86 Συμπληρωματικό Υλικό)

157 ★★★☆
Με ποιόν τρόπο «χτίζουμε» Αντικειμενοστραφή Προγράμματα και ποια μεθοδολογία χρησιμοποιούμε;

Το «χτίσιμο» μιας αντικειμενοστραφούς εφαρμογής επιτυγχάνεται με τη **δημιουργία και τον χειρισμό αντικειμένων** τα οποία πρέπει να συνεργαστούν για την επίτευξη του κοινού στόχου που είναι η επίλυση του προβλήματος.

Έτσι αναλύουμε το πρόβλημα το οποίο θέλουμε να επιλύσουμε, δηλαδή να αναγνωρίσουμε και να καταγράψουμε τα βασικά συστατικά στοιχεία της διαδικασίας επίλυσής του που είναι:

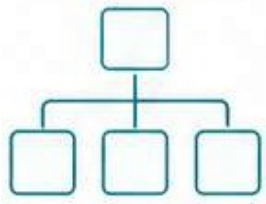
1. **τα αντικείμενα** που συμμετέχουν με βάση τον **ρόλο τους** στο συγκεκριμένο σενάριο,
2. **οι ιδιότητες** κάθε αντικειμένου, δηλ. τα σχετικά με το συγκεκριμένο πρόβλημα χαρακτηριστικά του, και
3. **οι υπηρεσίες** που προσφέρει ή **οι ενέργειες** που υλοποιεί κάθε αντικείμενο (**μέθοδοι**) προς αξιοποίηση από άλλες, ώστε να αναπτυχθούν οι απαραίτητες συνεργασίες μεταξύ των αντικειμένων για την επίλυση του προβλήματος.

(σελ 88 Συμπληρωματικό Υλικό)

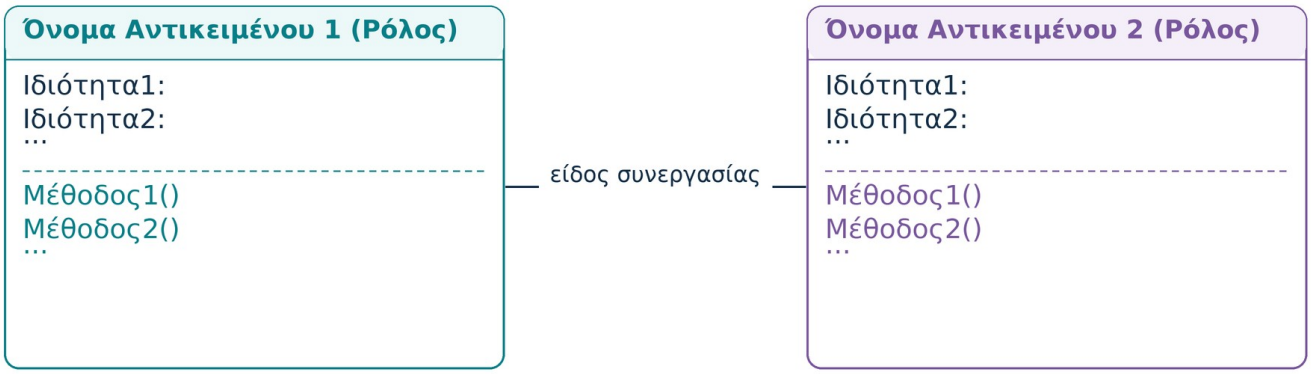
158 ★☆☆☆

Γιατί είναι σημαντική η Διαγραμματική Αναπαράσταση των αντικειμένων στη δημιουργία μίας εφαρμογής με αντικειμενοστραφή προσέγγιση;

Η αναπαράσταση είναι ιδιαίτερα σημαντική, διότι μας δίνει την **εποπτική εικόνα των συνεργαζόμενων οντοτήτων** του προβλήματός μας και ουσιαστικά αποτελεί το σχέδιο επίλυσής του με βάση την αντικειμενοστραφή προσέγγιση.







(σελ 90 Συμπληρωματικό Υλικό)

159 ★★☆☆

Πώς δομείται ένα αντικειμενοστραφές πρόγραμμα;

Ένα αντικειμενοστραφές πρόγραμμα δομείται ως ένα **δίκτυο συνεργαζόμενων οντοτήτων** που είναι **τα αντικείμενα**. Κάθε αντικείμενο έχει ένα συγκεκριμένο ρόλο στην εφαρμογή και παρέχει μια υπηρεσία ή εκτελεί μια ενέργεια (**μέθοδο**) που χρησιμοποιείται από άλλα μέλη του δικτύου, δηλαδή από άλλα αντικείμενα, για την υλοποίηση της συνεργασίας που θα επιλύσει το πρόβλημα.



(σελ 91 Συμπληρωματικό Υλικό)

160 ★★☆☆

Τι ονομάζεται «Ενθυλάκωση» στον Αντικειμενοστραφή προγραμματισμό;





Η δυνατότητα ενός αντικειμένου να συνδυάζει εσωτερικά τα δεδομένα και τις μεθόδους χειρισμού του καλείται **ενθυλάκωση (encapsulation)**. Σε μια αντικειμενοστραφή εφαρμογή κάθε αντικείμενο αποτελεί ξεχωριστή οντότητα και περιέχει ενσωματωμένες τις **ιδιότητες (δεδομένα)** και τους **κανόνες συμπεριφοράς του (μεθόδους)**. Την ενθυλάκωση μπορούμε να την παρομοιάσουμε σαν ένα κέλυφος που υπάρχει γύρω από κάθε αντικείμενο και διαχωρίζει τον εσωτερικό από τον εξωτερικό του κόσμο.



(σελ 92 Συμπληρωματικό Υλικό)

161 ★★★☆

Τι ονομάζεται «Κλάση» στον Αντικειμενοστραφή προγραμματισμό; Δώστε ένα παράδειγμα από τον πραγματικό κόσμο.

Ο γενικός τύπος ενός αντικειμένου καλείται **κλάση (class)** και καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικειμένου που προέρχεται από αυτή. Μια κλάση αποτελεί ένα **αφαιρετικό (abstract) στοιχείο** (τύπο) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων αντικειμένων.

Ένα παράδειγμα από τον πραγματικό κόσμο είναι τα Σκυλιά. Ένας σκύλος μπορεί να αποτελέσει μια κλάση με βάση την οποία έχουμε τη δυνατότητα να δημιουργούμε απεριόριστα διαφορετικά σκυλάκια για καθένα από τα οποία θα ορίζουμε τη **ράτσα, το μέγεθος, την ηλικία και το χρώμα** (ιδιότητες), όπως επίσης και τις δραστηριότητες που θα μπορεί να έχει, π.χ. **Τρώω(), Κοιμάμαι(), Κάθομαι(), Τρέχω()** και ότι άλλο θέλουμε να μπορούν να κάνουν τα σκυλάκια (μέθοδοι)!!!





(σελ 92 – 93 Συμπληρωματικό Υλικό)

162 ★★★★

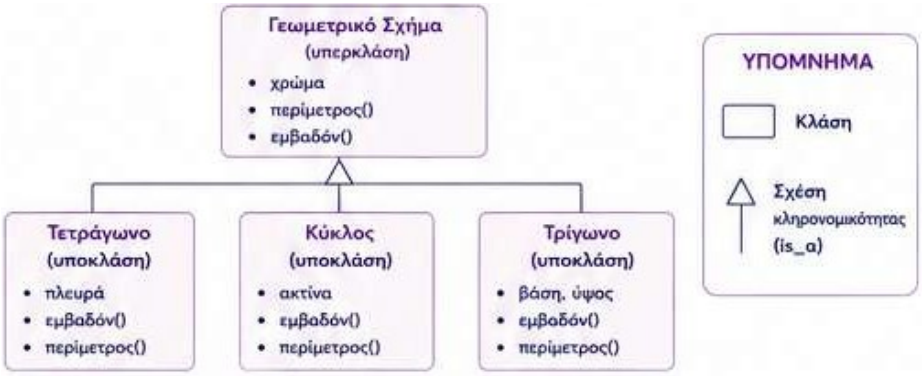
Τι ονομάζουμε «Κληρονομικότητα» στον Αντικειμενοστραφή Προγραμματισμό; Τι είναι οι κλάσεις πρόγονοι (ή υπερκλάσεις) και οι κλάσεις απόγονοι (ή υποκλάσεις);

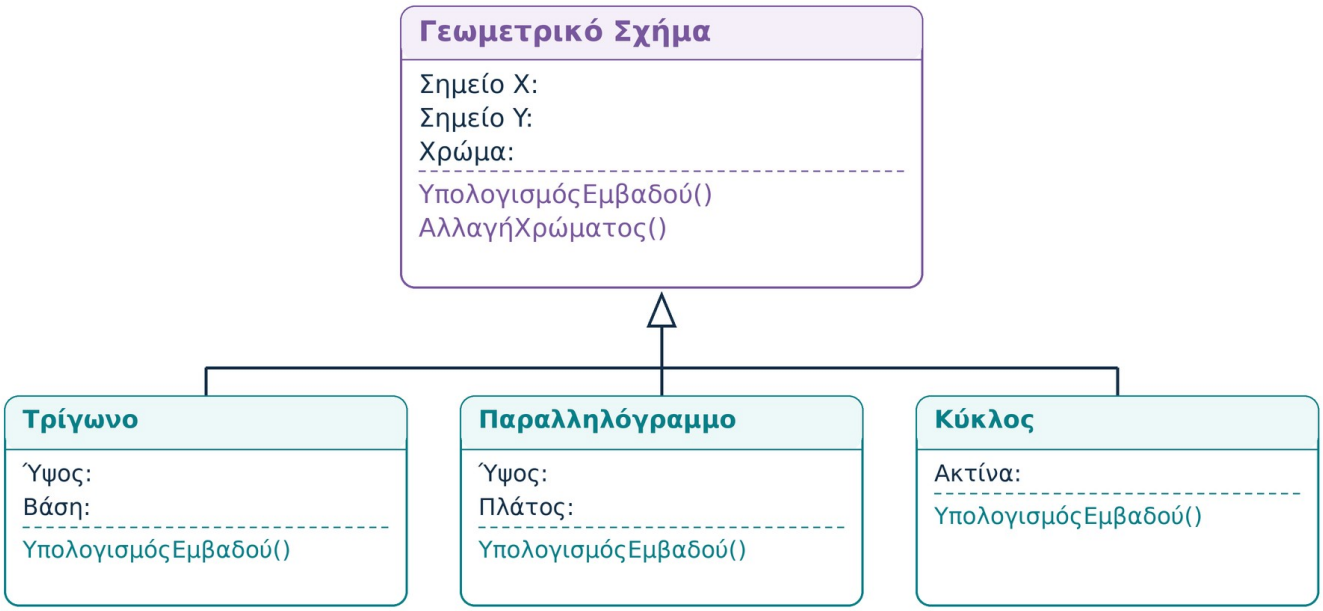
Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων καλείται **κληρονομικότητα (inheritance)**. Με βάση την κληρονομικότητα, μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης να οριστούν υποκλάσεις αντικειμένων.



Η **κλάση απόγονος (υποκλάση)** κληρονομεί και μπορεί να χρησιμοποιήσει όλα τα δεδομένα (ιδιότητες) και τις μεθόδους που περιέχει η **κλάση πρόγονος (υπερκλάση)** ενώ η κλάση πρόγονος περιλαμβάνει τις **κοινές ιδιότητες και μεθόδους** όλων των κλάσεων απογόνων της. Οπότε οι κλάσεις απόγονοι μπορούν να περιέχουν μόνο τις διαφορετικές τους ιδιότητες και μεθόδους αφού τις κοινές τις κληρονομούν από την κλάση πρόγονο και δεν χρειάζεται να τις περιέχουν.

Στο παρακάτω σχήμα φαίνεται η σχέση κληρονομικότητας μεταξύ της υπερκλάσης και των υποκλάσεών του για το Γεωμετρικό Σχήμα.



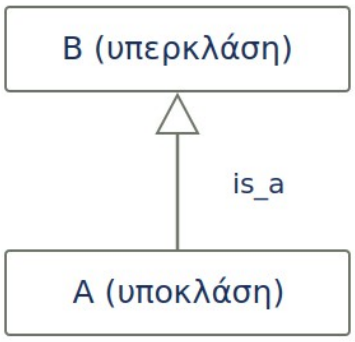


(σελ 99 – 101 Συμπληρωματικό Υλικό)

163 ★★★★

Πως ορίζεται η σχέση κληρονομικότητας σε ένα διάγραμμα κλάσεων; τι σημαίνει το “is_a” και πότε λέμε ότι μια κλάση Α είναι έγκυρη υποκλάση της Β;

Η διαγραμματική αναπαράσταση της σχέσης κληρονομικότητας που μόλις περιγράψαμε γίνεται με τη βοήθεια του **ειδικού συμβόλου γενίκευσης**  .



Όταν θέλουμε να πούμε ότι μια κλάση ανήκει σε μια άλλη, τότε χρησιμοποιούμε τον συμβολισμό “**is_a**”.

A is_a B

Μια κλάση Α μπορεί να είναι **έγκυρη υποκλάση** της Β αν έχει νόημα να πούμε «ένα Α είναι ένα (is_a) Β».

(Σημ. συντ.: για να δείτε αν ισχύει αυτό κάντε τις εξής ερωτήσεις στο μυαλό σας: 1. Η κλάση Α είναι κλάση Β; 2. Η κλάση Α εντάσσεται στη Β; 3. **Να μην είναι αντικείμενο**. Και αν οι απαντήσεις είναι και οι 3 θετικές τότε ισχύει η σχέση κληρονομικότητας. Για παράδειγμα στη φράση «Το αυτοκίνητο is_a μέσο μεταφοράς» σκεφτόμαστε ότι το αυτοκίνητο είναι μέσο μεταφοράς και εντάσσεται στα μέσα μεταφοράς και δεν είναι αντικείμενο, οπότε ισχύει η σχέση υπερκλάσης – υποκλάσης. Αυτό ισχύει και για να βρούμε αν ένα ζευγάρι είναι **έγκυρο ζευγάρι υπερκλάσης και υποκλάσης** όπως για παράδειγμα το ζεύγος «Νόμισμα – Ευρώ» όπου το ευρώ είναι νόμισμα, ανήκει στα νομίσματα αλλά είναι αντικείμενο, οπότε δεν είναι έγκυρο ζευγάρι)





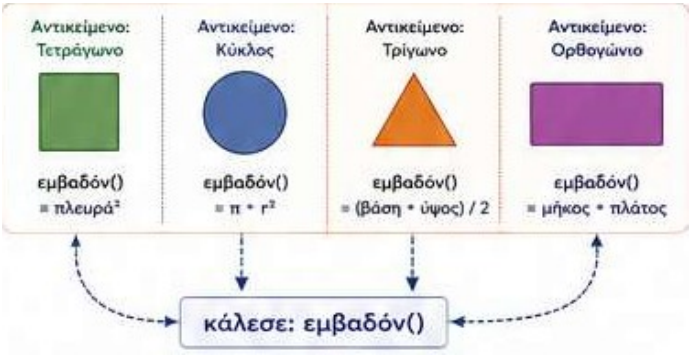
(σελ 99 – 102 Συμπληρωματικό Υλικό)

164 ★★★★
Τι ονομάζουμε «Πολυμορφισμό»; Πως υλοποιείται ο Πολυμορφισμός στον αντικειμενοστραφή προγραμματισμό;

Πολυμορφισμός (polymorphism) είναι μια ιδιότητα του αντικειμενοστραφούς προγραμματισμού με την οποία μια λειτουργία μπορεί να υλοποιείται με πολλούς διαφορετικούς τρόπους.

Οι συμπεριφορές των αντικειμένων στον αντικειμενοστραφή προγραμματισμό μπορούν να είναι επίσης πολυμορφικές. Τα αντικείμενα μπορούν δηλαδή να υποστηρίζουν συμπεριφορές (μεθόδους) με **κοινό όνομα** και τον **ίδιο βασικό σκοπό** αλλά με **διαφορετική λειτουργική υλοποίηση**. Κάθε φορά που καλείται μια πολυμορφική λειτουργία, το πρόγραμμα αποφασίζει ποια από τις διαφορετικές μεθόδους με την ίδια ονομασία θα ενεργοποιηθεί, με βάση την **κλάση του αντικειμένου** στην οποία απευθύνεται η εφαρμογή της λειτουργίας (να σημειωθεί πως μόνο οι μέθοδοι μπορεί να είναι πολυμορφικές και όχι οι ιδιότητες μίας κλάσης).

Για παράδειγμα ο υπολογισμός εμβαδού γεωμετρικών σχημάτων:



Τρίγωνο	Παραλληλόγραμμο	Κύκλος
Εμ <- Βάση*Ύψος/2	Εμ <- Μήκος*Πλάτος	Εμ <- 3.14*Ακτίνα*Ακτίνα

Η λειτουργία υπολογισμού του εμβαδού λοιπόν, είναι πολυμορφική και, κατά την εκτέλεση της εφαρμογής, ενεργοποιείται η **κατάλληλη μέθοδος υπολογισμού του εμβαδού** ανάλογα με το γεωμετρικό σχήμα. Μια υλοποίηση των αντίστοιχων συναρτήσεων υπολογισμού του εμβαδού (μέθοδοι) παρουσιάζεται παρακάτω:

Τρίγωνο

```
ΣΥΝΑΡΤΗΣΗ
ΥπολογισμόςΕμβαδού():
ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ:Εμ
ΑΡΧΗ
Εμ <- Βάση*Υψος/2
ΥπολογισμόςΕμβαδού <- Εμ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παραλληλόγραμμο

```
ΣΥΝΑΡΤΗΣΗ
ΥπολογισμόςΕμβαδού():
ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ:Εμ
ΑΡΧΗ
Εμ <- Μήκος*Πλάτος
ΥπολογισμόςΕμβαδού<-Εμ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Κύκλος

```
ΣΥΝΑΡΤΗΣΗ
ΥπολογισμόςΕμβαδού():
ΠΡΑΓΜΑΤΙΚΗ
ΣΤΑΘΕΡΕΣ
    Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ:Εμ
ΑΡΧΗ
Εμ <- Π*Ακτίνα*Ακτίνα
ΥπολογισμόςΕμβαδού <- Εμ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

(σελ 106 – 107 Συμπληρωματικό Υλικό)



ΕΝΟΤΗΤΑ 5η • ΚΕΦΑΛΑΙΟ 6ο

Λάθη και εκσφαλμάτωση

ΕΝΟΤΗΤΑ 5η Συμπληρωματικό Υλικό – ΚΕΦΑΛΑΙΟ 6ο Βιβλίο Μαθητή

165 ★★★★

Ποιες είναι οι 3 κατηγορίες λαθών σε ένα πρόγραμμα; Πως δημιουργούνται τα λάθη σε κάθε μια από αυτές τις κατηγορίες;

Μπορούμε να διακρίνουμε τις εξής κατηγορίες λαθών:



1. **Λάθη κατά την Υλοποίηση** ή **Συντακτικά Λάθη**: Κάποιες φορές ένα πρόγραμμα δεν μπορεί να εκτελεστεί, επειδή **κατά τη μετάφραση** εντοπίζονται συντακτικά λάθη. Π.χ. δε γράψαμε σωστά μια δεσμευμένη λέξη, παραλείψαμε μια δεσμευμένη λέξη ή παραλείψαμε να δηλώσουμε μια μεταβλητή.



2. **Λάθη κατά την Εκτέλεση** ή Λάθη που οδηγούν σε **αντικανονικό τερματισμό** του προγράμματος: Ένα πρόγραμμα μπορεί να τερματίσει αντικανονικά λόγω διαφόρων λαθών. Για παράδειγμα, αν επιχειρήσουμε να **διαιρέσουμε με το μηδέν** ή αν κατά την ανάγνωση ενός ακεραίου αριθμού εισαχθεί ένα γράμμα.



3. **Λογικά λάθη** ή Λογικά Λάθη που παράγουν **λανθασμένα αποτελέσματα**: Ακόμη κι αν το πρόγραμμά μας δεν περιέχει συντακτικά λάθη και μπορεί να εκτελεστεί, πρέπει οπωσδήποτε να ελεγχθεί, ώστε να διαπιστώσουμε αν κατά την εκτέλεσή του εμφανίζονται λογικά λάθη. Τα λογικά λάθη έχουν ως συνέπεια το πρόγραμμα σε κάποιες περιπτώσεις να εξάγει λανθασμένα αποτελέσματα. Για να εντοπίσουμε τα λογικά λάθη μπορούμε να κάνουμε **δοκιμαστικές εκτελέσεις** του προγράμματός μας και να ελέγξουμε αν για συγκεκριμένες τιμές εισόδου, το πρόγραμμά μας εξάγει σωστά αποτελέσματα. Μερικές φορές το λογικό λάθος δεν υπάρχει στην εντολή που εμφανίζεται το λανθασμένο αποτέλεσμα, αλλά σε προηγούμενη εντολή.

(σελ 115 – 120 Συμπληρωματικό Υλικό)

166 ★★★★

Που εντοπίζονται τα λάθη και από ποιόν; Ποια είναι πιο δύσκολα να εντοπιστούν και πιο «επώδυνα» για τη λειτουργία του προγράμματος; Πότε δημιουργείται το εκτελέσιμο πρόγραμμα;



Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση που το αρχικό πρόγραμμα δεν περιέχει συντακτικά λάθη. Ο **μεταγλωττιστής** ή ο **διερμηνευτής** ανιχνεύει λοιπόν τα συντακτικά λάθη και εμφανίζει κατάλληλα **διαγνωστικά μηνύματα**.



Τα λογικά λάθη εμφανίζονται μόνο **στην εκτέλεση**, ενώ τα συντακτικά λάθη **στο στάδιο της μεταγλώττισης** από τον μεταγλωττιστή ή τον διερμηνευτή. Τα λογικά λάθη δεν προκαλούν διακοπή της εκτέλεσης του προγράμματος και έχουν ως συνέπεια το πρόγραμμα να εξάγει λανθασμένα αποτελέσματα για κάποιες τιμές εισόδου. Τα λογικά λάθη δεν εντοπίζονται από κάποιο εργαλείο αλλά διαπιστώνονται με τη διαδικασία ελέγχου και ανάλυσης των αποτελεσμάτων του προγράμματος. Αυτό γίνεται με τη διενέργεια **δοκιμαστικών εκτελέσεων** του προγράμματος και ελέγχοντας αν για συγκεκριμένες τιμές εισόδου το πρόγραμμα παράγει τα αναμενόμενα αποτελέσματα.



Τα λάθη που προκαλούνται κατά την εκτέλεση του προγράμματος (δηλαδή τα **λάθη χρόνου εκτέλεσης – run time errors**) είναι πιο «επώδυνα» γιατί εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης και προκαλούν τον **αντικανονικό τερματισμό** του προγράμματος και το «κρέμασμα» του συστήματος.

(σελ 121 – 122 Βιβλίο Μαθητή)



167 ★★★★
Τι ονομάζουμε Εκσφαλμάτωση; Είναι δύσκολη διαδικασία; Τι είδους λάθη εντοπίζουμε;



Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται **εκσφαλμάτωση (debugging)**. Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί **βαθιά γνώση της γλώσσας προγραμματισμού** και φυσικά αντίστοιχες ικανότητες από τον προγραμματιστή. Η εκσφαλμάτωση είναι ένα **πρόβλημα λογικής** και όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.




Κατά τη διαδικασία της εκσφαλμάτωσης δεν μας απασχολούν τα συντακτικά λάθη καθώς αυτά έχουν ήδη διορθωθεί σε προηγούμενο στάδιο. Τα λάθη που κυρίως μας απασχολούν στη φάση της εκσφαλμάτωσης είναι τα **λογικά λάθη** και τα λάθη που παρουσιάζονται **κατά το χρόνο εκτέλεσης** του προγράμματος. Η εκσφαλμάτωση τέτοιων λαθών μπορεί να γίνει μέσα από **εργαλεία εκσφαλμάτωσης** ή από **ειδικές εντολές ή συναρτήσεις** που προσφέρει το περιβάλλον προγραμματισμού.

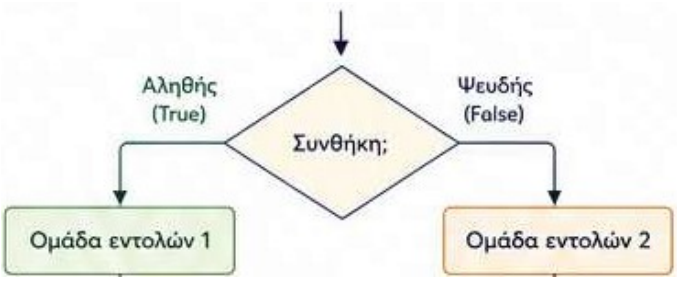
(σελ 235 Βιβλίο Μαθητή)

168 ★★☆☆

Ποια Λογικά λάθη μπορεί να εμφανιστούν σε μια δομή Επιλογής;

Σε μια δομή επιλογής μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

1. τη **συνθήκη ή τις συνθήκες**



2. τις **ομάδες εντολών** που εκτελούνται όταν μια συνθήκη είναι **αληθής ή ψευδής**.

Όπως είδαμε παραπάνω, μερικές φορές κατά την εκτέλεση της δομής επιλογής εμφανίζονται λανθασμένα αποτελέσματα, τα οποία σχετίζονται με λογικά λάθη σε προηγούμενες εντολές, που επηρεάζουν την τιμή που λαμβάνει η συνθήκη.

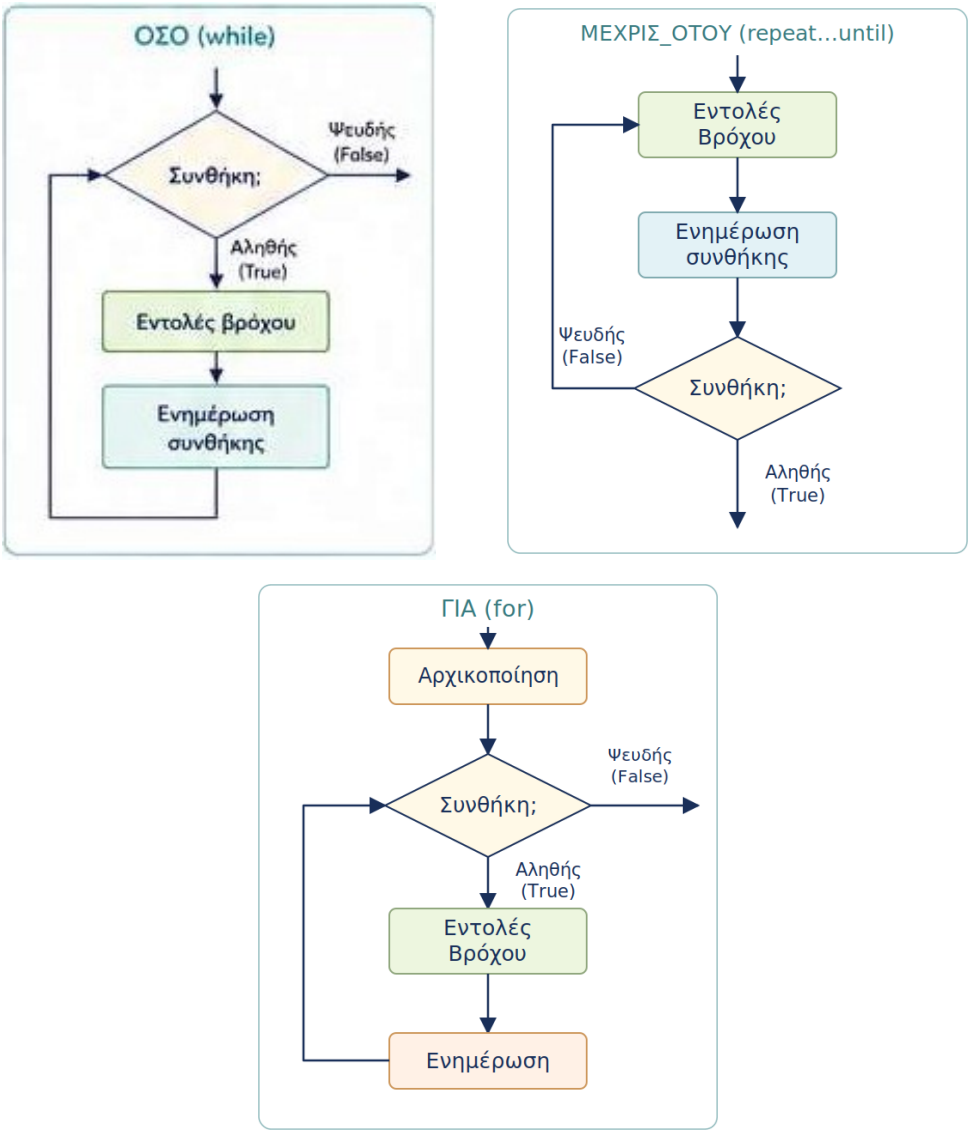
(σελ 121 Συμπληρωματικό Υλικό)

169 ★★☆☆

Ποια Λογικά λάθη μπορεί να εμφανιστούν σε μια δομή Επανάληψης;

Σε μια δομή επανάληψης μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

1. τη **συνθήκη επανάληψης ή τερματισμού**,
2. την **αρχικοποίηση της συνθήκης**,
3. την **ενημέρωση της συνθήκης** εντός του βρόχου επανάληψης,
4. τις **εντολές** που περιλαμβάνονται εντός του βρόχου.



(σελ 125 Συμπληρωματικό Υλικό)

170 ★★★☆

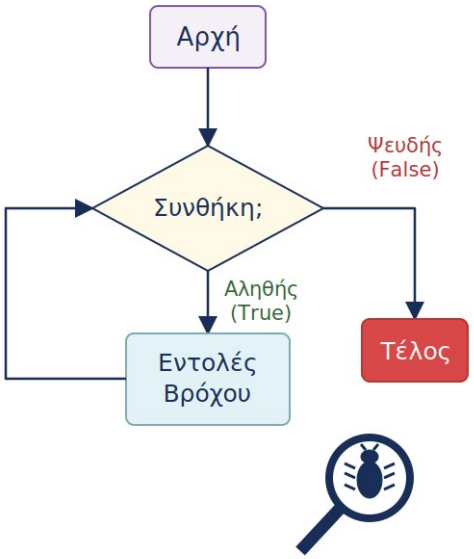
Που πρέπει να δίνεται προσοχή κατά την εκσφαλμάτωση των δομών επανάληψης;

Κατά την εκσφαλμάτωση των δομών επανάληψης χρειάζεται να δίνετε προσοχή στα εξής:

1. στους **συγκριτικούς και τους λογικούς τελεστές** των συνθηκών επανάληψης ή

τερματισμού

2. στην **αρχικοποίηση της συνθήκης**
3. στην **ενημέρωση της συνθήκης** εντός του βρόχου
4. στην **αλληλουχία των εντολών** του βρόχου και στη **σειρά εκτέλεσής τους**
5. στο **κριτήριο της περατότητας**
6. στην **πρώτη επανάληψη** και στην περίπτωση που ο βρόχος επανάληψης δεν πρέπει να εκτελεστεί ούτε μία φορά
7. στην **τελευταία επανάληψη**



(σελ 130 Συμπληρωματικό Υλικό)

171 ★☆☆☆

Που πρέπει να δίνεται προσοχή κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν Πίνακες;

Κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν πίνακες χρειάζεται να δίνετε ιδιαίτερη προσοχή:

1. στο **μέγεθος των πινάκων** κατά τη δήλωσή τους,
2. στους **δείκτες των πινάκων** κατά την προσπέλασή τους,

Πίνακας A[6]

1	2	3	4	5	6
A[1]	**A[2]**	**A[3]**	**A[4]**	**A[5]**	**A[6]**

3. στη μη υπέρβαση των ορίων του πίνακα.

(σελ 132 Συμπληρωματικό Υλικό)

172 ★☆☆☆

Που πρέπει να δίνεται προσοχή κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν Υποπρογράμματα;

Κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν υποπρογράμματα χρειάζεται να δίνεται προσοχή στον εντοπισμό λογικών λαθών που σχετίζονται με:

1. την **κλήση του υποπρογράμματος** και το πέρασμα των παραμέτρων



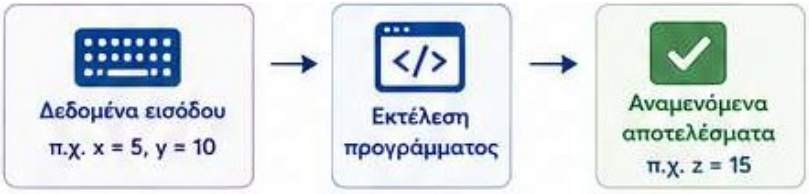
2. τα **λοιπά λογικά λάθη** που εμφανίζονται και στα προγράμματα.

(σελ 136 Συμπληρωματικό Υλικό)

173 ★★☆☆

Τι είναι τα «Σενάρια Ελέγχου»;

Ένα **σενάριο ελέγχου (test case)** περιγράφει τα **δεδομένα εισόδου** ολόκληρου του προγράμματος ή τμήματος του προγράμματος (διαδικασία, συνάρτηση) και τα **αναμενόμενα αποτελέσματα**. Τα σενάρια ελέγχου εκτελούνται, είτε σε πραγματικό περιβάλλον προγραμματισμού είτε εικονικά με δημιουργία πίνακα τιμών των μεταβλητών. Σε περίπτωση αποκλίσεων μεταξύ των αναμενόμενων και των πραγματικών αποτελεσμάτων, υπάρχει λάθος το οποίο πρέπει να εντοπιστεί και να διορθωθεί.



(σελ 138 Συμπληρωματικό Υλικό)

174 ★★☆☆
Τι είναι ο έλεγχος «Μαύρου Κουτιού»;



Μια δημοφιλής τεχνική ελέγχου είναι ο **έλεγχος μαύρου κουτιού (black-box testing)**. Ονομάζεται έτσι επειδή τα δεδομένα εισόδου στα σενάρια ελέγχου προκύπτουν από τις **προδιαγραφές του προγράμματος**, αγνοώντας εντελώς τον κώδικα. Δηλαδή το πρόγραμμα μοιάζει σαν να βρίσκεται μέσα σε ένα μαύρο κουτί που κρύβει το περιεχόμενό του.



(σελ 138 Συμπληρωματικό Υλικό)

175 ★★★★
Τι είναι τα Ισοδύναμα διαστήματα τιμών;



Ιδανικά θα θέλαμε να ελέγξουμε όλες τις τιμές εισόδου και όλα τα πιθανά αποτελέσματα. Αυτό όμως είναι αδύνατο. Γι' αυτό προσπαθούμε να βρούμε **αντιπροσωπευτικές τιμές** για τα δεδομένα εισόδου που θα παράγουν αντιπροσωπευτικά αποτελέσματα. Το πρώτο βήμα είναι η δημιουργία **ισοδύναμων διαστημάτων τιμών (equivalence partitioning)** για τα δεδομένα εισόδου. Τα διαστήματα θεωρούνται ισοδύναμα, καθώς αν δεν υπάρχουν λάθη, τότε όλες οι τιμές ενός διαστήματος εισόδου θα παράγουν τιμές που θα ανήκουν στο **ίδιο διάστημα αποτελεσμάτων** (Είναι σημαντικό να δημιουργούνται διαστήματα και για τις μη έγκυρες τιμές εισόδου, καθώς δεν μπορούμε να είμαστε σίγουροι ότι ένα πρόγραμμα θα τροφοδοτείται μόνο με έγκυρες τιμές).



(σελ 138 Συμπληρωματικό Υλικό)

176 ★★★☆
Τι είναι ο έλεγχος ακραίων τιμών κάθε διαστήματος;

Μετά τον καθορισμό των διαστημάτων πρέπει να επιλεγούν τιμές για τα σενάρια ελέγχου που να καλύπτουν **όλα τα διαστήματα**. Αφού τα διαστήματα είναι ισοδύναμα, μπορεί να επιλεγεί οποιαδήποτε τιμή από κάθε διάστημα. Μια καλύτερη στρατηγική είναι να γίνει έλεγχος των **ακραίων τιμών κάθε διαστήματος (boundary value analysis)**, καθώς η εμπειρία έχει δείξει ότι τα περισσότερα λάθη γίνονται σε αυτά τα σημεία. Αυτό είναι λογικό, αν σκεφτούμε ότι τα διαστήματα τιμών θα υλοποιηθούν με κάποια μορφή δομής επιλογής, οπότε μπορεί να υπάρχουν λάθη στις λογικές συνθήκες, π.χ. **συμπερίληψη ακραίας τιμής** (<= αντί για <, >= αντί για >), **παράλειψη ακραίας τιμής** (< αντί για <=, > αντί για >=).



(σελ 138 Συμπληρωματικό Υλικό)