ΠΛΗΡΟΦΟΡΙΚΗ – ΣΥΝΟΠΤΙΚΗ ΘΕΩΡΙΑ 2024-2025 (ΠΡΑΣΙΝΟ ΒΙΒΛΙΟ)

Κεφάλαιο 1: Ανάλυση προβλήματος

1.1 Η έννοια πρόβλημα

1) Να δώσετε τον ορισμό του προβλήματος**(ορισμός)**.
Με τον όρο πρόβλημα εννοείται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι ούτε γνωστή, ούτε προφανής.

2) Περιγράψτε το πρόβλημα του έτους 2000 (millennium bug).
Οι υπολογιστές για εξοικονόμηση αποθηκευτικού χώρου, αποθήκευαν τις ημερομηνίες με 2 ψηφία για κάθε ένα από τα τρία συστατικά της. Για παράδειγμα η ημερομηνία 15 Απριλίου 1999 αναπαριστάται με 150499. Με τον παραπάνω τρόπο αποθήκευσης, η ημερομηνία 1 Ιανουαρίου 2000 θα είχε την μορφή 010100, γεγονός που θα αποτελούσε πρόβλημα στους υπολογισμούς που θα πραγματοποιούσαν οι υπολογιστές.

1.2 Κατανόηση προβλήματος.

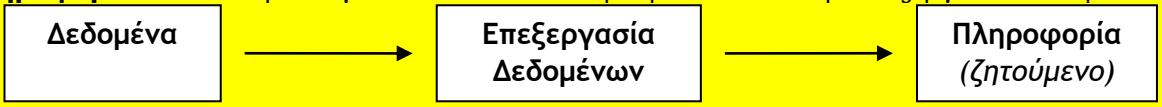
1) Από ποιους παράγοντες επηρεάζεται η κατανόηση ενός προβλήματος;
Η κατανόηση ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων : της σωστής διατύπωσης εκ μέρους του δημιουργού του και της αντίστοιχα σωστής ερμηνείας από τη μεριά εκείνου που καλείται να το αντιμετωπίσει.

2) Ποια είναι η μορφή με την οποία πρέπει να παρουσιάζεται ένα πρόβλημα;
Η μορφή με την οποία παρουσιάζεται ένα πρόβλημα μπορεί να είναι οποιαδήποτε αρκεί να μπορεί να γίνει αντιληπτή από μία από τις πέντε ανθρώπινες αισθήσεις.

3) Τι γνωρίζετε για την σαφήνεια της διατύπωσης ενός προβλήματος;
Τα προβλήματα που μπορεί να κληθούμε να αντιμετωπίσουμε κατά τη διάρκεια της ζωής μας μπορούν να αναφέρονται σε οποιοδήποτε τομέα, μπορεί να αφορούν στα μαθηματικά, στη φυσική, στη λογική, στην καθημερινή ζωή ή οτιδήποτε άλλο θα μπορούσε κάποιος να σκεφτεί. Η κατανόηση ενός προβλήματος εξαρτάται σε μεγάλο βαθμό από την διατύπωσή του. Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός, είτε ο γραπτός. Ο λόγος σαν μέσο επικοινωνίας και συνεννόησης πρέπει να χαρακτηρίζεται από σαφήνεια. Άστοχη χρήση ορολογίας, λανθασμένη σύνταξη, είναι δύο στοιχεία που μπορούν να προκαλέσουν παρερμηνείες και παραπλανήσεις.

4) Τι γνωρίζετε για τον χώρο στον οποίο αναφέρεται ένα πρόβλημα;
Σημαντικός παράγοντας για την κατανόηση ενός προβλήματος είναι η αποσαφήνιση του χώρου στον οποίο αναφέρεται. Η πληροφορία αυτή παρέχεται από την διατύπωση-δεδομένα του προβλήματος. Παράδειγμα στον χώρο: Έγραψα στο διαγώνισμα 7,5. Είναι καλός ή κακός βαθμός; Η απάντηση εξαρτάται από τον χώρο: Αν μιλάμε για πανεπιστήμιο είναι καλός!

5) Να δώσετε τους **ορισμούς** των εννοιών: **δεδομένο, πληροφορία, επεξεργασία δεδομένων** και να δώσετε μία διαγραμματική σχέση μεταξύ τους.
Δεδομένο: Οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις του.
Επεξεργασία δεδομένων: Μία διαδικασία κατά την οποία ένας μηχανισμός (π.χ. ο Η/Υ, η ο ανθρώπινος εγκέφαλος) δέχεται δεδομένα, κάνει πράξεις και υπολογισμούς σύμφωνα με ένα προκαθορισμένο τρόπο και αποδίδει πληροφορίες. Για παράδειγμα: η μέση θερμοκρασία κατά τον μήνα Μάρτιο (πληροφορία), προκύπτει από τον υπολογισμό του μέσου όρου (επεξεργασία) των ημερησίων θερμοκρασιών του μήνα (δεδομένα).
Πληροφορία: Οποιοδήποτε γνωσιακό στοιχείο προέρχεται από την επεξεργασία δεδομένων.



1.3 Δομή προβλήματος.

1) Τι ονομάζεται δομή ενός προβλήματος **(ορισμός)**;
Με τον όρο **δομή προβλήματος**, εννοούμε τα συστατικά μέρη από τα οποία συντίθεται το πρόβλημα, δηλαδή τα επιμέρους τμήματα που το αποτελούν καθώς και τον τρόπο με τον οποίο αυτά συνδέονται μεταξύ τους. Χωρίζοντας το πρόβλημα σε μικρότερα και απλούστερα υποπροβλήματα, αυτά λύνονται ευκολότερα, καταλήγοντας στη λύση του αρχικού προβλήματος.

2) Τι γνωρίζετε για την διαγραμματική αναπαράσταση ενός προβλήματος;
- ✓ Το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο.
- ✓ Κάθε ένα από τα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο.
- ✓ Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, σχηματίζοντας ένα επίπεδο χαμηλότερα. Έτσι σε κάθε κατώτερο επίπεδο,

δημιουργείται η γραφική αναπαράσταση των προβλημάτων στα οποία αναλύονται τα προβλήματα του αμέσως ψηλότερου επιπέδου.

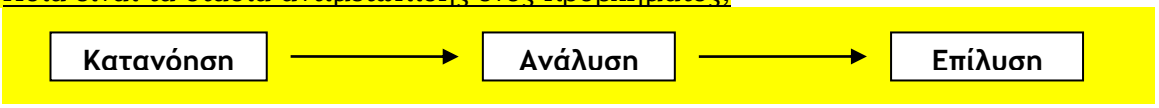
1.4 Καθορισμός απαιτήσεων.

1) Τι είναι ο καθορισμός απαιτήσεων ενός προβλήματος;
Για τη σωστή επίλυση του προβλήματος βασική προϋπόθεση είναι ο **καθορισμός απαιτήσεων**. Είναι η διαδικασία κατά την οποία πρέπει να κάνουμε:

- ✓ τον επακριβή προσδιορισμό των δεδομένων που παρέχει το πρόβλημα καθώς και
- ✓ την λεπτομερειακή καταγραφή των ζητούμενων που αναμένονται σαν αποτελέσματα της επίλυσης,

Τα δεδομένα δεν είναι πάντα εύκολο να διακριθούν.(Σε πολλά προβλήματα τα δεδομένα θα πρέπει να «ανακαλυφθούν» μέσα στα λεγόμενα)

2) Ποια είναι τα στάδια αντιμετώπισης ενός προβλήματος;



- ✓ κατανόηση, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος
- ✓ ανάλυση, όπου το αρχικό πρόβλημα διασπάται σε άλλα επί μέρους απλούστερα προβλήματα
- ✓ επίλυση, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.

Κεφάλαιο 2 : Βασικές Έννοιες Αλγορίθμων

2.1 Τι είναι αλγόριθμος.

1) Να δοθεί ο **ορισμός** του Αλγόριθμου.
Είναι μία πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.
Η σειρά – αλληλουχία των ενεργειών δεν είναι μοναδική. Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά με έννοιες της πληροφορικής.

2) Ποια είναι τα κριτήρια που πρέπει να πληρεί ένας αλγόριθμος;

- **Είσοδος:** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδος. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται όταν: ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με την βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με την βοήθεια άλλων απλών εντολών.
- **Έξοδος:** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον ένα αποτέλεσμα προς τον χρήστη ή προς ένα άλλο αλγόριθμο.
- **Καθοριστικότητα**: Κάθε εντολή πρέπει να καθορίζεται χωρίς αμφιβολία για τον τρόπο εκτέλεσής της. Πχ μία εντολή ζ ← x / ψ πρέπει να λαμβάνει υπ' όψιν της το γεγονός ότι μπορεί το ψ να είναι μηδέν.
- **Περατότητα:** Ο αλγόριθμος πρέπει να τελειώνει μετά από πεπερασμένα βήματα. Αν δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν είναι αλγόριθμος αλλά υπολογιστική διαδικασία.
- **Αποτελεσματικότητα:** Κάθε εντολή ενός αλγορίθμου πρέπει να είναι απλή και εκτελέσιμη (δεν αρκεί να έχει οριστεί).

2.2 Σπουδαιότητα αλγορίθμων.

1) Ποιες είναι οι σκοπιές από τις οποίες μελετάει τους αλγορίθμους η Πληροφορική;

- **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 κοκ.)
- **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.
- **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.

- **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.

2.3 Περιγραφή και αναπαράσταση αλγορίθμων

1) Με ποιους τρόπους μπορούμε να αναπαραστήσουμε έναν αλγόριθμο;

- **Ελεύθερο κείμενο** (Free text): Αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης. Υπάρχει κίνδυνος να οδηγήσει σε μη εκτελέσιμη παρουσίαση, παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων το κριτήριο της αποτελεσματικότητας.
- **Διαγραμματικές τεχνικές** (diagramming techniques): Αποτελούν ένα γραφικό τρόπο αναπαράστασης. Μια τεχνική (από τις πιο παλιές και γνωστές) είναι το διάγραμμα ροής (Flow Chart). Η χρήση διαγρ. τεχνικών δεν είναι η καλύτερη λύση, γι' αυτό εμφανίζονται όλο και σπανιότερα στην βιβλιογραφία και στην πράξη.
- **Φυσική γλώσσα κατά βήματα** (natural language): Μοιάζει με το Ελεύθερο κείμενο απλά είναι κατά βήματα. Κίνδυνος να παραβιαστεί το κριτήριο της καθοριστικότητας.
- **Κωδικοποίηση** (coding): Δηλαδή με ένα πρόγραμμα γραμμένο είτε με μία ψευδογλώσσα είτε σε κάποιο προγραμματιστικό περιβάλλον που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

2) Ποια είναι τα σύμβολα που χρησιμοποιούνται στο διάγραμμα ροής;

Αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα σχήματα ενώνονται μεταξύ τους με βέλη που δηλώνουν την σειρά εκτέλεσης των ενεργειών αυτών.

Τα κυριότερα σχήματα είναι τα εξής:

- **Έλλειψη**: Η αρχή και το τέλος του αλγορίθμου.
- **Ρόμβος**: Μία συνθήκη με δύο εξόδους ανάλογα με την τιμή της. {αληθής - ψευδής}
- **Ορθογώνιο**: Η εκτέλεση μίας ή περισσοτέρων πράξεων.
- **Πλάγιο παρ/μο**: Η είσοδος και η έξοδος στοιχείων του αλγορίθμου.

2.4 Βασικές συνιστώσες / εντολές ενός αλγορίθμου.

Ένας αλγόριθμος διαμορφώνεται από τα παρακάτω :

1. **Τελεσταίοι**
2. **Τελεστές**
3. **Εκφράσεις**
4. **Εντολές**

{Αναλυτικότερα:}

Τελεσταίοι (operands)

•**Σταθερές(constants)**: Προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου.

•**Μεταβλητές(variables)**: Είναι αντικείμενα {δηλαδή λέξεις..} που χρησιμοποιούνται για να παραστήσουν στοιχεία δεδομένων. Η διαφορά τους από τις σταθερές είναι ότι δέχονται μία τιμή που μπορεί να αλλάξει κατά την εκτέλεση ενός αλγορίθμου. Οι τελεσταίοι (μεταβλητές-σταθερές) διακρίνονται σε τρεις κατηγορίες ανάλογα με το είδος των τιμών που μπορούν να λάβουν.

{βλέπε για αντιπαραβολή το κεφάλαιο 7 όπου οι κατηγορίες είναι τέσσερις}

i. Αριθμητικοί:

Είναι ακέραιες τιμές ή πραγματικές

πχ 12, 3.14, -19.99 κλπ.

ii. Αλφαριθμητικοί:

Αποτελούνται από σειρές χαρακτήρων μέσα σε εισαγωγικά. Οι χαρακτήρες μπορεί να είναι γράμματα, ψηφία, σημεία στίξης κλπ.

πχ "γειά σας", "μεσος1", "12", κλπ

iii. Λογικοί: Η τιμές που μπορούν να πάρουν είναι ακριβώς δύο : αληθής, ψευδής.

Τελεστές (operators)

Είναι σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Διακρίνονται σε τρεις κατηγορίες:

Αριθμητικοί:	Λογικοί:	Συγκριτικοί:
^	Οχι (άρνησης)	= (ίσο)
* / div mod	Και (σύζευξης)	<> (διάφορο)
+ -	Η (διάξευξης)	>= (μεγαλύτερο ή ίσο) κλπ

Εκφράσεις (expressions)

Όταν μία τιμή προκύπτει από υπολογισμό τότε αναφερόμαστε σε εκφράσεις. Οι τελεστές μαζί με τους τελεσταίους διαμορφώνουν τις εκφράσεις. Η διεργασία αποτίμησης μίας έκφρασης συνίσταται στην απόδοση

τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Η εκτέλεση των πράξεων εξαρτάται από την ιεραρχία των πράξεων και την χρήση παρενθέσεων. Μία έκφραση μπορεί να αποτελείται από μία μόνο μεταβλητή ή σταθερά μέχρι μία πολύπλοκη μαθηματική παράσταση.
Χωρίζονται σε δύο κατηγορίες : τις αριθμητικές (Σχολ. Κεφ. 7) και τις λογικές (Σχολ. Κεφ. 8)
SOS: Οι λογικές εκφράσεις ονομάζονται αλλιώς και: Συνθήκες

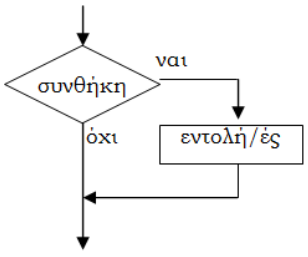
Παραδείγματα εκφράσεων

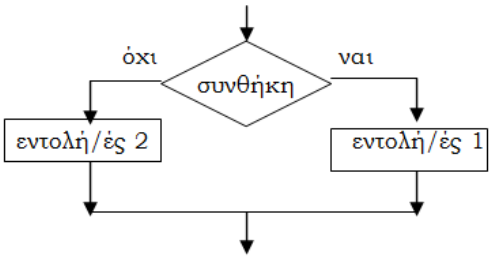
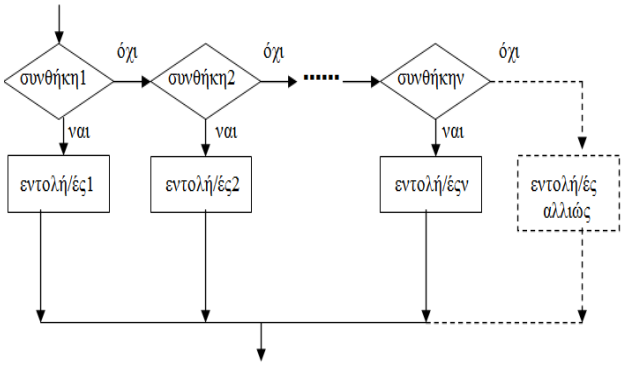
Αριθμητικές :	*Λογικές*
X	*X+2>0*
X+2	*X+2>0 και X<> 8*
X+C/2^3	*Αληθής και όχι (X=3)*

Εντολές
Αποκαλείται κάθε μία λέξη που προσδιορίζει μία σαφή ενέργεια.
Οι εντολές μπορούν να διακριθούν στις τρεις βασικές δομές του δομημένου προγραμματισμού:
•Δομή ακολουθίας
•Δομή επιλογής
•Δομή επανάληψης
Οι εντολές διακρίνονται επίσης σε **δηλωτικές** και σε **εκτελέσιμες**.
Πχ

Δηλωτικές	Εκτελέσιμες
Αλγόριθμος	Διάβασε χ

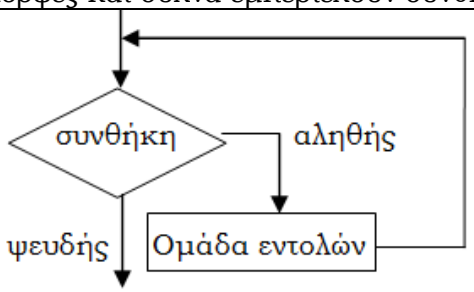
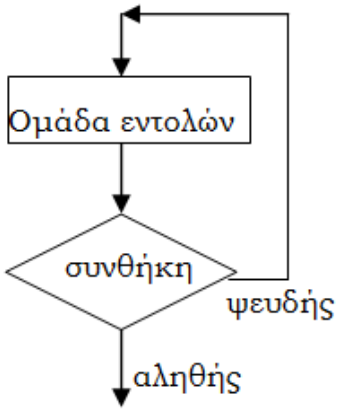
Άλλες δηλωτικές εντολές:
// Δεδομένα // : Τα δεδομένα εισόδου αν υπάρχουν περιγράφονται στην δεύτερη γραμμή του αλγορίθμου (μετά την εντολή αλγόριθμος)
// Αποτελέσματα // : Τα αποτελέσματα εξόδου δίνονται στην προτελευταία γραμμή (πριν την εντολή τέλος)
Δομή ακολουθίας.
Ονομάζεται και σειριακή ή ακολουθιακή δομή. Αποτελείται από ένα σύνολο εντολών που τοποθετούνται η μία κάτω από την άλλη. Χρησιμοποιείται (από μόνη της) για την επίλυση πολύ απλών προβλημάτων όπου η σειρά εκτέλεσης ενός συνόλου ενεργειών είναι δεδομένη. Χρησιμοποιείται ευρύτατα σε συνδυασμό με άλλες δομές (επιλογής, επανάληψης). Στη δομή αυτή ανήκουν οι εντολές:
•**Εισόδου:** Διάβασε. Η εντολή αυτή συνοδεύεται με το όνομα μίας ή περισσοτέρων μεταβλητών. Η λειτουργία της είναι: μετά την ολοκλήρωσή της η μεταβλητή/μεταβλητές θα έχει λάβει τιμή ως περιεχόμενο.
Πχ : διάβασε x,ψ
•**Εξόδου:** Εμφάνισε – Εκτύπωσε. Οι εντολές αυτές εμφανίζουν τα αποτελέσματα στην οθόνη και στον εκτυπωτή αντίστοιχα. Η σύνταξή της εντολής αυτής είναι ανάλογη με του διάβασε.
•**Εκχώρησης τιμής**: Η γενική μορφή της εντολής είναι : Μεταβλητή←έκφραση. Η λειτουργία της είναι: γίνονται οι πράξεις στην έκφραση και το αποτέλεσμά της αποδίδεται , μεταβιβάζεται, εκχωρείται στη μεταβλητή. Ας σημειωθεί ότι δεν πρόκειται για εξίσωση και ότι οι διάφορες γλώσσες προγραμματισμού χρησιμοποιούν διάφορα σύμβολα αντί για το βέλος.
•**Δηλωτικές** : Αλγόριθμος , Τέλος. Ένας αλγόριθμος διατυπωμένος σε ψευδογλώσσα αρχίζει πάντα με τη λέξη Αλγόριθμος συνοδευόμενη με το όνομα του και τελειώνει με την λέξη Τέλος συνοδευόμενη επίσης με το όνομά του.
Δομή επιλογής
Αποτελείται από ένα σύνολο εντολών που εκτελούνται κατά περίπτωση. Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης με δύο δυνατές τιμές (αληθής, ψευδής) και στη συνέχεια την απόφαση εκτέλεσης κάποιας εντολής ανάλογα με τη συνθήκη.

Απλή επιλογή	**ΑΝ** συνθήκη **ΤΟΤΕ** εντολή/ές **ΤΕΛΟΣ_ΑΝ**	

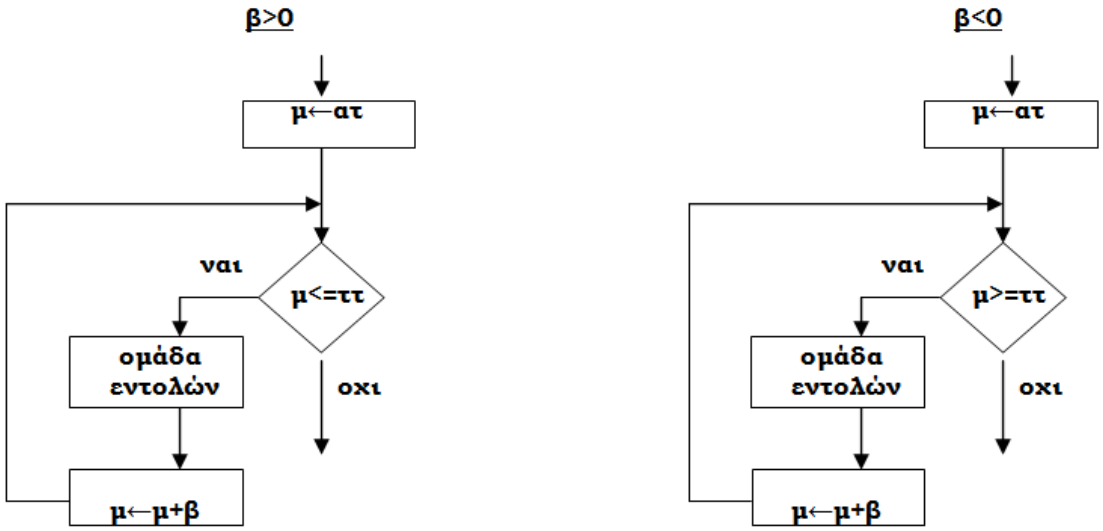
Σύνθετη επιλογή	**ΑΝ** συνθήκη **ΤΟΤΕ** εντολή/ές 1 **ΑΛΛΙΩΣ** εντολή/ές 2 **ΤΕΛΟΣ_ΑΝ**	
Πολλαπλή επιλογή	**ΑΝ** συνθήκη1 **ΤΟΤΕ** εντολή/ές1 **ΑΛΛΙΩΣ_ΑΝ** συνθήκη2 **ΤΟΤΕ** εντολή/ές2 . . . **ΑΛΛΙΩΣ_ΑΝ** συνθήκην **ΤΟΤΕ** εντολή/ές (**ΑΛΛΙΩΣ** εντολή/ές αλλιώς) **ΤΕΛΟΣ_ΑΝ**	

Δομή επανάληψης

Η δομή επανάληψης εφαρμόζεται στις περιπτώσεις όπου μία ακολουθία εντολών πρέπει να γίνει περισσότερες από μία φορές. Εφαρμόζεται σε ένα σύνολο περιπτώσεων που έχουν κάτι κοινό. Οι επαναληπτικές διαδικασίες έχουν τρεις μορφές και συχνά εμπεριέχουν συνθήκες επιλογών.

ΟΣΟ συνθήκη **ΕΠΑΝΑΛΑΒΕ** Ομάδα εντολών **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	
ΑΡΧΗ _ΕΠΑΝΑΛΗΨΗΣ Ομάδα εντολών **ΜΕΧΡΙΣ_ΟΤΟΥ** συνθήκη	

ΓΙΑ μ **ΑΠΟ** ατ **ΜΕΧΡΙ** ττ **ΜΕ_ΒΗΜΑ** β
Ομάδα εντολών
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ → μπορεί να παραληφθεί όταν β=1



Ολίσθηση

Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή , δηλαδή 0 και 1. Έτσι ο αριθμός 5 του δεκαδικού συστήματος αντιστοιχεί στον αριθμό 00000101 του δυαδικού συστήματος (με χρήση 8 bits). Η ολίσθηση διακρίνεται σε ολίσθηση προς τα αριστερά και προς τα δεξιά.

- Προς τα αριστερά : Ο αριθμός 00000101 γράφεται 00001010 (μετακινείται προς τα αριστερά).

Ο ισοδύναμος αριθμός στο δεκαδικό είναι ο 10 (δέκα). Άρα ολίσθηση προς τα αριστερά ισοδυναμεί με διπλασιασμό του αριθμού.

Πχ προς τα αριστερά :

5:	0	0	0	0	0	1	0	1
10:	0	0	0	0	1	0	1	0

- Προς τα δεξιά: Ο αριθμός 00000101 γράφεται 00000010 (μετακινείται προς τα δεξιά). Ο ισοδύναμος αριθμός στο δεκαδικό είναι ο 2 (δύο) Άρα ολίσθηση προς τα δεξιά ισοδυναμεί με υποδιπλασιασμό του αριθμού.

Πχ προς τα δεξιά:

5:	0	0	0	0	0	1	0	1
2:	0	0	0	0	1	0	1	0

Πολλαπλασιασμός αλά ρωσικά

Η πράξη του πολλαπλασιασμού δύο αριθμών δεν εκτελείται από τον υπολογιστή με τον τρόπο που την εκτελούμε εμείς. Έστω ότι θέλω να πολλαπλασιάσω τους αριθμούς 45 και 19 Οι αριθμοί που πρέπει να πολλαπλασιαστούν γράφονται δίπλα δίπλα και ο πρώτος διπλασιάζεται ενώ ο δεύτερος υποδιπλασιάζεται. Η διαδικασία αυτή επαναλαμβάνεται ώσπου ο δεύτερος να γίνει μονάδα :

45	**19**	*45*
90	**9**	*90*
180	4	
360	2	
720	**1**	*720*

Αλγόριθμος Πολλσμός_αλά_ρωσικά
Δεδομένα //M1,M2 ακέραιοι //
P ← 0
Όσο M2>0 **επανάλαβε**
Αν (M2 MOD 2) = 1 **τότε** P ← P + M1
M1 ← M1 * 2
M2 ← M2 DIV 2
Τέλος_επανάληψης
Αποτελέσματα // P //
Τέλος Πολλσμός_αλά_ρωσικά

Τελικά το ζητούμενο γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης όπου αντίστοιχα στην δεύτερη στήλη υπάρχει περιττός αριθμός. Στο παράδειγμα τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη.

Δηλαδή είναι 45+90+720=855

Κεφάλαιο 3 : Δομές Δεδομένων και Αλγόριθμοι

3.1 Δεδομένα.

Τα δεδομένα (data) είναι η αφαιρετική αναπαράσταση της πραγματικότητας. Είναι ακατέργαστα γεγονότα και κάθε φορά η επιλογή τους εξαρτάται από την φύση του προβλήματος.

Πληροφορία

Η συλλογή ακατέργαστων γεγονότων και ο συσχετισμός τους δίνει ως αποτέλεσμα την πληροφορία (information). Δεν είναι εύκολο να δοθεί ακριβής ορισμός για την πληροφορία αλλά μπορεί να θεωρηθεί ότι ο αλγόριθμος είναι το μέσο για την παραγωγή της πληροφορίας από τα δεδομένα. Με βάση τις δεδομένες πληροφορίες λαμβάνονται διάφορες αποφάσεις και γίνονται ενέργειες. Στην συνέχεια οι ενέργειες παράγουν νέα δεδομένα , νέες πληροφορίες , νέες αποφάσεις , νέες ενέργειες κοκ.

Θεωρία πληροφοριών

Είναι ιδιαίτερος κλάδος της πληροφορικής ο οποίος μελετά την
•μέτρηση,
•κωδικοποίηση,
•μετάδοση της πληροφορίας

1) Από ποιες σκοπιές μελετά τα δεδομένα η πληροφορική;

- **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.
- **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.
- **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 κλπ.
- **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 της Αναπαράστασης Γνώσης (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

3.2 Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα.

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στην δευτερεύουσα.

Η αποθήκευση δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή.

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).

{ένας πίνακας 100 θέσεων έχει 100 κόμβους }

1) Να δοθεί ο **ορισμός** της δομής δεδομένων.

Είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

2) Ποιες είναι οι βασικές λειτουργίες επί των δομών δεδομένων;

- **Προσπέλαση:** πρόσβαση σε ένα κόμβο με σκοπό να εξεταστεί ή να αλλάξει το περιεχόμενό του.
- **Εισαγωγή:** προσθήκη νέων κόμβων σε μία δομή.
- **Διαγραφή:** (αντίθετο της εισαγωγής), δηλαδή αφαίρεση ενός κόμβου από μία δομή.

(όχι σε πίνακες (στατικές δομές))

- **Αναζήτηση:** γίνεται προσπέλαση των κόμβων μίας δομής προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μία δεδομένη ιδιότητα.
- **Ταξινόμηση:** όπου οι κόμβοι διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
- **Αντιγραφή:** όλοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη.
- **Συγχώνευση:** δύο ή περισσότερες δομές ενώνονται σε μία δομή.
- **Διαχωρισμός:** (αντίστροφα από την συγχώνευση).

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή. Συνήθως μία δομή είναι αποδοτικότερη από μία άλλη σε κάποια λειτουργία. Άλλη είναι πιο αποδοτική στην αναζήτηση άλλη στην εισαγωγή κλπ. Ανάλογα με την λειτουργία την οποία θέλουμε να κάνουμε επιλέγουμε και την κατάλληλη δομή.

Αλγόριθμοι + Δομές δεδομένων = Προγράμματα

Υπάρχει μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται αυτήν την δομή.Ενα πρόγραμμα πρέπει να θεωρεί τον αλγόριθμο καθώς και την δομή που αυτός επεξεργάζεται ως μια αδιάσπαστη ενότητα.

3.3 Πίνακες

1) Ποιες είναι οι κατηγορίες δομών δεδομένων;

Δυναμικές δομές:

•Οι δομές αυτές δεν έχουν σταθερό μέγεθος αλλά ο αριθμός των κόμβων τους μεγαλώνει καθώς εισάγονται νέα δεδομένα , και μικραίνει καθώς διαγράφονται.

•Δεν αποθηκεύονται απαραίτητα σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της δυναμικής παραχώρησης μνήμης .

•Όλες οι σύγχρονες γλώσσες προγραμματισμού τις υποστηρίζουν.

Στατικές δομές:

• Οι δομές αυτές έχουν σταθερό μέγεθος και το ακριβές μέγεθός τους καθορίζεται κατά την στιγμή της μετάφρασης και όχι κατά την στιγμή της εκτέλεσης του προγράμματος.

•Οι κόμβοι τους αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

•Στην πράξη οι στατικές δομές υλοποιούνται με πίνακες

Πίνακες

•(ορισμός). Είναι μία δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους , πραγματικούς).

•Η αναφορά στα στοιχεία του γίνεται με την χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσοτέρων δεικτών. *{ΠΧ A[3,4]}* Ο τρόπος δήλωσης του πίνακα και η μέθοδος αναφοράς του εξαρτάται από την γλώσσα προγραμματισμού που χρησιμοποιείται. *{ΠΧ στην γλώσσα Java A[3][4]}*

•Ένας πίνακας μπορεί να είναι μονοδιάστατος , δισδιάστατος , τρισδιάστατος και γενικά ν-διάστατος. Ειδικά για τους δισδιάστατους πίνακες αν το μέγεθος των δύο διαστάσεων είναι ίδιο τότε ο πίνακας ονομάζεται τετραγωνικός. Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών της στοίβας και της ουράς.

3.5.Αναζήτηση

•Το πρόβλημα της αναζήτησης έχει ιδιαίτερο ενδιαφέρον λόγω της χρησιμότητάς του σε πλήθος εφαρμογών *{ΠΧ αναζήτηση του καταλόγου του ΟΤΕ για κάποιο συγκεκριμένο όνομα}*

•Υπάρχουν αρκετοί αλγόριθμοι (μέθοδοι) αναζήτησης σε πίνακα. Η επιλογή του πιο κατάλληλου εξαρτάται από δύο παράγοντες:

α.Αν ο πίνακας είναι ταξινομημένος ή όχι.

β.Αν όλα τα στοιχεία του πίνακα είναι διαφορετικά μεταξύ τους ή όχι.

•Τα στοιχεία του πίνακα μπορεί να είναι οποιουδήποτε τύπου: αριθμητικά (ακέραιοι-πραγματικοί), αλφαριθμητικά (χαρακτήρες) ακόμη και λογικά.

Σειριακή (sequential) – γραμμική (linear) αναζήτηση

•Είναι η πιο απλή αλλά και η πιο αναποτελεσματική μέθοδος αναζήτησης σε πίνακα.

•Η χρήση της δικαιολογείται σε περιπτώσεις όπου ο πίνακας:

είναι μη ταξινομημένος,
είναι μικρού μεγέθους (ν<=20),
η αναζήτηση γίνεται σπάνια.

```
Αλγόριθμος Sequential_Search
Δεδομένα // n, table, key //
done ← ψευδής
position ← 0
i ← 1
Όσο (done=ψευδής) και (i<=n) επανάλαβε
   Αν table[i]=key τότε
   done ← αληθής
   position ← i
   αλλιώς
   i ← i+1
  Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα //done, position //
Τέλος Sequential_Search
```

3.7.Ταξινόμηση

•(Ορισμός): Δοθέντων των στοιχείων α1, α2, … αn η ταξινόμηση ορίζεται ως μετάθεση της θέσης των στοιχείων ώστε να τοποθετηθούν σε μια σειρά ακ1, ακ2, … ακn έτσι ώστε: Αν δοθεί συνάρτηση διάταξης (ordering function) F να ισχύει : F(ακ1) ≤ F(ακ2) ≤ … ≤ F(ακn).

Η παραπάνω συνάρτηση διάταξης μπορεί να τροποποιηθεί ώστε να καλύπτει και την φθίνουσα ταξινόμηση.
{Αν η συνάρτηση διάταξης είναι η F(x)= x τότε η παραπάνω διάταξη είναι αύξουσα. Αν η συνάρτηση διάταξης είναι η F(x)=-x τότε η παραπάνω διάταξη είναι φθίνουσα.}

Ταξινόμηση ευθείας ανταλλαγής-φυσαλίδας

•Η μέθοδος της φυσαλίδας βασίζεται στην σύγκριση και ανταλλαγή ζευγών γειτονικών στοιχείων μέχρις ότου διαταχθούν όλα τα στοιχεία.

•Αν θεωρήσουμε ένα πίνακα μονοδιάστατο σε κατακόρυφη θέση, σύμφωνα με την μέθοδο αυτή γίνονται διαδοχικές προσπελάσεις στον πίνακα και μετακινείται το μικρότερο στοιχείο του πίνακα προς τα άνω (αύξουσα ταξινόμηση).

•Η ταξινόμηση φυσαλίδας είναι η πιο απλή και πιο αργή μέθοδος ταξινόμησης.

```
Δεδομένα // table, n //
Για i από 2 μέχρι n
   Για j από n μέχρι i με_βήμα -1
      Αν table[j-1] > table[j] τότε
         αντιμετάθεσε table[j—1], table[j]
     Τέλος_αν
  Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // table //
Τέλος Φυσαλίδα
```

Δομές δεδομένων δευτερεύουσας μνήμης.

Για την αποθήκευση δεδομένων στην **περιφερειακή ή δευτερεύουσα μνήμη** κάνουμε χρήση ειδικών δομών που λέγονται ***αρχεία (files).***

Κάθε αρχείο περιέχει μία συλλογή εγγραφών (π.χ. ένα αρχείο με εγγραφές μαθητών).

Μία ***εγγραφή (Record)*** αποτελεί τη συλλογή στοιχειωδών πληροφοριών σχετικά με ένα αντικείμενο (π.χ. μαθητής, ένα βιβλίο κλπ).

Μία στοιχειώδης πληροφορία συνιστά ένα ***πεδίο (field).*** Π.χ. μία εγγραφή μαθητή περιέχει στοιχειώδεις πληροφορίες (πεδία) όπως : Επώνυμο, Όνομα, Πατρώνυμο, Έτος γέννησης, Τάξη κλπ.

Το πεδίο που ταυτοποιεί μία εγγραφή (την κάνει μοναδική) ονομάζεται *πρωτεύον κλειδί*, π.χ. ο αριθμός μητρώου ενός μαθητή, ενώ το Ονοματεπώνυμο *δευτερεύον κλειδί* (αν υπήρχε το πρωτεύον κλειδί).

Κεφάλαιο 4. Τεχνικές Σχεδίασης Αλγορίθμων

4.1 Ανάλυση προβλημάτων.

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

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 και στη συνέχεια: **4)** την πρόταση επίλυσης με χρήση κάποιας μεθόδου **5)** την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

2) Σε ποιες ερωτήσεις θα πρέπει να απαντήσουμε 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).

Δεν υπάρχει ένας ενιαίος κανόνας, μία γενική φόρμουλα που να αναφέρεται στην επίλυση του συνόλου των προβλημάτων. Υπάρχουν όμως "συγγενή" προβλήματα, δηλαδή προβλήματα που μπορούν να αναλυθούν με παρόμοιο τρόπο και να αντιμετωπισθούν με αντίστοιχες μεθόδους και τεχνικές.

3) Για ποιο λόγο παρουσιάζουν ιδιαίτερο ενδιαφέρον οι μέθοδοι ανάλυσης και επίλυσης προβλημάτων;

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.

Κεφάλαιο 6. Εισαγωγή στον Προγραμματισμό

6.1 Η έννοια του προγράμματος

1. Ποια είναι τα στάδια επίλυσης ενός προβλήματος με τον υπολογιστή;

1) τον ακριβή προσδιορισμό του προβλήματος **2)** την ανάπτυξη του αντίστοιχου αλγορίθμου **3)** η διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή.

2. Με ποιο από τα παραπάνω στάδια ασχολείται ο προγραμματισμός;

Ο προγραμματισμός ασχολείται με το τρίτο αυτό στάδιο, τη δημιουργία του προγράμματος δηλαδή του συνόλου των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος. 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.

3. Ποιες είναι οι καταστάσεις που καταλαβαίνει ένας υπολογιστής και ποιες είναι οι στοιχειώσεις ενέργειες που 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 μα 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.

6.3 Φυσικές και τεχνητές γλώσσες

1. Ποια είναι τα στοιχεία που προσδιορίζουν μία γλώσσα;

Το αλφάβητο.

Είναι το σύνολο των στοιχείων που αποτελεί την γλώσσα. (Πχ η ελληνική γλώσσα περιέχει τα εξής στοιχεία: 48 χαρακτήρες, τα σημεία στίξης καθώς και τα ψηφία)

Το λεξιλόγιο.

Είναι το υποσύνολο όλων των ακολουθιών που δημιουργούνται από το αλφάβητο και είναι δεκτές από την γλώσσα. Πχ η ακολουθία ΑΒΓΑ είναι δεκτή ενώ η ΑΒΓΒΑ δεν είναι.

Την γραμματική.

Η γραμματική αποτελείται από το τυπικό και το συντακτικό.

Τυπικό: είναι το σύνολο των κανόνων που καθορίζουν αν μία λέξη είναι αποδεκτή.

Πχ Η λέξη «ΓΡΑΨΕ» είναι αποδεκτή αλλά η «ΓΡΑΠΣΕ» όχι.

Συντακτικό: Είναι το σύνολο των κανόνων που καθορίζει αν η διάταξη και η σύνδεση των λέξεων σε μία πρόταση είναι σωστή. Η γνώση συντακτικού στις φυσικές γλώσσες επιτρέπει την δημιουργία σωστών προτάσεων ενώ στις γλώσσες προγραμματισμού επιτρέπει την δημιουργία σωστών εντολών. ΠΧ:

Γλώσσα	Σωστό	Λάθος
Φυσική	Πάω να φάω	Φάω να πάω
Προγραμματισμού	Χ←3	3→Χ

Την σημασιολογία.

Είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων άρα και το νόημα των προτάσεων που δημιουργούνται.

Πχ Απογειώθηκε το ντροπαλό κατσαβίδι. (δεν έχει νόημα..)

Στις γλώσσες προγραμματισμού ο δημιουργός τους αποφασίζει για την σημασιολογία των λέξεων της γλώσσας

2. Ποιες είναι οι διαφορές φυσικών και τεχνητών γλωσσών;

Μια βασική διαφορά είναι η δυνατότητα εξέλιξης:

•Οι φυσικές γλώσσες εξελίσσονται συνεχώς και δημιουργούνται νέες λέξεις, αλλάζουν οι κανόνες γραμματικής/ συντακτικού με την πάροδο του χρόνου. Αυτό γίνεται γιατί χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων.

•Οι γλώσσες προγραμματισμού χρησιμοποιούνται για την επικοινωνία ανθρώπου Η/Υ και χαρακτηρίζονται από στασιμότητα. Ωστόσο και αυτές εξελίσσονται από τους δημιουργούς τους για να διορθώσουν αδυναμίες τους ή να καλύψουν μεγαλύτερο εύρος εφαρμογών αλλά και να ακολουθήσουν τις νέες εξελίξεις. Αλλάζουν σε:

επίπεδο διαλέκτου. Πχ Basic → QuickBasic

σε επίπεδο επέκτασης Πχ Basic → Visual Basic

6.4.1 Ιεραρχική σχεδίαση προγράμματος.

Η τεχνική της ιεραρχικής σχεδίασης (ή αλλιώς τεχνική από επάνω προς τα κάτω ή top-down), χρησιμοποιεί την στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα τα οποία είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος. Περιλαμβάνει:

•Τον καθορισμό των βασικών λειτουργιών ενός προγράμματος σε ανώτερο επίπεδο,

•Την διάσπαση των λειτουργιών αυτών σε όλο και μικρότερες λειτουργίες μέχρι το τελευταίο επίπεδο όπου οι λειτουργίες είναι πολύ απλές.

•Χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές για την υποβοήθηση της σχεδίασης.

•Υλοποιείται με τμηματικό προγραμματισμό.

6.4.2 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.
Εδώ πρέπει να σημειωθεί ότι ο τμηματικός προγραμματισμός διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.

6.4.3 Δομημένος προγραμματισμός.

Δεν είναι απλώς ένα είδος προγραμματισμού αλλά μια μεθοδολογία σύνταξης προγραμμάτων.
•Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο τις τρεις παρακάτω λογικές δομές καθώς και συνδυασμών τους.
Δομή ακολουθίας.
Δομή επιλογής.
Δομή επανάληψης.
•Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μία είσοδο και μόνο μία έξοδο.
` *{δηλαδή: έχει μία αρχή και μόνο ένα τέλος_προγράμματος}*
Αν και ο δομημένος προγραμματισμός αρχικά εμφανίστηκε ως μία προσπάθεια περιορισμού της εντολής GOTO σήμερα αποτελεί την βασική μεθοδολογία προγραμματισμού. Ο δομημένος προγραμματισμός βοηθάει την ανάλυση ενός προγράμματος σε τμήματα έτσι περιέχει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.
Γιατί η εντολή GOTO κρίνεται ακατάλληλη
•Η χρήση της εντολής αλλάζει την ροή ενός προγράμματος
•Η αλλαγή αυτή της ροής κάνει τα προγράμματα δύσκολα στην παρακολούθηση την κατανόηση και την συντήρηση.
Πλεονεκτήματα δομημένου προγραμματισμού
•Δημιουργία απλούστερων προγραμμάτων.
•Άμεση μεταφορά των αλγορίθμων σε προγράμματα.
•Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα.
•Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος.
•Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
•Ευκολότερη διόρθωση και συντήρηση.

6.5 Αντικειμενοστραφής προγραμματισμός.

Η ιδέα του αντικειμενοστραφούς προγραμματισμού ή της αντικειμενοστραφούς σχεδίασης έχει τις ρίζες της σε πολύ απλοϊκή ιδέα. Ένα πρόγραμμα περιγράφει "ενέργειες" (επεξεργασία) που εφαρμόζονται πάνω σε δεδομένα. Ένα βασικό ερώτημα που τίθεται είναι αν η φιλοσοφία, η δομή του προγράμματος είναι προτιμότερο να στηρίζεται στις "ενέργειες" ή στα δεδομένα. Η απάντηση σε αυτό το ερώτημα προσδιορίζει και τη βασική διαφορά ανάμεσα στις παραδοσιακές προγραμματιστικές τεχνικές και στην αντικειμενοστραφή προσέγγιση.
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.

6.7 Προγραμματιστικά περιβάλλοντα.

Πηγαίο πρόγραμμα (source)
•Το αρχικό πρόγραμμα που γράφει ο προγραμματιστής λέγεται πηγαίο πρόγραμμα.
•Η συγγραφή του γίνεται με τον συντάκτη *{βλέπε παρακάτω}*.
•Το πηγαίο πρόγραμμα δεν είναι κατανοητό από τον υπολογιστή. (εκτός αν είναι γραμμένο απ' ευθείας σε γλώσσα μηχανής)
Λάθη στο πηγαίο πρόγραμμα
Τα λάθη σε ένα πρόγραμμα διακρίνονται σε λογικά και συντακτικά.
•Λογικά λάθη.
Οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου.
{πχ ΜΟ ←x+ψ / 3 (ήθελε παρενθέσεις) …}
Εντοπίζονται παρά μόνο κατά την εκτέλεση του προγράμματος.
Είναι τα πλέον σοβαρά και δύσκολα στην διόρθωση.
•Συντακτικά λάθη.
Οφείλονται σε αναγραμματισμούς γραμμάτων εντολών, παράληψη δήλωσης δεδομένων κλπ.
{πχ Γραεψ αυτί για Γράψε}

Εντοπίζονται από τον μεταγλωττιστή ή τον διερμηνευτή
Πρέπει να διορθωθούν ώστε να δημιουργηθεί το εκτελέσιμο πρόγραμμα.

Συντάκτης

•Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου. *(Σαν το Word - Wordpad κλπ.)*
•Η συγγραφή του πηγαίου προγράμματος και η διόρθωση των λαθών του πηγαίου γίνεται με την βοήθεια του συντάκτη.

Μεταγλωττιστής

•Δέχεται σαν είσοδο ένα πρόγραμμα γραμμένο σε μία γλώσσα προγραμματισμού (υψηλού επιπέδου).
•Ανιχνεύει τα τυχόν συντακτικά λάθη. Αν βρεθούν λάθη ο προγραμματιστής τα διορθώνει (με τον συντάκτη) και υποβάλλει το πρόγραμμα ξανά προς μεταγλώττιση μέχρι να παραχθεί το σωστό.
•Αν δεν υπάρχουν λάθη και μόνο τότε, παράγει το **αντικείμενο πρόγραμμα,** το οποίο είναι ισοδύναμο με το πηγαίο αλλά εκφρασμένο πλέον σε γλώσσα μηχανής. Αυτό είναι πλέον τελείως ανεξάρτητο από το αρχικό πρόγραμμα, αλλά δεν είναι ακόμη εκτελέσιμο:
Η διαδικασία μέσω της οποίας καταλήγουμε στο εκτελέσιμο πρόγραμμα (γλώσσα μηχανής) είναι συνοπτικά:
Πηγαίο πρόγραμμα →μεταγλωττιστής → αντικείμενο πρόγραμμα → συνδέτης-φορτωτής → Εκτελέσιμο πρόγραμμα.

Συνδέτης φορτωτής

Το αντικείμενο πρόγραμμα που παράγει ο μεταγλωττιστής είναι μεν σε μορφή κατανοητή από τον υπολογιστή (γλώσσα μηχανής) αλλά δεν είναι εκτελέσιμο. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του. (Τα τμήματα αυτά είτε τα γράφει ο προγραμματιστής (υποπρογράμματα), είτε βρίσκονται στις βιβλιοθήκες της γλώσσας που χρησιμοποιεί.) Το πρόγραμμα που κάνει την σύνδεση αυτή ονομάζεται συνδέτης- φορτωτής.

Διερμηνευτής

Ο διερμηνευτής δέχεται ως είσοδο ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου και εκτελεί μία μία τις εντολές του αρχικού προγράμματος.
Η διαδικασία εκτέλεσης είναι η εξής:
•Διαβάζει μία από τις εντολές του αρχικού προγράμματος,
•ανιχνεύει τα (τυχόν) συντακτικά λάθη κάθε της εντολής,
•την μεταφράζει σε γλώσσα μηχανής
•την εκτελεί.
•Κατόπιν διαβάζει την επόμενη εντολή και επαναλαμβάνει την ίδια διαδικασία !

Ομοιότητες – Διαφορές διερμηνευτή-μεταγλωττιστή

Ομοιότητες:
•Και οι δύο μεταφράζουν το πηγαίο πρόγραμμα (υψηλού επιπέδου) σε γλώσσα μηχανής.
•Και οι δύο ανιχνεύουν τα συντακτικά λάθη.
Διαφορές:
•Ο μεταγλωττιστής μεταγλωττίζει όλο το πρόγραμμα και με την βοήθεια του συνδέτη – φορτωτή παράγεται το εκτελέσιμο.
•Ο διερμηνευτής εκτελεί μία μία τις εντολές και δεν χρειάζεται συνδέτη-φορτωτή
Διαφορές ως απόρροια των παραπάνω είναι:
•Ο διερμηνευτής αφού εκτελεί τις εντολές μία μία έχει το πλεονέκτημα της άμεσης διόρθωσης των λαθών. Για τον λόγο αυτό χρησιμοποιείται συνήθως κατά την συγγραφή-διόρθωση ενός προγράμματος.
•Η εκτέλεση ενός προγράμματος με τον διερμηνευτή είναι πιο αργή, γιατί για να εκτελεστεί το πρόγραμμα, πρέπει κάθε φορά να ξαναγίνεται η διερμηνεία από την αρχή, ενώ ο μεταγλωττιστής παράγει μια φορά το αντικείμενο πρόγραμμα και δεν χρειάζεται ξανά μεταγλώττιση αφού είναι σχεδόν εκτελέσιμο (θυμήσου τον συνδέτη)

(Σύγχρονα) προγραμματιστικά περιβάλλοντα

•Για την δημιουργία - εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον 3 προγράμματα:
1.Συντάκτης
2.Μεταγλωττιστής
3.Συνδέτης
•Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν τα προγράμματα αυτά με ενιαίο τρόπο *{και τα 3 προγράμματα σε 1}*
•Το κάθε προγραμματιστικό περιβάλλον έχει διαφορετικά εργαλεία και ιδιότητες (ανάλογα με την γλώσσα προγραμματισμού). Για παράδειγμα ένα περιβάλλον οπτικού προγραμματισμού παρέχει και ειδικό Συντάκτη για την δημιουργία γραφικών (μενού διαλόγου κλπ).

Κεφ7: Βασικά στοιχεία προγραμματισμού

Στο κεφάλαιο 7ο και 8ο το βιβλίο περιγράφει την γλώσσα προγραμματισμού ΓΛΩΣΣΑ

1.Το αλφάβητο της ΓΛΩΣΣΑΣ

Το αλφάβητο αποτελείται από

•Γράμματα
•Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
•Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
•Ψηφία
•Ειδικοί χαρακτήρες : **+ - * / ^ = > < , . ! & [ ] ( )** _ (Και ο κενός χαρακτήρας:)

2.Τύποι δεδομένων της ΓΛΩΣΣΑΣ

Ακέραιος.

Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. (δηλαδή θετικούς και αρνητικούς καθώς και το 0)

Πραγματικός.

Ο τύπος αυτός περιλαμβάνει τους πραγματικούς που είναι γνωστοί από τα μαθηματικά. (δηλαδή αριθμούς με δεκαδικό μέρος θετικούς και αρνητικούς καθώς και το 0)

Λογικός.

Ο τύπος αυτός δέχεται μόνο δύο τιμές ΑΛΗΘΗΣ και ΨΕΥΔΗΣ.

Χαρακτήρας.

Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και σε μία σειρά χαρακτήρων. Περιέχει οποιοδήποτε χαρακτήρα μπορεί να γραφεί στο πληκτρολόγιο. Οι χαρακτήρες πρέπει να βρίσκονται υποχρεωτικά ανάμεσα σε μονά εισαγωγικά. Επειδή μπορεί να περιέχει και αριθμούς ονομάζεται συχνά και αλφαριθμητικά.

{Πχ ‘ α ’ ‘ αλλά %’ ‘ 2αλλα23 ’}

3.Σταθερές (ή αλλιώς συμβολικές σταθερές)

(ορισμός) Είναι προκαθορισμένες τιμές που δεν μεταβάλλονται κατά την διάρκεια εκτέλεσης του προγράμματος. Μπορεί να είναι τύπου ακέραιες, πραγματικές, λογικές, χαρακτήρες. Η χρήση σταθερών κάνει το πρόγραμμα πιο κατανοητό, άρα πιο εύκολο να συντηρηθεί και να διορθωθεί.

4.Δήλωση (συμβολικών) Σταθερών

Η δήλωση των συμβολικών σταθερών γίνεται στο τμήμα δήλωσης σταθερών:

•Σύνταξη:

ΣΤΑΘΕΡΕΣ
Ονομα-1=σταθερή-τιμή-1
Ονομα-2=σταθερή-τιμή-2
…
Ονομα-ν=σταθερή-τιμή-ν

•Παράδειγμα

ΣΤΑΘΕΡΕΣ
Π=3.14
Ονομα=’Κώστας’

•Λειτουργία του τμήματος δήλωσης σταθερών

Αποδίδει ονόματα σε σταθερές τιμές.

Κάθε μία από αυτές τις σταθερές μπορεί να χρησιμοποιηθεί οπουδήποτε στο πρόγραμμα αλλά δεν είναι δυνατή η μεταβολή της τιμής της.

5.Μεταβλητές

(Ορισμός) Μια μεταβλητή παριστάνει μία ποσότητα που η τιμή της μπορεί να μεταβάλλεται.

•Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα αντιστοιχούνται από τον μεταγλωττιστή σε συγκεκριμένες θέσεις μνήμης.
•Η τιμή της μεταβλητής είναι η τιμή της αντίστοιχης θέσης μνήμης.
•Ενώ η τιμή της μεταβλητής (θέσης μνήμης) μπορεί να αλλάξει, ο τύπος της δεν αλλάζει ποτέ.
•Η ΓΛΩΣΣΑ επιτρέπει την χρήση τεσσάρων τύπων δεδομένων (ακέραιες λογικές πραγματικές χαρακτήρες)
•Το όνομα κάθε μεταβλητής ακολουθεί τους κανόνες δημιουργίας ονομάτων. {βλέπε παρακάτω … }
•Ως ονόματα μεταβλητών είναι καλή πρακτική να χρησιμοποιούμε ονόματα που υπονοούν το περιεχόμενό τους. Πχ «εμβαδόν» και όχι σκέτο «Ε»

6.Δήλωση Μεταβλητών

Η δήλωση των μεταβλητών γίνεται στο τμήμα δήλωσης μεταβλητών:

•Σύνταξη:

ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: Λίστα-μεταβλητών-1
ΠΡΑΓΜΑΤΙΚΕΣ: Λίστα-μεταβλητών-2
ΛΟΓΙΚΕΣ: Λίστα-μεταβλητών-3

ΧΑΡΑΚΤΗΡΕΣ: Λίστα-μεταβλητών-4
•Παράδειγμα:
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: α,β
ΠΡΑΓΜΑΤΙΚΕΣ: πι
ΛΟΓΙΚΕΣ: εύρεση
ΧΑΡΑΚΤΗΡΕΣ: όνομα,μαθημα
•Λειτουργία του τμήματος δήλωσης μεταβλητών:
Δηλώνει τον τύπο όλων των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα

7.Ονόματα-Κανόνες δημιουργίας

Κάθε προγράμματα όπως και τα δεδομένα (μεταβλητές και σταθερές) που χρησιμοποιεί έχουν ένα όνομα με το οποίο αναφερόμαστε σε αυτό.
Τα ονόματα αυτά μπορούν να αποτελούνται από
•Γράμματα:
Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
•Ψηφία: 0-9
•Ειδικούς χαρακτήρες: _ (κάτω παύλα)
Ενώ
•Απαγορεύεται η χρήση δεσμευμένων λέξεων ως ονόματα μεταβλητών (πχ γράψε διάβασε)
•Τα ονόματα μεταβλητών υποχρεωτικά αρχίζουν από γράμμα (όχι ψηφίο ή underscore)

8.Αριθμητικοί τελεστές

Αριθμητικός τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολ/μος
/	Διαίρεση
^	Ύψωση στην δύναμη
Div	Ακέραια διαίρεση
Mod	Υπόλοιπο ακέραιας διαίρεσης

9.Ιεραρχία αριθμητικών τελεστών

Η ιεραρχία είναι:
• ^
• *, / , div ,mod
•+, -
Προσοχή:
Αν συναντάμε παρενθέσεις προηγούνται οι τελεστές εντός των παρενθέσεων. Όσοι τελεστές έχουν ίδια ιεραρχία εκτελούνται από αριστερά προς τα δεξιά (όπως στα μαθηματικά).

10.Συναρτήσεις της ΓΛΩΣΣΑΣ

Η γλώσσα περιέχει 8 εγγενείς συναρτήσεις:
ΗΜ(x), ΣΥΝ(x), ΕΦ(x), ΛΟΓ(x), Ε(x), Τ_Ρ (x), Α_Μ(x), Α_Τ(x)
{οι συναρτήσεις αυτές είναι στην ουσία υποπρογράμματα τύπου συνάρτησης}

11.Αριθμητικές εκφράσεις.

{βλέπε και κεφάλαιο 2ο για εκφράσεις γενικότερα}
•Όταν μία τιμή προέρχεται από αριθμητικό υπολογισμό τότε αναφερόμαστε σε αριθμητικές εκφράσεις.
•Για την σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται:
-Αριθμητικές σταθερές.
-Αριθμητικές μεταβλητές.
-Αριθμητικοί τελεστές
-Συναρτήσεις.(από τις 8 παραπάνω)
-Παρενθέσεις.
•Κάθε έκφραση παριστάνει μία αριθμητική τιμή η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι απαραίτητο όλες οι μεταβλητές που εμφανίζονται σε μία έκφραση να έχουν πάρει προηγούμενα κάποια τιμή.
•Η εκτέλεση των πράξεων γίνεται με βάση την ιεραρχία των αριθμητικών τελεστών.

12.Εντολή εκχώρησης

Χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά την εκτέλεση του προγράμματος. Σε μία εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου. Το σύμβολο ←

διαφοροποιείται από το = το οποίο χρησιμοποιείται ως συγκριτικός τελεστής καθώς και ως τελεστής απόδοσης τιμών στο τμήμα δηλώσεων των σταθερών.
•Σύνταξη:
Όνομα-μεταβλητής←έκφραση
•Παράδειγμα:
Α←5
•Λειτουργία:
Υπολογίζεται η έκφραση δεξιά του ← και η τιμή της εκχωρείται στην μεταβλητή που βρίσκεται αριστερά.

13.Εντολή εισόδου (διάβασε)
•Σύνταξη:
Διάβασε λίστα-μεταβλητών
•Παράδειγμα:
διάβασε x,ψ
•Λειτουργία:
Η εντολή οδηγεί στην είσοδο τιμών από το πληκτρολόγιο και την εκχώρηση τους στις μεταβλητές που αναφέρονται στην λίστα.
Αναλυτική Λειτουργία:
•Η εντολή διάβασε ακολουθείται πάντα από τουλάχιστον ένα όνομα μεταβλητής. Αν υπάρχουν περισσότερες τότε αυτές χωρίζονται με κόμμα (,).
•Διακόπτεται η εκτέλεση του προγράμματος καθώς το πρόγραμμα περιμένει την εισαγωγή τιμών από το πληκτρολόγιο. Μόλις εισαχθούν οι τιμές συνεχίζεται η εκτέλεση του προγράμματος στην επόμενη εντολή.
•Εξυπηρετεί την επικοινωνία χρήστη → Η/Υ.

14.Εντολή εξόδου (γράψε)
•Σύνταξη:
Γράψε λίστα-στοιχείων
•Παράδειγμα:
Γράψε ‘ Ο φόρος είναι ’ , φπα, ’ € ’
•Λειτουργία:
Εμφανίζει σταθερές τιμές καθώς και τιμές μεταβλητών (ότι αναγράφεται στην λίστα)
Αναλυτική λειτουργία:
•Έχει ως αποτέλεσμα την εμφάνιση τιμών στην μονάδα εξόδου. Η μονάδα εξόδου μπορεί να είναι η οθόνη, ο εκτυπωτής, ή γενικά οποιαδήποτε μονάδα εξόδου έχει οριστεί.
•Η λίστα στοιχείων μπορεί να περιέχει σταθερές τιμές και ονόματα μεταβλητών. Όταν κάποιο όνομα μεταβλητής περιέχεται στην λίστα τότε αρχικά ανακτάται η τιμή της και μετά η τιμή αυτή εμφανίζεται στην μονάδα εξόδου.
•Η χρήση της αναφέρεται κυρίως στην εμφάνιση μηνυμάτων και αποτελεσμάτων
•Εξυπηρετεί την επικοινωνία Η/Υ → χρήστη

15.Δομή προγράμματος σε ΓΛΩΣΣΑ
Το πρόγραμμα δομείται με την εξής σειρά:
•Επικεφαλίδα:
ΠΡΟΓΡΑΜΜΑ Ονομα-προγράμματος
Το όνομα πρέπει να ακολουθεί τους κανόνες δημιουργίας ονομάτων
•Τμήμα δήλωσης σταθερών:
Αν χρησιμοποιεί σταθερές δηλώνονται εδώ
Τμήμα δήλωσης μεταβλητών
Δηλώνονται υποχρεωτικά τα ονόματα όλων των μεταβλητών μαζί με τον τύπο τους.
•Κύριο μέρος
Περιλαμβάνει όλες τις εκτελέσιμες εντολές ανάμεσα στην αρχή και στο τέλος_προγράμματος. Κάθε εντολή γράφεται σε ξεχωριστή γραμμή. Αν κάποια εντολή πρέπει να συνεχιστεί σε επόμενη γραμμή τότε ο πρώτος χαρακτήρας της επόμενης πρέπει να είναι ο: &
Αν ο πρώτος χαρακτήρας μίας γραμμής είναι ο ! τότε η γραμμή αυτή αποτελεί σχόλιο και δεν εκτελείται
{αγνοείται από τον μεταγλωττιστή}
•Αν το πρόγραμμα χρησιμοποιεί υποπρογράμματα τότε αυτά γράφονται μετά το τέλος προγράμματος.

Κεφ8: Επιλογή και επανάληψη

1. Λογική έκφραση
Μία λογική έκφραση είναι μία έκφραση που έχει λογική τιμή δηλαδή αληθής ή ψευδής.
Για την σύνταξη μιας λογικής έκφρασης χρησιμοποιούνται:
•Σταθερές
•Μεταβλητές
•Αριθμητικές εκφράσεις

•Συγκριτικοί τελεστές
•Λογικοί τελεστές
•Παρενθέσεις

2. Συγκριτικοί τελεστές

Τελεστής	Ελεγχόμενη σχέση
=	Ισότητα
<>	Ανισότητα
>	Μεγαλύτερο
<	Μικρότερο
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο

3. Συγκρίσεις

Οι συγκρίσεις έχουν ως αποτέλεσμα μία λογική τιμή (Αληθής-ψευδής). Γίνονται σε δεδομένα όλων των τύπων της ΓΛΩΣΣΑΣ : ακέραια και πραγματικά (αριθμοί), λογικά, χαρακτήρες:

•**Αριθμοί**: Η σύγκριση μεταξύ δύο αριθμών γίνεται με προφανή τρόπο.

•**Χαρακτήρες:**

Η σύγκριση μεταξύ ατομικών χαρακτήρων βασίζεται στην αλφαβητική σειρά ΠΧ το α < β :αληθής

Η σύγκριση σειράς χαρακτήρων βασίζεται στην σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρι να βρεθεί κάποια διαφορά Πχ κα**κ**ός < κα**λ**ός : αληθής (γιατί το κ είναι μικρότερο του λ)

•**Λογικοί:** Η σύγκριση μεταξύ λογικών έχει νόημα μόνο στην περίπτωση του = και του <>

4. Σύνθετες λογικές εκφράσεις

Σχηματίζονται από απλές λογικές εκφράσεις με την χρήση των λογικών τελεστών:

•Όχι
•Και
•Η

Η ιεραρχία τους είναι Όχι , Και , Η

5. Εμφωλευμένοι βρόχοι (επανάληψης)

(Ορισμός) Ονομάζονται δύο ή περισσότεροι βρόχοι που βρίσκονται ο ένας μέσα στον άλλο. Στην χρήση τους πρέπει να ακολουθούνται αυστηρά τρεις κανόνες:

•Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό, έτσι ο βρόχος που ξεκινάει τελευταίος ολοκληρώνεται πρώτος.
•Η είσοδος σε ένα βρόχο υποχρεωτικά γίνεται από την αρχή του.
•Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής.

Κεφ9: Πίνακες

1.Ορισμός πίνακα

Είναι ένα σύνολο αντικειμένων ιδίου τύπου τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που τον αποτελούν ονομάζεται στοιχείο. Η αναφορά σε ένα στοιχείο το πίνακα γίνεται με το όνομά του πίνακα ακολουθούμενο από **ένα** δείκτη.

2.Γενικά περί πινάκων

•**Τύπος δεδομένων:** Κάθε πίνακας περιέχει υποχρεωτικά δεδομένα του ίδιου τύπου (ακέραια , πραγματικά κλπ)

•**Δήλωση πίνακα:** Ο τύπος του πίνακα καθώς και ο αριθμός των στοιχείων του δηλώνεται στο τμήμα δηλώσεων μεταβλητών.

•**Όνομα:** Το όνομα ενός πίνακα καθορίζει μια ομάδα διαδοχικών θέσεων μνήμης.

•**Στοιχείο:** Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη.

•**Χρήση πινάκων:** Η ανάγνωση, επεξεργασία και εμφάνιση των στοιχείων των πινάκων γίνεται πάντοτε από βρόχους. Συνήθως με την χρήση της δομής Για αφού είναι προκαθορισμένο το πλήθος των στοιχείων.

3.Πότε πρέπει να χρησιμοποιούνται πίνακες- μειονεκτήματα

Μειονεκτήματα:

•Οι πίνακες απαιτούν μνήμη:Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.

•Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος: Οι πίνακες είναι στατικές δομές δεδομένων αφού το μέγεθός τους, δηλώνεται στην αρχή του προγράμματος παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση.

Πότε πρέπει να χρησιμοποιούνται;

•Όταν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στην μνήμη μέχρι το τέλος της εκτέλεσης.
•Η απόφαση για την χρήση ή όχι πίνακα είναι θέμα εμπειρίας στον προγραμματισμό.

{Γενικά καλό είναι να αποφεύγονται κυρίως γιατί καταναλώνουν πολύ μνήμη}

4.Πολυδιάστατοι πίνακες
Αν ο καθορισμός των στοιχείων ενός πίνακα απαιτεί παραπάνω από ένα δείκτη τότε μιλάμε για πολυδιάστατους πίνακες: πχ 3 δείκτες τρισδιάστατος 2 δείκτες δισδιάστατος κλπ
5.Τυπικές επεξεργασίες πινάκων
Είναι οι εξής:
•**Υπολογισμός αθροισμάτων στοιχείων του πίνακα**. Πολύ συχνά απαιτείται η εύρεση του αθροίσματος των στοιχείων ενός πίνακα ή του αθροίσματος των στοιχείων που έχουν μία ιδιότητα.
•**Εύρεση μεγίστου-ελαχίστου**. Αν ο πίνακας δεν είναι ταξινομημένος τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, αλλιώς αν είναι ταξινομημένος το μέγιστο και ελάχιστο βρίσκονται προφανώς στα δύο ακριανά στοιχεία του.
•**Ταξινόμηση των στοιχείων του πίνακα**. Ένας αλγόριθμος ταξινόμησης είναι ο αλγόριθμος φυσαλίδας αν και δεν είναι ο αποδοτικότερος.
•**Αναζήτηση ενός στοιχείου του πίνακα**. Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι:
1. Η σειριακή αναζήτηση
2. Δυαδική αναζήτηση (μόνο σε ταξινομημένους πίνακες)
•**Συγχώνευση δύο πινάκων**. Σκοπός της είναι η ένωση δύο ή περισσότερων ταξινομημένων πινάκων σε έναν που είναι και αυτός ταξινομημένος.

Κεφ:10 Υποπρογράμματα

10.1 Τμηματικός προγραμματισμός

Ορισμός: Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.
Τι είναι υποπρόγραμμα:
Ονομάζεται ένα τμήμα προγράμματος που επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα (subprogram).

10.2 Χαρακτηριστικά των υποπρογραμμάτων

• Ο χωρισμός ενός προγράμματος σε υποπρογράμματα προϋποθέτει την ανάλυση του προβλήματος σε υποπροβλήματα.
• Η ανάλυση αυτή δεν είναι πάντα εύκολη και δεν υπάρχουν συγκεκριμένοι κανόνες !
• Η δυσκολία όμως αυξάνεται όσο πιο μεγάλο και σύνθετο είναι το υποπρόγραμμα.
Βασικές 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.
{σημείωση: αυτό δεν σημαίνει ότι ένα υποπρόγραμμα πρέπει να έχει μόνο μια είσοδο με την έννοια της εισόδου τιμών σε αυτό. Το σχολικό εννοεί ότι το υποπρόγραμμα θα ενεργοποιείται από την αρχή του και όχι από την μέση !! Όμοια θα απενεργοποιείται στο τέλος του , όταν δει την εντολή τέλος_διαδικασίας ή τέλος_συνάρτησης. }
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.

10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού

• **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος**
Αντί να αντιμετωπίζουμε το συνολικό πρόβλημα εξετάζουμε και επιλύουμε τα υποπροβλήματα στα οποία αναλύεται, τα οποία είναι πιο απλά! Με τη σταδιακή επίλυση των υποπροβλημάτων και τη δημιουργία των αντιστοίχων υποπρογραμμάτων τελικά επιλύεται το αντίστοιχο πρόβλημα.
•**Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.**
Ο χωρισμός του προγράμματος σε μικρότερα 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. Αυτό είναι πολύ σημαντικό χαρακτηριστικό του σωστού προγραμματισμού, αφού ένα μεγάλο πρόγραμμα, χρειάζεται να συντηρηθεί από διαφορετικούς προγραμματιστές.
•**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 κατανοητό.
• **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.**
Ένα υποπρόγραμμα 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, όπως για τον υπολογισμό του ημίτονου ή του συνημίτονου ή την εντολή με την οποία εκτελεί μία συγκεκριμένη διαδικασία, για παράδειγμα γράφει στην οθόνη (εντολή ΓΡΑΨΕ). Αν λοιπόν χρειάζεται συχνά κάποια λειτουργία που δεν υποστηρίζεται απευθείας από τη γλώσσα, όπως για παράδειγμα η εύρεση του μικρότερου δύο αριθμών, τότε μπορεί να γραφεί το αντίστοιχο υποπρόγραμμα. Η συγγραφή πολλών υποπρογραμμάτων και η δημιουργία βιβλιοθηκών με αυτά, ουσιαστικά επεκτείνουν την ίδια τη γλώσσα προγραμματισμού.

10.4 Παράμετροι

(ορισμός) Μια παράμετρος είναι μια μεταβλητή που επιτρέπει το πέρασμά της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.
• Κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται από το αρχικό πρόγραμμα (που ονομάζεται κυρίως πρόγραμμα) ή από ένα υποπρόγραμμα.
• Το υποπρόγραμμα είναι μεν αυτόνομο αλλά συχνά πρέπει να επικοινωνεί με το υπόλοιπο πρόγραμμα. Έτσι δέχεται τιμές και επιστρέφει τιμές σε αυτόν που τον κάλεσε.
• Οι τιμές αυτές περνούν από το ένα υποπρόγραμμα στο άλλο με τις παραμέτρους.
• Οι παράμετροι είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μια ουσιώδη διαφορά:
Χρησιμοποιούνται για να περνάνε τιμές στα υποπρογράμματα.

10.5 Διαδικασίες και Συναρτήσεις

Διαδικασία: Είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελέσει όλες τις λειτουργίες ενός προγράμματος
• Μπορούν να εισάγουν δεδομένα να εκτελέσουν υπολογισμούς να μεταβάλλουν τις τιμές τωνμεταβλητών και να τυπώσουν αποτελέσματα.
• Με την χρήση των παραμέτρων μπορούν να μεταφέρουν **τιμές** και στα άλλα υποπρογράμματα.
• Τοποθετούνται μετά το τελος_προγράμματος
• Έχουν διαφορετικό τρόπο σύνταξης από τις συναρτήσεις
• Εκτελούνται με την ειδική εντολή κάλεσε και το όνομα της διαδικασίας
Συνάρτηση: Είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μια τιμή με το όνομά της.
• Έχουν πιο περιορισμένη λειτουργία από τις διαδικασίες
• Επιστρέφουν μόνο μια τιμή (ακέραια-πραγματική λογική ή και χαρακτήρα) σε αυτόν που την κάλεσε
• Μοιάζουν με τις συναρτήσεις των μαθηματικών
• Η χρήση τους είναι όμοια με τις **ενσωματωμένες συναρτήσεις** που υποστηρίζει η γλώσσα (Τ_Ρ κλπ)
• Έχουν διαφορετικό τρόπο σύνταξης από τις διαδικασίες.
• Εκτελούνται απλά με την εμφάνιση του ονόματός τους σε οποιαδήποτε έκφραση

10.5.1 Ορισμός και κλήση συναρτήσεων

• Κάθε συνάρτηση έχει την ακόλουθη δομή:
ΣΥΝΑΡΤΗΣΗ Ονομα (λίστα παραμέτρων) **:** τύπος συνάρτησης
Τμήμα δηλώσεων
ΑΡΧΗ
Εντολές
Όνομα← έκφραση
Εντολές
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
• Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της γλώσσας. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται προς την συνάρτηση κατά την κλήση της.
• Ο τύπος της συνάρτησης (ακέραια -πραγματική-λογική ή και χαρακτήρας) , δηλώνει την τιμή που επιστρέφει η συνάρτηση σε αυτόν που την κάλεσε.Στις εντολές της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μια εντολή εκχώρησης τιμής στο όνομα της συνάρτησης
Κλήση συναρτήσεων
• Κάθε συνάρτηση εκτελείται όπως ακριβώς εκτελούνται οι ενσωματωμένες συναρτήσεις της γλώσσας
• Απλώς αναφέρεται το όνομά της σε μια έκφραση ή σε μια εντολή και επιστρέφεται η τιμή της.
• Παράδειγμα
Χ← Τ_Ρ (Ψ)
• Περιγραφή του μηχανισμού της παραπάνω εντολής

- ο Το κύριο πρόγραμμα πριν από την κλήση της συνάρτησης γνωρίζει την τιμή της μεταβλητής Ψ
- ο Κατά την κλήση μεταβιβάζεται αυτή η τιμή στην αντίστοιχη παράμετρο της συνάρτησης.
- ο Η συνάρτηση υπολογίζει την τετραγωνική ρίζα.
- ο Το αποτέλεσμα αυτό εκχωρείται στο όνομα της συνάρτησης.
- ο Με το τέλος της συνάρτησης γίνεται επιστροφή στο κυρίως πρόγραμμα όπου η τιμή της ρίζας εκχωρείται στην μεταβλητή Χ

10.5.2 Ορισμός και κλήση διαδικασιών

• Κάθε διαδικασία έχει την ακόλουθη δομή:

ΔΙΑΔΙΚΑΣΙΑ Ονομα (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
Εντολές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

• Το όνομα της διαδικασίας είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ
• Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται προς την διαδικασία από αυτόν που την κάλεσε η/και επιστρέφονται σε αυτόν που την κάλεσε μετά το τέλος_διαδικασίας
• Στις εντολές της διαδικασίας μπορεί να υπάρχουν οποιεσδήποτε εντολές της γλώσσας

Κλήση διαδικασιών

• Κάθε διαδικασία εκτελείται όταν καλείται από το κύριο πρόγραμμα ή άλλη διαδικασία.
•Η κλήση σε διαδικασία πραγματοποιείται με την εντολή κάλεσε που ακολουθείται από το όνομα της διαδικασίας συνοδευόμενο μέσα σε παρενθέσεις με τη λίστα παραμέτρων:

Σύνταξη της ΚΑΛΕΣΕ:
ΚΑΛΕΣΕ Ονομα (λίστα παραμέτρων)

• **Λειτουργία της ΚΑΛΕΣΕ:**
- ο Η εκτέλεση του προγράμματος/ υποπρογράμματος, το οποίο χρησιμοποιεί την κάλεσε διακόπτεται και εκτελούνται οι εντολές της διαδικασίας που καλείται.
- ο Μετά το τέλος της διαδικασίας η εκτέλεση του προγράμματος συνεχίζεται από την εντολή που ακολουθεί την εντολή κάλεσε.
- ο Η λίστα των παραμέτρων ορίζει τις τιμές που περνούν στη διαδικασία **και** τις τιμές που αυτή επιστρέφει.
- ο Η λίστα παραμέτρων δεν είναι υποχρεωτική. Δηλαδή μια διαδικασία μπορεί να έχει μια καμία ή περισσότερες παραμέτρους
- ο Όταν υπάρχουν πολλές παράμετροι τότε άλλες χρησιμοποιούνται για να μεταβιβάσουν τιμές προς την διαδικασία και άλλες για να επιστρέψουν τιμές στο κυρίως πρόγραμμα.

10.5.3 Πραγματικές και τυπικές παράμετροι

• Ολες οι μεταβλητές έχουν ισχύ μόνο για το τμήμα προγράμματος στο οποίο έχουν δηλωθεί.
•**Ορισμός**: Η λίστα των τυπικών παραμέτρων καθορίζει τις παραμέτρους στην δήλωση του υποπρογράμματος. Μερικές γλώσσες προγραμματισμού τις ονομάζουν ορίσματα
•**Ορισμός:** Η λίστα των πραγματικών παραμέτρων καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος. Μερικές γλώσσες προγραμματισμού τις ονομάζουν απλά παραμέτρους.
• Τα ονόματα των τυπικών παραμέτρων και πραγματικών παραμέτρων μπορούν να είναι οποιαδήποτε. Αυτό δεν σημαίνει ότι αν έχουν το ίδιο όνομα είναι η ίδια μεταβλητή. Απεναντίας είναι υποχρεωτικά διαφορετικές μεταβλητές αφού ανήκουν σε διαφορετικά τμήματα προγράμματος

Κανόνες της λίστας παραμέτρων.

• Ο αριθμός των πραγματικών και τυπικών παραμέτρων πρέπει να είναι ο ίδιος.
• Κάθε παράμετρος αντιστοιχεί στην τυπική που βρίσκεται στην αντίστοιχη θέση.
• Η τυπική παράμετρος και αντίστοιχή της πραγματική πρέπει να είναι του ιδίου τύπου.

Η χρήση της στοίβας στην κλήση διαδικασιών.

• Όταν μια διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα τότε η αμέσως επόμενη εντολή-διεύθυνση του κυρίως προγράμματος (διεύθυνση επιστροφής) αποθηκεύεται από τον μεταφραστή σε μια στοίβα που ονομάζεται στοίβα χρόνου εκτέλεσης.
• Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής απωθείται από την στοίβα και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κυρίως πρόγραμμα.
• Η τεχνική αυτή εφαρμόζεται οποτεδήποτε μια διαδικασία ή συνάρτηση καλεί μια άλλη διαδικασία ή συνάρτηση.
• Παράδειγμα:
Έστω ότι μια διαδικασία α καλεί μια διαδικασία β η οποία με την σειρά της καλεί μια διαδικασία γ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στην στοίβα με την σειρά γ,β,α. Μετά την εκτέλεση κάθε υποπρογράμματος απωθείται η διεύθυνση από την στοίβα και ο έλεγχος μεταβιβάζεται στην διεύθυνση που μόλις απωθήθηκε.

10.6 Εμβέλεια μεταβλητών-σταθερών.

•Οι μεταβλητές αυτές στη **ΓΛΩΣΣΑ** είναι γνωστές στο αντίστοιχο υποπρόγραμμα και μόνο σε αυτό. Όλες οι μεταβλητές (και οι συμβολικές 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.
•Αυτό που καθορίζει την περιοχή που ισχύουν οι μεταβλητές και οι σταθερές είναι η **εμβέλεια** των μεταβλητών της συγκεκριμένης γλώσσας προγραμματισμού.

Ορισμός: το τμήμα του προγράμματος που ισχύουν οι μεταβλητές λέγεται εμβέλεια (scope) μεταβλητών.

Απεριόριστη εμβέλεια
•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**.
•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.
Περιορισμένη εμβέλεια
•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 μεταβλητής, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.
Μερικώς περιορισμένη 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.

Κεφάλαιο 13. Εκσφαλμάτωση Προγράμματος

13.1 Κατηγορίες λαθών

1. Να αναφέρετε τις κατηγορίες λαθών που μπορεί να παρουσιαστούν σε κάποιο πρόγραμμα.
   1) λάθη κατά την υλοποίηση (συντακτικά λάθη) **2)** λάθη κατά την εκτέλεση (λάθη χρόνου εκτέλεσης) **3)** λογικά λάθη.
2. Τι γνωρίζετε για τα λάθη κατά την υλοποίηση;

✓ Τα λάθη κατά το χρόνο υλοποίησης, προκαλούνται κυρίως από λανθασμένη σύνταξη εντολών προγράμματος. Τέτοια λάθη μπορεί να είναι η λανθασμένη συγγραφή μιας δεσμευμένης λέξης της γλώσσας προγραμματισμού ή η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της.
✓ Ένα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει προς το προγραμματιστή κάποιο προειδοποιητικό μήνυμα. Αν το πρόγραμμα περιέχει ένα λάθος αυτής της μορφής, δεν επιτρέπεται η εκτέλεσή του, μέχρι να το διορθώσει ο προγραμματιστής.
✓ Τα σύγχρονα προγραμματιστικά περιβάλλοντα μας προφυλάσσουν αυτόματα από τα λάθη κατά την υλοποίηση , αφού παρέχουν εργαλεία αυτόματου ελέγχου σύνταξης των εντολών και παρακολουθούν τον προγραμματιστή κατά τη συγγραφή του προγράμματος. Μόλις διαπιστώσουν κάποιο συντακτικό λάθος, σταματούν και απαιτούν τη διόρθωσή του. Συνήθως αντιλαμβάνονται ακριβώς το λάθος που δημιουργήθηκε και προτείνουν αναλυτικά τον τρόπο διόρθωσής του, εμφανίζοντας σε ενημερωτικό πλαίσιο την ορθή σύνταξη της εντολής που προκλήθηκε το λάθος.

3. Τι γνωρίζετε για τα λάθη κατά την εκτέλεση;

✓ Τα λάθη που προκαλούνται κατά το χρόνο εκτέλεσης του προγράμματος, είναι πιο επώδυνα γιατί συνήθως εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης και τις περισσότερες φορές προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το *κρέμασμα* (crash) του συστήματος.
✓ Όταν ένα λάθος προκληθεί κατά την εκτέλεση της εφαρμογής, είναι δυνατό να αντιμετωπισθεί μόνο με τη χρήση εντολών προγράμματος που το παγιδεύουν και εκτελούν τις κατάλληλες διαδικασίες χειρισμού του.

- ✓ Η πρόληψη τέτοιων λαθών είναι αρκετά δύσκολη, αφού συνήθως οφείλονται σε καταστάσεις που δεν είναι εύκολο να ελεγχθούν από τον προγραμματιστή, ενώ πολλές φορές εμφανίζονται μετά από μεγάλο χρονικό διάστημα. Τέτοια λάθη είναι δυνατό να προκληθούν από την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως η αναζήτηση διαγραμμένων αρχείων, η προσπάθεια διαίρεσης ενός αριθμού με το μηδέν, η υπερχείλιση μιας αριθμητικής μεταβλητής ή από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως η καταστροφή του σκληρού δίσκου του συστήματος, ο τερματισμός μιας σύνδεσης δικτύου και η αποσύνδεση του εκτυπωτή.

4. Τι γνωρίζετε για τα λογικά λάθη;

- ✓ Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος. Ενώ ο μεταγλωττιστής της γλώσσας προγραμματισμού δεν ανιχνεύσει κανένα συντακτικό λάθος και κατά την εκτέλεση του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων, τελικά δεν παράγονται τα επιθυμητά αποτελέσματα.
  Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη *διαδικασία ελέγχου* (testing) και την ανάλυση των αποτελεσμάτων των προγραμμάτων

13.2 Εκσφαλμάτωση.

1. Τι γνωρίζεται για την διαδικασία της εκσφαλμάτωσης;

- ✓ 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.
- ✓ Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά και τρυκ. 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.
- ✓ Σε ένα σύγχρονο προγραμματιστικό περιβάλλον δεν χρειάζεται ιδιαίτερα μνεία για τα λάθη που παρουσιάζονται κατά το χρόνο σχεδιασμού, αφού αυτά, όπως αναφέρθηκε, είναι συντακτικά λάθη και τις περισσότερες φορές το περιβάλλον προγραμματισμού τα ανιχνεύει αυτόματα και προτείνει τη διόρθωσή τους ή τα εντοπίζει ο μεταγλωττιστής.
- ✓ 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.

Ενότητα 1. Δομές δεδομένων και Αλγόριθμοι

1.1 Στοίβα

1. Ποια δομή δεδομένων ονομάζεται στοίβα;

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.

2. Πως ονομάζεται η μέθοδος λειτουργίας μίας στοίβας; Δώστε ένα παράδειγμα από την καθημερινή σας ζωή.

Η μέθοδος λειτουργίας της στοίβας ονομάζεται «Τελευταίο μέσα πρώτο Έξω» (LIFO – Last In First Out). Χαρακτηριστικό παράδειγμα είναι η στοίβα με τα πιάτα. Όταν τα πλένουμε, τα τοποθετούμε το ένα πάνω στο άλλο και χρησιμοποιούμε πρώτο το πιάτο που βρίσκεται πάνω-πάνω (δηλαδή αυτό που μπήκε τελευταίο στη στοίβα).

3. Ποιες είναι οι κύριες λειτουργίες που εφαρμόζονται σε μία στοίβα;

Οι δύο βασικές λειτουργίες που εφαρμόζονται σε μία στοίβα είναι:

i. 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.

ii. 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.

Υλοποίηση ώθησης και απώθησης σε κώδικα: Να γράψετε τμήμα αλγορίθμου το οποίο θα υλοποιεί τις λειτουργίες της ώθησης και της απώθησης σε μία στοίβα χρησιμοποιώντας πίνακα Α 5 θέσεων.

Κώδικας για Ώθηση στοιχείου	**Παρατηρήσεις**
Διάβασε στοιχείο **Αν** top<5 **τότε** **! έλεγχος για γεμάτη στοίβα** top←top+1 **! αύξηση δείκτη top κατά 1** A[top]←στοιχείο **! τοποθέτηση στοιχείου** **Αλλιώς** **Γράψε** 'Η στοίβα είναι γεμάτη, υπερχείλιση' **Τέλος_αν**	Κατά την διαδικασία της ώθησης, θα πρέπει να ελέγξουμε πως η στοίβα δεν είναι γεμάτη, για να μην δημιουργήσουμε υπερχείλιση. Εφόσον υπάρχει χώρος, πρώτα αυξάνουμε τον δείκτη top κατά 1 και στη νέα θέση τοποθετούμε το στοιχείο.

Κώδικας για Απώθηση στοιχείου	**Παρατηρήσεις**
Αν top>=1 **τότε** **! έλεγχος για άδεια στοίβα** **Γράψε** A[top] **! εμφάνιση στοιχείου** top←top-1 **! μείωση δείκτη top κατά 1** **Αλλιώς** **Γράψε** 'Η στοίβα είναι άδεια, υποχείλιση' **Τέλος_αν**	Κατά την διαδικασία της απώθησης, θα πρέπει να ελέγξουμε πως η στοίβα δεν είναι άδεια, για να μην δημιουργήσουμε υποχείλιση. Εφόσον υπάρχει στοιχείο, πρώτα το εμφανίζουμε και στη συνέχεια μειώνουμε τον δείκτη top κατά 1.

1.2 Ουρά

1. Ποια δομή δεδομένων ονομάζεται «ουρά»;

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.

2. Πως ονομάζεται η μέθοδος λειτουργίας μίας ουράς; Να παρουσιάσετε ένα παράδειγμα ουράς από τη καθημερινής σας ζωή.

Η μέθοδος λειτουργίας της ουράς ονομάζεται Πρώτο Μέσα – Πρώτο Έξω (FIFO – First In First Out). Ένα παράδειγμα είναι η ουρά στην τράπεζα, στα ταμεία ενός σούπερ μάρκετ κτλ.

3. Ποιες είναι οι βασικές λειτουργίες που εφαρμόζονται σε μία ουρά;

Οι δύο βασικές λειτουργίες που εφαρμόζονται σε μία ουρά είναι:

✓ Η εισαγωγή (enqueue) ενός στοιχείου στο πίσω άκρο της ουράς. Θα πρέπει να προσέξουμε να μην είναι γεμάτη η ουρά.
✓ Η εξαγωγή (dequeue) ενός στοιχείου από το εμπρός άκρο της ουράς. Θα πρέπει να προσέξουμε να υπάρχει κάποιο στοιχείο στην ουρά.

Υλοποίηση εισαγωγής και εξαγωγής σε κώδικα: Να γράψετε τμήμα αλγορίθμου το οποίο θα υλοποιεί τις λειτουργίες της εισαγωγής και της εξαγωγής σε μία ουρά με πίνακα Α 5 θέσεων.

Κώδικας για εισαγωγή στοιχείου	**Παρατηρήσεις**
Διάβασε στοιχείο **Αν** πίσω= 5 **τότε** ! **έλεγχος για γεμάτη ουρά** **Γράψε** 'γεμάτη ουρά' **Αλλιώς_αν** πίσω=0 **και** εμπρός=0 **τότε** ! **άδεια ουρά** εμπρός←1 ! **οι δείκτες εμπρός και πίσω** πίσω←1 ! **γίνονται 1** Α[πίσω]←στοιχείο ! **και εισάγεται το στοιχείο** **Αλλιώς** ! **υπάρχει χώρος για το στοιχείο** πίσω←πίσω+1 ! **ο δείκτης πίσω αυξάνεται + 1** Α[πίσω]←στοιχείο ! **και εισάγεται το στοιχείο** **Τέλος_αν** **Παρατήρηση:** επειδή στην ουρά η εισαγωγή γίνεται μόνο από το πίσω άκρος της, η συνθήκη πίσω=μέγεθος_ουράς σημαίνει πως είναι γεμάτη, ακόμη και αν υπάρχει χώρος στο μπροστά άκρο.	Κατά την διαδικασία της εισαγωγής θα πρέπει αρχικά να ελέγξουμε να μην είναι γεμάτη η ουρά. Αν δεν είναι, διακρίνουμε δύο περιπτώσεις: 1) Να είναι άδεια, οπότε αρχικοποιούμε τους δύο δείκτες με 1 και εισάγουμε το στοιχείο. 2) Να έχει χώρο για το στοιχείο, οπότε αυξάνουμε τον δείκτη πίσω κατά 1 και στη νέα θέση εισάγουμε το στοιχείο.

Κώδικας για εξαγωγή στοιχείου	**Παρατηρήσεις**
Αν εμπρός=0 και πίσω=0 **τότε** ! **άδεια ουρά** **Γράψε** 'Η ουρά είναι άδεια' **Αλλιώς_αν** εμπρός=πίσω **τότε** ! **ένα μόνο στοιχείο** **Γράψε** 'Εξάγεται το στοιχείο', Α[εμπρός] εμπρός←0 ! **οι δείκτες εμπρός και πίσω** πίσω←0 ! **αρχικοποιούνται με 0 (άδεια ουρά)** **Αλλιώς** ! **υπάρχουν και άλλα στοιχεία** **Γράψε '** Εξάγεται το στοιχείο', Α[εμπρός] εμπρός←εμπρός+1 **Τέλος_αν**	Κατά την διαδικασία της εξαγωγής, θα πρέπει να ελέγξουμε πως η ουρά δεν είναι άδεια. Αν δεν είναι, διακρίνουμε δύο περιπτώσεις: 1) Να υπάρχει μόνο ένα στοιχείο, οπότε οι δείκτες εμπρός και πίσω αρχικοποιούνται με 0, καθώς η ουρά αδειάζει. 2) Να υπάρχουν και άλλα στοιχεία, οπότε ο δείκτης εμπρός αυξάνεται κατά.

Ενότητα 1.3.1: Λίστες

1. Ποια είναι τα γενικά χαρακτηριστικά μίας λίστας;

Οι λίστες είναι δυναμικές δομές δεδομένων, οπότε μπορούμε να προσθέσουμε και να αφαιρέσουμε κόμβους από αυτές κατά την διάρκεια της εκτέλεσης του προγράμματος.

2. Περιγράψτε την μορφή της ελεύθερης μνήμης ενός υπολογιστή και πως αυτή επηρεάζει την αποθήκευση δεδομένων.

Συνήθως η μνήμη του υπολογιστή δεν περιέχει τις ελεύθερες θέσεις της με την σειρά σε συνεχόμενες θέσεις. Άρα στην περίπτωση αυτή δεν θα μπορούσαμε να αποθηκεύσουμε τα δεδομένα μας σε πίνακα. Μπορούμε όμως να χρησιμοποιήσουμε τις διάσπαρτες αυτές θέσεις για να αποθηκεύσουμε τα δεδομένα μας σε μορφή λίστας.

3. Από ποια επιμέρους τμήματα αποτελείται μία συνδεδεμένη λίστα;

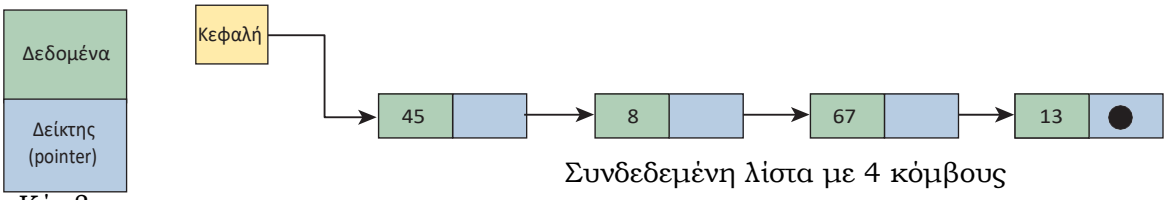
Μία συνδεδεμένη λίστα αποτελείται από μία σειρά από κόμβους, που συνήθως βρίσκονται σε απομακρυσμένες θέσεις μνήμης. Κάθε κόμβος αποτελείται από δύο μέρη:

✓ Το πρώτο τμήμα περιέχει τα δεδομένα, τα οποία μπορεί να περιέχει μία ή περισσότερες αριθμητικές ή αλφαριθμητικές τιμές.
✓ Το δεύτερο τμήμα είναι ονομάζεται δείκτης (pointer). Είναι ένας ιδιαίτερος τύπος δεδομένων, που προσφέρεται από τις περισσότερες γλώσσες προγραμματισμού. Δεν λαμβάνει αριθμητικές τιμές, αλλά οι τιμές του είναι διευθύνσεις στην κύρια μνήμη του υπολογιστή που χρησιμοποιούνται για τη σύνδεση στοιχείων μιας δομής που βρίσκονται σε μη συνεχόμενες θέσεις μνήμης. Στην ουσία «δείχνει» σε ποια διεύθυνση βρίσκεται ο επόμενος κόμβος μίας συνδεδεμένης λίστας

4. Να δώσετε τον ορισμό της συνδεδεμένης λίστας.

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** και αναπαρίσταται με το σύμβολο «⏚».
- ✓ Οι δείκτες των υπόλοιπων κόμβων περιέχουν την διεύθυνση του επόμενου κόμβου και τους συμβολίζουμε με ένα βέλος που υποδηλώνει την σύνδεση μεταξύ του προηγούμενου και του επόμενου κόμβου.
- ✓ 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)**.



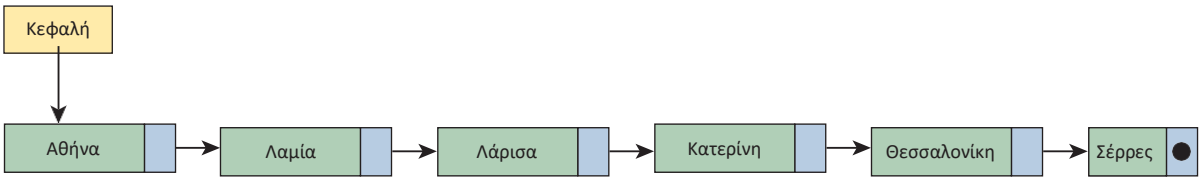
Συνδεδεμένη λίστα με 4 κόμβους

5. Πως γίνεται η πρόσβαση στους κόμβους μίας συνδεδεμένης λίστας;

Οι κόμβοι μιας (απλά) συνδεδεμένης λίστας δεν έχουν ονόματα και 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, μέσω της διεύθυνσης που περιέχεται στον δείκτη Κεφαλή. 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 επιθυμούμε να προσπελάσουμε τον τρίτο κόμβο μίας λίστας, θα ξεκινήσουμε από την κεφαλή, η οποία θα μας δείξει την διεύθυνση του πρώτου κόμβου, ο δείκτης του πρώτου θα μας δείξει την διεύθυνση του δεύτερου και ο δείκτης του δεύτερου την διεύθυνση του τρίτου.

6. Να δώσετε ένα παράδειγμα προσθήκης και αφαίρεσης κόμβου σε μία συνδεδεμένη λίστα.

Το παρακάτω σχήμα δείχνει μία συνδεδεμένη λίστα που αναπαριστά αν ταξίδι, με αφετηρία την Αθήνα και τερματισμό τις Σέρρες.

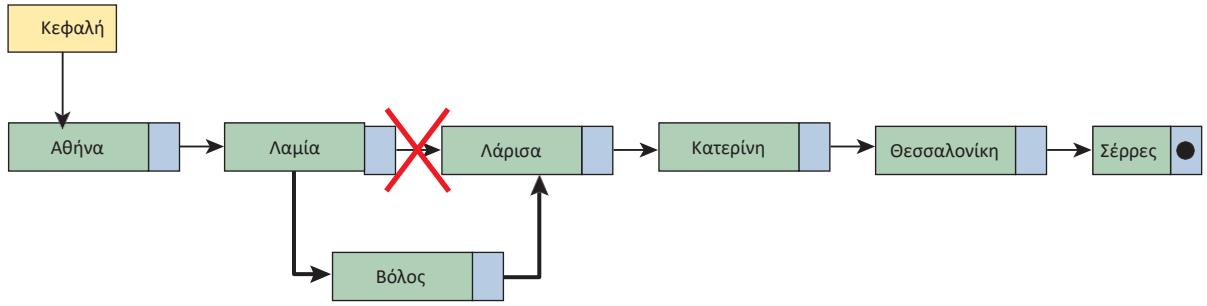


Μορφή αρχικής λίστας

Αν αποφασίζουμε να επισκεφτούμε και τον Βόλο μετά τη Λαμία και πριν τη Λάρισα, τότε θα πρέπει να εισάγουμε άλλο ένα κόμβο. Οι ενέργειες που απαιτούνται είναι:

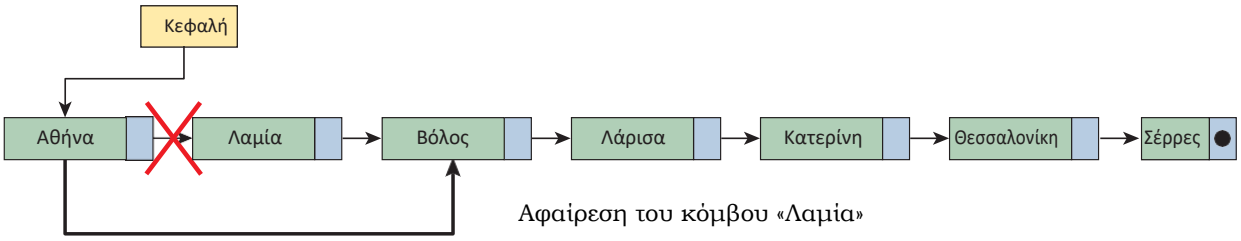
- ✓ Ο δείκτης του δεύτερου κόμβου θα «δείχνει» το νέο κόμβο.
- ✓ Ο δείκτης του νέου κόμβου θα «δείχνει» τον τέταρτο κόμβο.

Έτσι οι κόμβοι της λίστας διατηρούν τη λογική τους σειρά, αλλά οι φυσικές θέσεις στη μνήμη μπορεί να είναι τελείως διαφορετικές.



Εισαγωγή του κόμβου «Βόλος» μετά την Λαμία και πριν τη Λάρισα

Αν επιπλέον αποφασίζουμε να μην επισκεφτούμε τη Λαμία, θα πρέπει να διαγράψουμε τον αντίστοιχο κόμβο. Πρακτικά θα πρέπει να αλλάξει τιμή ο δείκτης του προηγούμενου κόμβου και να δείχνει πλέον στον επόμενο αυτού που διαγράφεται. Ο κόμβος που διαγράφεται αποτελεί «άχρηστο δεδομένο» και ο χώρος μνήμης που καταλάμβανε παραχωρείται για άλλη χρήση.



Αφαίρεση του κόμβου «Λαμία»

7. Σε ποιες περιπτώσεις αξιοποιούνται οι συνδεδεμένες λίστες;

Οι συνδεδεμένες λίστες αξιοποιούνται για την υλοποίηση της στοίβας και της ουράς λόγω της δυνατότητας αυξομείωσης του μεγέθους τους.

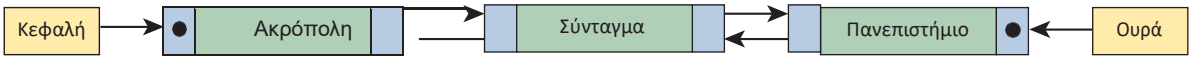
8. Τι ονομάζουμε «διπλά συνδεδεμένη»; Να αναλύσετε τα χαρακτηριστικά της.

Μία λίστα ονομάζεται «διπλά συνδεδεμένη» όταν μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις. Για την υλοποίηση της χρειαζόμαστε δύο δείκτες:

- ✓ Τον δείκτη «κεφαλή» που δείχνει τον πρώτο κόμβο της λίστας.
- ✓ Τον δείκτη «ουρά» που δείχνει τον τελευταίο κόμβο της λίστας.

Με τον τρόπο αυτό μπορούμε, να την διαβάσουμε και προς τις δύο κατευθύνσεις. Ένα παράδειγμα είναι οι σταθμοί του μετρό. Άλλα χαρακτηριστικά είναι :

- ✓ Κάθε κόμβος της διπλά συνδεδεμένης λίστας συνδέεται με τον αμέσως προηγούμενο και τον αμέσως επόμενο κόμβο της λίστας, εκτός από τον πρώτο και τον τελευταίο.
- ✓ Κάθε κόμβος περιέχει δύο δείκτες.
- ✓ Ο πρώτος και ο τελευταίος κόμβος έχουν δείκτη με τιμή NULL.
- ✓ Σε μία διπλά συνδεδεμένη λίστα διευκολύνεται η ταξινόμηση και η αναζήτηση, ωστόσο, αυξάνεται η πολυπλοκότητα στη διαχείριση των κόμβων, καθώς απαιτείται επιπλέον χώρος για τον δεύτερο δείκτη (επιπρόσθετη μνήμη για κάθε κόμβο)



Διπλά συνδεδεμένη λίστα

9. Ποιες είναι οι διαφορές ενός πίνακα από μία λίστα;

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

10. Να αναφέρετε τα πλεονεκτήματα και τα μειονεκτήματα των λιστών έναντι των πινάκων.

Πλεονεκτήματα:

- ✓ Το δυναμικό τους μέγεθος.
- ✓ Η ευκολία εισαγωγής και διαγραφής από οποιοδήποτε μέρος της λίστας.
- ✓ Η μη αναγκαιότητα δήλωσης του μεγέθους τους.

Μειονεκτήματα:

- ✓ 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.
- ✓ 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.

11. Ποιες είναι οι βασικές πράξεις των συνδεδεμένων λιστών;

Οι βασικές πράξεις των συνδεδεμένων λιστών είναι οι ακόλουθες:

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

Παράδειγμα 1 - Δημιουργία λίστας από τις πληροφορίες των δεικτών: Για την δημιουργία μίας λίστας δίνονται οι ακόλουθες πληροφορίες: στον πίνακα δεδομένα[5] αποθηκεύονται τα δεδομένα των κόμβων και στον πίνακα δείκτης[5] η διεύθυνση του επόμενου κόμβου. Στην πρώτη θέση έχουν τοποθετηθεί τα στοιχεία του πρώτου κόμβου. Να δημιουργήσετε την λίστα.

Δεδομένα

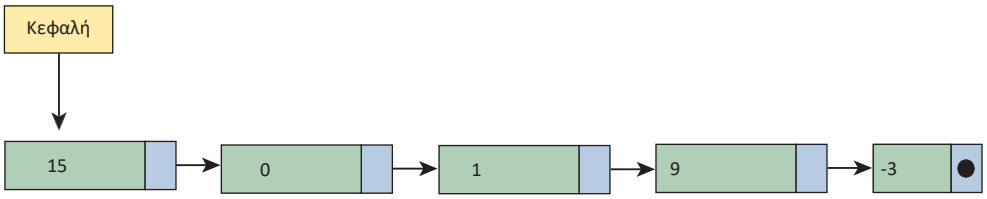
15	-3	0	1	9
1	2	3	4	5

Δείκτης

3	0	4	5	2
1	2	3	4	5

Επεξήγηση λύσης:
Σε κάθε κόμβο έχουμε δύο πληροφορίες: Δεδομένα και Δείκτης, ο οποίος δείχνει τον επόμενο κόμβο που θα εισαχθεί στη λίστα. Στην θέση 1 έχουμε τον πρώτο κόμβο από τα δεδομένα της άσκησης. Στη συνέχεια σκεπτόμαστε ως εξής:

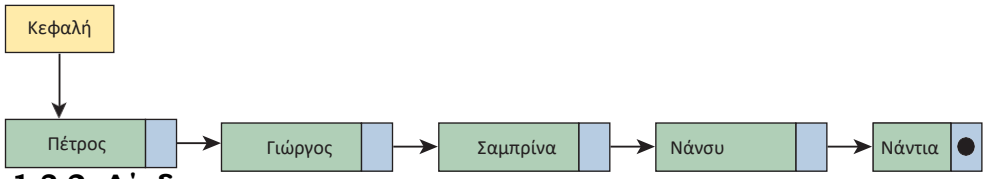
- ✓ Έχουμε δείκτης[1]=3, άρα ο επόμενος κόμβος βρίσκεται στο δεδομένα[3]=0 το οποίο τοποθετούμε δεύτερο στη λίστα.
- ✓ Έχουμε δείκτης[3]=4, άρα ο επόμενος κόμβος βρίσκεται στο δεδομένα[4]=1, το οποίο τοποθετούμε τρίτο στη λίστα.
- ✓ Έχουμε δείκτης[4]=5, άρα ο επόμενος κόμβος βρίσκεται στο δεδομένα[5]=9, το οποίο τοποθετούμε τέταρτο στη λίστα.
- ✓ Έχουμε δείκτης[5]=2, άρα ο επόμενος κόμβος βρίσκεται στο δεδομένα[2]=-3, το οποίο τοποθετούμε τελευταίο στη λίστα. Επίσης έχουμε δείκτης[2]=0, κάτι που επαληθεύει πως είναι το τελευταίο στοιχείο (τιμή Null).



Παράδειγμα 2 - Δημιουργία λίστας με βάση τον επόμενο κόμβο: Για την δημιουργία μίας λίστας, κάποιος μαθητής ακολούθησε την παρακάτω διαδικασία: για κάθε κόμβο σημείωσε δύο πληροφορίες: αρχικά τα δεδομένα και στη συνέχεια έναν αριθμό που έδειχνε ποιος είναι ο επόμενος στη σειρά κόμβος. Να δημιουργήσετε τη λίστα.
Γιώργος 3, Πέτρος 2, Νάνσυ 5, Νάντια 0, Σαμπρίνα 4.

Επεξήγηση λύσης:

- ✓ Ο πρώτος κόμβος θα είναι αυτός που έχει ως επόμενο κόμβο το 2, οπότε θα είναι ο Πέτρος.
- ✓ Τελευταίος θα είναι το κόμβος με επόμενο κόμβο το 0, δηλαδή η Νάντια.
- ✓ Οι υπόλοιποι κόμβοι τοποθετούνται με αύξουσα σειρά αριθμού.



Ενότητα 1.3.2: Δένδρα

1. Να περιγράψετε τις ακόλουθες έννοιες: δένδρο, ακμή, γονέας, παιδί, ρίζα, αδέρφια, φύλλα.

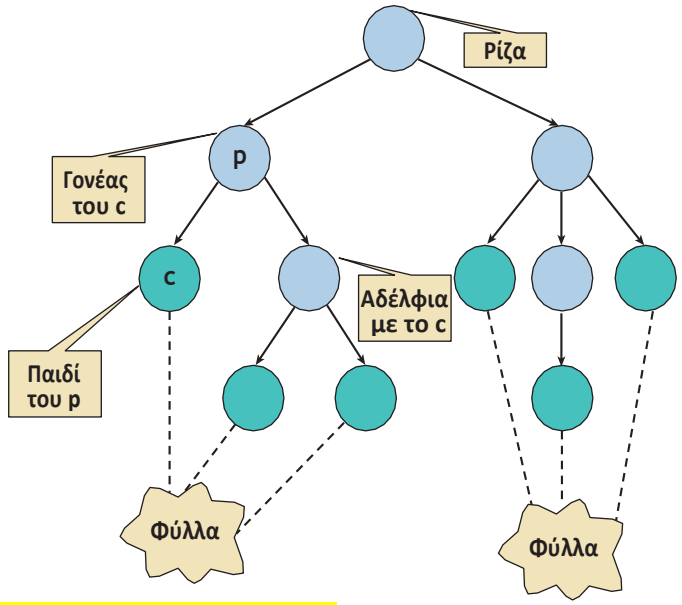
- ✓ **Δένδρο:** είναι ένα σύνολο από κόμβους οι οποίοι συνδέονται μεταξύ τους με ακμές.
- ✓ **Ακμή:** είναι μία «γραμμή» που συνδέει δύο κόμβους.
- ✓ **Γονέας + Παιδί:** όταν δύο κόμβοι ενώνονται μεταξύ τους με μία ακμή, γονέας είναι ο κόμβος από τον οποίο ξεκινάει η ακμή και παιδί είναι ο κόμβος στον οποίο καταλήγει η ακμή.
- ✓ **Ρίζα (root):** είναι ο κόμβος που δεν έχει γονέα και βρίσκεται στην κορυφή του δένδρου.

- ✓ **Αδέρφια:** είναι κόμβοι που έχουν τον ίδιο γονέα.
- ✓ **Φύλλα:** είναι κόμβοι που δεν έχουν παιδιά.

Ένας κόμβος μπορεί να έχει κανένα, ένα η περισσότερα παιδιά

2. Να δώσετε τον ορισμό του δένδρου.

Ένα **δένδρο (tree)** είναι μία δομή που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο ακμών μεταξύ των



κόμβων με βάση τους εξής κανόνες:

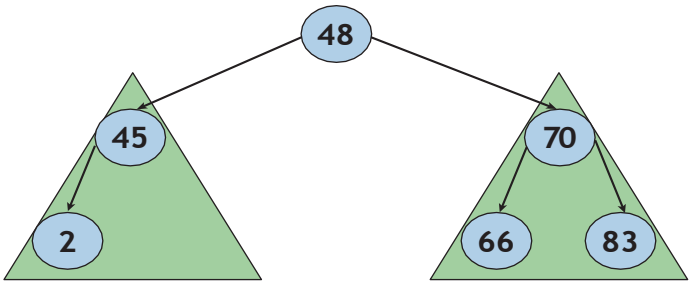
- ✓ Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται ρίζα. Αυτός είναι ένας κόμβος χωρίς γονέα.
- ✓ Για κάθε κόμβο c, εκτός από τη ρίζα, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p.
- ✓ Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.
  - Μπορούμε να έχουμε και ένα απλό δέντρο το οποίο απαρτίζεται από έναν μόνο κόμβο. Αυτός ο κόμβος είναι και η ρίζα του δέντρου, διότι δεν έχει γονέα και φύλο, και διότι δεν έχει παιδιά.
  - 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.

3. Να περιγράψετε τι είναι ένα «υποδένδρο».

Μέσα σε ένα δένδρο μπορούμε να εντοπίσουμε και άλλα μικρότερα δένδρα, που ονομάζονται υποδένδρα. Πιο συγκεκριμένα, κάθε κόμβος ενός δένδρου μπορεί να θεωρηθεί ως ρίζα ενός υποδένδρου, δηλαδή ενός άλλου μικρότερου δένδρου, που ξεκινάει από τον κόμβο αυτόν. Όπως φαίνεται και από το ακόλουθο σχήμα, ο κόμβος 48 είναι ρίζα και έχει δύο υποδένδρα που ξεκινούν από τους κόμβους 45 και 70 αντίστοιχα. Ο κόμβος 45 έχει ένα υποδένδρο που αποτελείται από τον κόμβο 2. Ο κόμβος 70 έχει δύο υποδένδρα που αποτελούνται από τους κόμβους 66 και 83 αντίστοιχα. Τα υποδένδρα των κόμβων 2, 66 και 83 είναι κενά.

4. Δώστε ένα παράδειγμα περίπτωσης ενός «διατεταγμένου δένδρου».

Αν για παράδειγμα θέλουμε να μοντελοποιήσουμε την ιεραρχική σχέση των μελών μας οικογένειας και μας ενδιαφέρει να οργανώσουμε τα αδέλφια σύμφωνα με την ηλικία τους, τότε τα αδέλφια που θα έχουν γεννηθεί



νωρίτερα θα τοποθετηθούν στην δενδρική δομή πιο αριστερά σε σχέση με αυτά που θα έχουν γεννηθεί αργότερα. Σε αυτή την περίπτωση, που για κάθε κόμβο υπάρχει μία γραμμική σχέση μεταξύ των παιδιών του κόμβου αυτού, αναφερόμαστε σε ένα διατεταγμένο δένδρο.

5. Να δώσετε παραδείγματα της χρησιμότητας των δένδρων.

Τα δένδρα είναι μία μη – γραμμική ευέλικτη δομή και έχουν αρκετές δυνατότητες και εφαρμογές, κάποιες από τις οποίες είναι οι ακόλουθες:

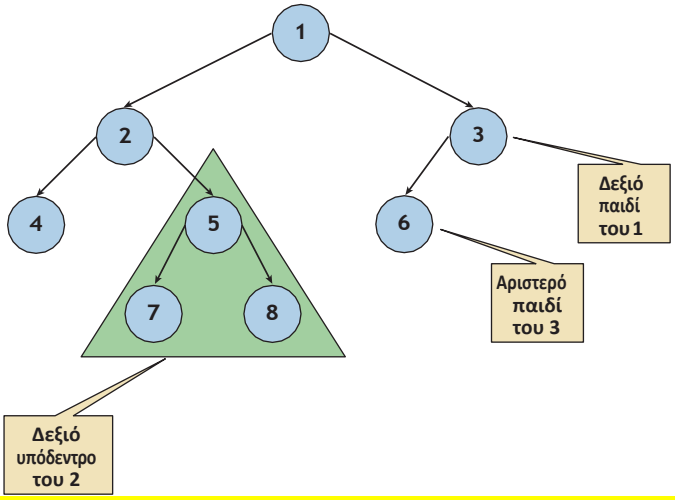
- ✓ Έχουν μεγάλη «δυναμικότητα», είναι δηλαδή εύκολο να προσθέσουμε, να αφαιρέσουμε ή αν αναζητήσουμε στοιχεία σε ένα δένδρο.

- ✓ Η δομή των δένδρων μεταφέρει πληροφορίες. Για παράδειγμα, σε ένα σύστημα αρχείων, αν ο κατάλογος “users” είναι παιδί της ρίζας “/” και ότι ο κατάλογος “student” είναι παιδί του “users” μπορούμε να συμπεράνουμε ότι υπάρχει η διαδρομή “/users/student”.
- ✓ Χρησιμοποιούνται για αναπαραστάσεις δεδομένων του πραγματικού χρόνου, αλλά και της πληροφορικής, που διέπονται από ένα είδος ιεραρχίας, όπως πχ οικογενειακό δένδρο, πίνακας περιεχομένων ενός βιβλίου, τα αρχεία και οι φάκελοι ενός υπολογιστή κτλ.
- ✓ Μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων, όπως για παράδειγμα η συμπίεση εικόνων, η ταξινόμηση, η αυτόματη συμπλήρωση λέξεων σε συσκευές κινητών, η μεταγλώττιση ενός προγράμματος, η λήψη αποφάσεων. Στα δένδρα απόφασης, κάθε κόμβος αντιπροσωπεύει ένα χαρακτηριστικό (ιδιότητα), κάθε ακμή αντιπροσωπεύει μία απόφαση (κανόνα) και κάθε φύλλο αντιπροσωπεύει ένα αποτέλεσμα.
- ✓ Τα «δένδρα παιχνιδιού» μπορούν να χρησιμοποιηθούν για την μοντελοποίηση πιθανών κινήσεων σε παιχνίδια όπως πχ το σκάκι ή η τρίλιζα.
- ✓ Διαδεδομένα είναι και τα δένδρα για την αναπαράσταση και κατ επέκταση τον υπολογισμό αριθμητικών εκφράσεων.

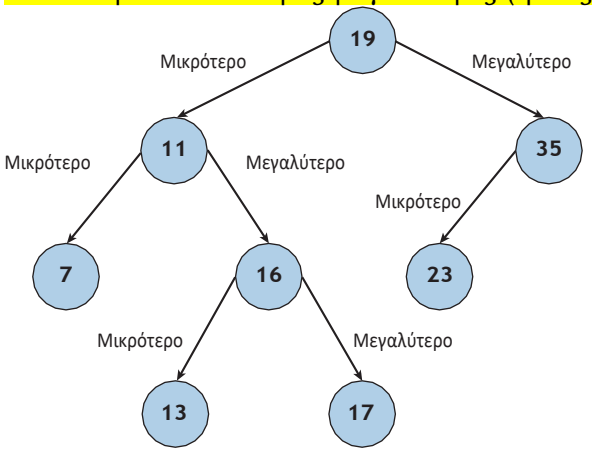
Τι είναι ένα δυαδικό δένδρο;

Ένα δυαδικό δένδρο αναζήτησης είναι ένα διατεταγμένο δένδρο, στο οποίο κάθε κόμβος έχει το πολύ δύο παιδιά, το αριστερό και το δεξί παιδί.

6. Τι είναι το δυαδικό δένδρο αναζήτησης;



Ένα δυαδικό δένδρο αναζήτησης, 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.

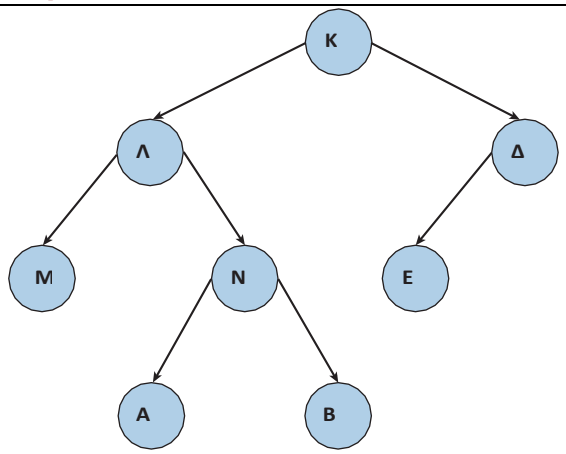


7. Ποια είναι τα πλεονεκτήματα των δυαδικών δένδρων αναζήτησης;

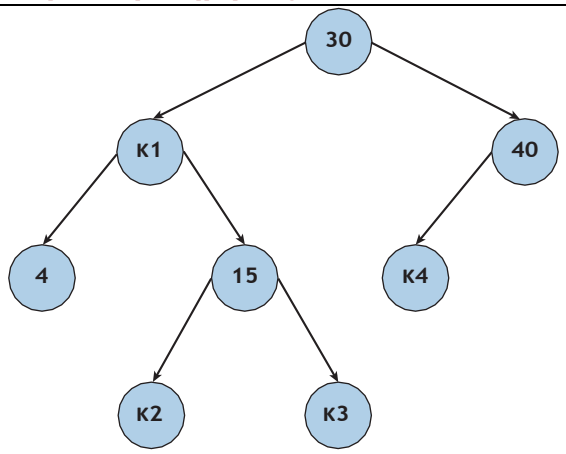
Τα δυαδικά δένδρα αναζήτησης συνδυάζουν τα εξής πλεονεκτήματα:

- ✓ Των λιστών, όσον αφορά τις πράξεις της εισαγωγής και της εξαγωγής, δηλαδή μπορούν να προσθέτουν και να αφαιρούν εύκολα κόμβους.
- ✓ Των ταξινομημένων πινάκων, όσον αφορά την πράξη της αναζήτησης, την οποίο εκτελεί ταχύτερα χάρη στον τρόπο αποθήκευσης τιμών.
  - Γενικά όταν θέλουμε να έχουμε γρήγορους αλγόριθμους αναζήτησης, πρέπει να αποθηκεύουμε τις τιμές στα δένδρα με ένα συγκεκριμένο τρόπο.

Παράδειγμα 1 - Δημιουργία δένδρου: Να δημιουργήσετε το δένδρο που προκύπτει από τις ακόλουθες πληροφορίες: **1)** ο κόμβος Κ έχει παιδιά τους κόμβους Λ και Δ **2)** ο κόμβος Μ έχει πατέρα τον κόμβο Λ **3)** ο κόμβος Ν είναι αδέλφια με τον κόμβο Μ έχει παιδιά τους κόμβους Α και Β **3)** ο κόμβος Ε έχει πατέρα τον κόμβο Δ.

Λύση	**Παρατηρήσεις**
	Για να συνδυάσουμε τις παραπάνω πληροφορίες θα πρέπει να προσέξουμε: ✓ Ό κόμβος-πατέρας βρίσκεται πάνω από τον κόμβο-παιδί και ενώνονται με ακμή. ✓ Όλοι οι κόμβοι-παιδιά ενός κόμβου-πατέρα βρίσκονται ένα επίπεδο κάτω από τον κόμβο πατέρα και στο ίδιο επίπεδο μεταξύ τους. ✓ Όταν δύο κόμβοι είναι αδέλφια έχουν τον ίδιο κόμβο-πατέρα. ✓ Όταν ένας κόμβος – παιδί έχει δικά του παιδιά, τότε αυτά βρίσκονται ένα επίπεδο κάτω. ✓ Η ρίζα του δένδρου είναι ο κόμβος που δεν έχει πατέρα.

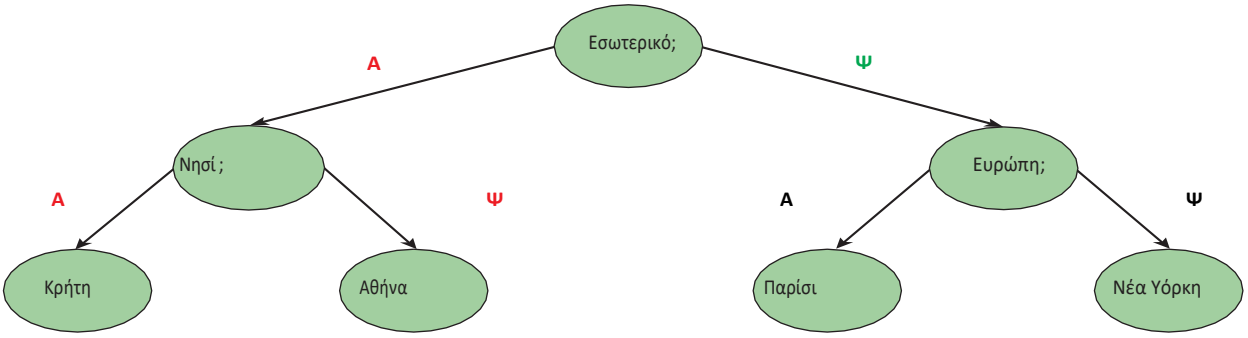
Παράδειγμα 2 – Δυαδικό δένδρο αναζήτησης: Να συμπληρώσετε τα κενά Κ1, Κ2,Κ3, Κ4 στο παρακάτω δένδρο με κατάλληλες τιμές, ώστε να προκύψει ένα δυαδικό δένδρο αναζήτησης.

Λύση - Παρατηρήσεις	
	Για να προκύψει δυαδικό δένδρο αναζήτησης θα πρέπει κάθε κόμβος, να έχει τιμή μεγαλύτερη από όλους τους κόμβους του αριστερού υποδένδρου του και τιμή μικρότερη από όλους τους κόμβους του δεξιού υποδένδρου του. Οπότε: **Κ1:** θα πρέπει να έχει τιμή μικρότερη από το 30 (το Κ1 βρίσκεται στο αριστερό υποδένδρο του 30), μεγαλύτερη από το 4 (το 4 βρίσκεται στο αριστερό υποδένδρο του Κ1) και μικρότερη από το 15 (το 15 βρίσκεται στο δεξί υποδένδρο του Κ1), **για παράδειγμα το 10**. **Κ2:** θα πρέπει να έχει τιμή μεγαλύτερη από το 10 που επιλέξαμε στο Κ1 (το Κ2 βρίσκεται στο δεξί υποδένδρο του Κ1) και τιμή μικρότερη από το 15 (το Κ2 βρίσκεται στο αριστερό υποδένδρο του 15), **για παράδειγμα το 14**. **Κ3:** θα πρέπει να έχει τιμή μεγαλύτερη από το 10 που επιλέξαμε στο Κ1 (το Κ3 βρίσκεται στο δεξί υποδένδρο του Κ1) και τιμή μεγαλύτερη από το 15 (το Κ3 βρίσκεται στο δεξί υποδένδρο του 15), **για παράδειγμα το 16**. **Κ4:** θα πρέπει να έχει τιμή μεγαλύτερη από το 30 (το Κ4 βρίσκεται στο δεξί υποδένδρο του 30) και τιμή μικρότερη από το 40 (το Κ4 βρίσκεται στο αριστερό υποδένδρο του 40), **για παράδειγμα το 35**

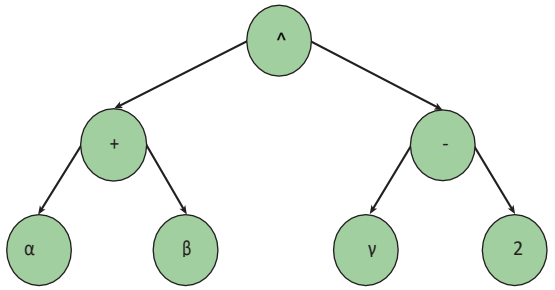
Παράδειγμα 3 – Δένδρο απόφασης: Να δημιουργήσετε ένα δένδρο απόφασης, που θα κατηγοριοποιεί τους προορισμούς: Ηράκλειο, Αθήνα, Παρίσι, Νέα Υόρκη σύμφωνα με τα χαρακτηριστικά: **1)** αν είναι

προορισμός εσωτερικού ή εξωτερικού **2)** στην περίπτωση του εσωτερικού αν είναι νησί ή όχι **3)** στην περίπτωση του εξωτερικού αν είναι Ευρώπη ή όχι.

Επεξήγηση λύσης: Σε ένα δένδρο απόφασης, κάθε κόμβος αποτελεί ένα χαρακτηριστικό, κάθε ακμή μία απόφαση (Ναι – Όχι, Αληθής - Ψευδής) και κάθε φύλο ένα χαρακτηριστικό. Στο παράδειγμά μας, αρχικά θα χρησιμοποιήσουμε ένα κόμβο για το ερώτημα να είναι προορισμός εσωτερικού η όχι, με δύο ακμές (μία για Αληθής και μία για Ψευδής). Στην περίπτωση Αληθής θα χρειαστούμε άλλο ένα κόμβο, για το ερώτημα αν είναι νησί ή όχι (ξανά με ακμές Αληθής - Ψευδής) και θα καταλήξουμε σε δύο φύλλα, ένα για Κρήτη και ένα για Ηράκλειο. Στην περίπτωση Ψευδής, θα χρειαστούμε άλλο ένα κόμβο, για το ερώτημα αν είναι Ευρώπη ή όχι (ξανά με ακμές Αληθής – Ψευδής) και θα καταλήξουμε σε δύο φύλα, ένα για Παρίσι και ένα για Νέα Υόρκη.



Παράδειγμα 4 – Δένδρο υπολογισμού αριθμητικών εκφράσεων: Να δημιουργήσετε ένα δένδρο το οποίο θα αναπαριστά την λύση της πράξης (α+β)^(γ-2).

Λύση	**Παρατηρήσεις**
	✓ Η πράξη (τελεστής) που θα εκτελεστεί τελευταία στην ιεραρχία θα τοποθετηθεί ως ρίζα του δένδρου. ✓ Οι πράξεις (τελεστές) που θα εκτελεστούν πρώτες, θα τοποθετηθούν ως παιδιά της ρίζας. ✓ Στα φύλα θα τοποθετήσουμε της μεταβλητές ή αριθμούς στους οποίους εφαρμόζονται οι πράξεις (τελεστές)

Ενότητα 1.3.3: Γράφοι

1. Να περιγράψετε τις γραμμικές και τις μη γραμμικές δομές δεδομένων.

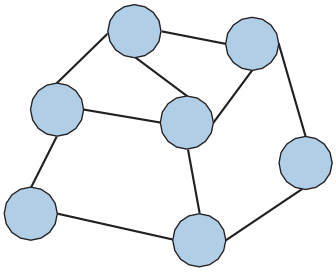
✓ **Γραμμικές δομές:** σε μία γραμμική δομή, μετά από κάθε στοιχείο ακολουθεί ένα άλλο στοιχείο, εκτός αν είναι τελευταίο. Παράδειγμα γραμμικής δομής είναι οι πίνακες και οι συνδεδεμένες λίστες.

✓ **Μη-Γραμμικές δομές:** Τα δεδομένα τους δεν ακολουθούν κάποια σειρά, όπως για παράδειγμα στ δένδρα, όπου ξεκινούν από έναν κόμβο-ρίζα και μπορεί να συνδεθεί με άλλους κόμβους, περισσότερους από έναν. Επίσης κάθε δένδρο μπορεί να περιέχει άλλα υποδένδρα.

2. Να δώσετε τον ορισμό του γράφου.

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

3. Να αναφέρετε κάποιες διαφορές των γράφων σε σχέση με το δένδρα.



- ✓ Ένα δένδρο μπορεί να ρέει μόνο προς μία κατεύθυνση, από τον κόμβο ρίζα σε κόμβους φύλλων ή κόμβους παιδιών. Αντίθετα, οι γράφοι δεν έχουν την έννοια ενός κόμβου ρίζας.
- ✓ Τα δένδρα μπορούν να έχουν μόνο μονόδρομες συνδέσεις, δηλαδή ένας κόμβος παιδιού μπορεί να έχει μόνο έναν γονέα. Επίσης ένα δένδρο δεν μπορεί να έχει βρόγχους ή κυκλικούς δεσμούς. Αντίθετα, οι γράφοι μπορούν να συνδεθούν με οποιονδήποτε τρόπο, για παράδειγμα ένας κόμβος μπορεί να συνδεθεί με άλλους πέντε. Επίσης, οι γράφοι δεν έχουν «μονοκατευθυντική» ροή, αντ αυτού μπορεί να έχουν κατεύθυνση ή να μην έχουν καμία κατεύθυνση.
- ✓ Το δένδρα είναι περιορισμένοι τύποι γράφων. Ένα δένδρο θα είναι πάντα γράφος, ενώ ένας γράφος δεν είναι πάντα δένδρο.
- ✓ Τα δένδρα, γενικά διέπονται από κανόνες, όπως για παράδειγμα στα δυαδικά δένδρα που κάθε κόμβος μπορεί να έχει συνδέσεις μέχρι δύο κόμβους, ενώ οι γράφοι δεν έχουν αυτούς τους περιορισμούς και είναι η πιο γενική δομή δεδομένων.

4. Τι ονομάζουμε κατευθυνόμενες ακμές και τι μη κατευθυνόμενη ακμή;

- ✓ Κατευθυνόμενη ακμή: είναι η ακμή που έχει κατεύθυνση. Στην περίπτωση αυτή οι κόμβοι συνδέονται σε πολύ συγκεκριμένο τρόπο.
- ✓ Μη κατευθυνόμενη ακμή: είναι η ακμή που δεν έχει κατεύθυνση. Στην περίπτωση αυτή η σύνδεση μεταξύ των κόμβων είναι πιο γενική.

A ⟶ B	**A ——— B**
✓ Στο συγκεκριμένο παράδειγμα, η ακμή είναι κατευθυνόμενη. ✓ Ο κόμβος Α ονομάζεται προέλευση και ο κόμβος Β προορισμός. ✓ Μπορούμε να «ταξιδέψουμε» μόνο από την πηγή στον προορισμό, δηλαδή από τον κόμβο Α στον κόμβο Β.	✓ Αντίθετα, εδώ η ακμή είναι μη – κατευθυνόμενη. ✓ Οι κόμβοι προέλευσης και προορισμού δεν είναι σταθεροί. ✓ Μπορούμε να «ταξιδέψουμε» και προς τις δύο κατευθύνσεις, δηλαδή η διαδρομή είναι αμφίδρομη.

5. Τι ονομάζουμε κατευθυνόμενο και τι μη- κατευθυνόμενο γράφο;

- ✓ Εάν όλες οι ακμές σε έναν γράφο έχουν κατεύθυνση, ο γράφος ονομάζεται **κατευθυνόμενος γράφος (directed graph)**.
- ✓ Εάν όλες οι ακμές σε έναν γράφο δεν έχουν κατεύθυνση, ο γράφος ονομάζεται **μη κατευθυνόμενος γράφος (undirected graph).**

6. Να δώσετε διάφορα παραδείγματα γράφων.

- ✓ **Παγκόσμιος Ιστός (WWW):** Είναι ένας τεράστιος τύπος γράφου, όπου κάποιοι κόμβοι έχουν μη κατευθυνόμενες ακμές (δηλαδή μπορούμε να πάμε από την ιστοσελίδα Α στην ιστοσελίδα Β και το αντίστροφο) και κάποιοι κατευθυνόμενες ακμές (μπορούμε να πάμε μόνο από την ιστοσελίδα Α στην ιστοσελίδα Β και όχι το αντίστροφο).
- ✓ **Facebook:** Είναι τύπος γράφου, με αμφίδρομη σχέση κόμβων, δηλαδή μη κατευθυνόμενες ακμές, καθώς αν κάποιος χρήστης είναι φίλος με κάποιον άλλο, ισχύει και το αντίστροφο.

Twitter: Είναι τύπος γράφου, που γενικά υπάρχουν κατευθυνόμενες ακμές, καθώς αν ο χρήστης Α ακολουθεί τον χρήστη Β, δεν είναι απαραίτητο να ισχύει και το αντίθετο

Ενότητα 2. Τεχνικές Σχεδίασης Αλγορίθμων

2.1 Μέθοδος Διαίρει και Βασίλευε

1. Να περιγράψετε τη μέθοδο Διαίρει και Βασίλευε.

Η **«Διαίρει και Βασίλευε»** (divide and conquer) αποτελεί μια μέθοδο σχεδίασης αλγορίθμων στην οποία εντάσσονται οι τεχνικές που υποδιαιρούν ένα πρόβλημα σε μικρότερα υποπροβλήματα, που έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα, αλλά είναι μικρότερα σε μέγεθος. Με όμοιο τρόπο, τα υποπροβλήματα αυτά μπορούν να διαιρεθούν σε ακόμη μικρότερα υποπροβλήματα κ.ο.κ. Έτσι η επίλυση ενός προβλήματος έγκειται στη σταδιακή επίλυση των όσο το δυνατόν μικρότερων υποπροβλημάτων, ώστε τελικά να προκύψει η συνολική λύση του αρχικού ευρύτερου προβλήματος.
Η προσέγγιση αυτή ονομάζεται «από πάνω προς τα κάτω» (top-down).

2. Με ποια βήματα μπορεί να αποδοθεί η μέθοδος Διαίρει και Βασίλευε;

1. Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.
2. Το στιγμιότυπο του προβλήματος υποδιαιρείται 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.

Ενότητα 3. Επιλογή και επανάληψη

Γενική μορφή της εντολής ΕΠΙΛΕΞΕ

ΕΠΙΛΕΞΕ <έκφραση>
ΠΕΡΙΠΤΩΣΗ <λίστα_τιμών_1>
<εντολές_1>
ΠΕΡΙΠΤΩΣΗ <λίστα_τιμών_2>
<εντολές_2>
.....
ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
<εντολές_αλλιώς>
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

Όπου:

• **<έκφραση> :**

είναι μια μεταβλητή, η τιμή της οποίας θα ελεγχθεί με τις τιμές που δίνονται στις ΠΕΡΙΠΤΩΣΕΙΣ και ανάλογα σε ποια ΠΕΡΙΠΤΩΣΗ ανήκει θα εκτελεστούν οι αντίστοιχες εντολές **ή** η πράξη, που υπολογίζει την τιμή της.

Δηλαδή, η **<έκφραση>** μπορεί να είναι:

- Μεταβλητή
- Αριθμητική πράξη
- Συγκριτική πράξη

• **<λίστα_τιμών_Ν>:**

οι τιμές που μπορεί να πάρει μια έκφραση. Οι τιμές αυτές μπορεί να είναι διακριτές τιμές, περιοχή τιμών από…έως ή να υπακούν σε μια συνθήκη.

Τρόπος εκτέλεσης

Κατά την εκτέλεση της εντολής υπολογίζεται η τιμή της έκφρασης και στη συνέχεια εκτελούνται οι εντολές που ανήκουν στην αντίστοιχη περίπτωση τιμών. Στην περίπτωση που η τιμή έκφρασης δεν αντιστοιχεί σε καμία περίπτωση, τότε εκτελούνται οι εντολές της ΠΕΡΙΠΤΩΣΗΣ_ΑΛΛΙΩΣ. Η ΠΕΡΙΠΤΩΣΗ_ΑΛΛΙΩΣ είναι προαιρετική. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί μετά το ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ.

Ενότητα 4: Αντικειμενοστραφής προγραμματισμός

1. Ποια είναι η βασική φιλοσοφία του αντικειμενοστραφούς προγραμματισμού;

- ✓ Σύμφωνα με την αντικειμενοστραφή θεωρία ανάπτυξης εφαρμογών, η προσέγγιση κάθε προβλήματος πρέπει να γίνεται με φυσική ερμηνεία και να μη στηρίζεται σε πολύπλοκα τεχνικά ζητήματα.
- ✓ Αυτή η αντίληψη, ότι δηλαδή η επίλυση ενός προβλήματος επιτυγχάνεται με τη σύνθεση ικανοτήτων (ο τρόπος υλοποίησης των οποίων μας είναι άγνωστος) που διαθέτουν διαφορετικές ανεξάρτητες οντότητες, οι οποίες αλληλεπιδρούν για τον σκοπό αυτό, βρίσκεται στην καρδιά της αντικειμενοστραφούς προσέγγισης.
- ✓ Για παράδειγμα, για το χτίσιμο ενός σπιτιού, συνεργάζονται αρχιτέκτονες, μηχανικοί, υδραυλικοί, οικοδόμοι κτλ. Η κάθε ειδικότητα δεν γνωρίζει τις λεπτομέρειες για το πώς υλοποιείται η εργασία των άλλων ιδιοτήτων, όμως συνεργάζονται μεταξύ τους, άμεσα η έμμεσα, ώστε να λύσουν το αρχικό πρόβλημα.
- ✓ Στηρίζεται 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.

2. Να δώσετε τους ορισμούς για τις ακόλουθες έννοιες: Αντικειμενοστραφής προγραμματισμός, αντικείμενα, ιδιότητες, μέθοδοι.

- ✓ **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. 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) ενώ οι ενέργειες καθορίζουν τη συμπεριφορά του. Οι ενέργειες στον αντικειμενοστραφή προγραμματισμό αναφέρονται και ως **μέθοδοι** (methods).

3. Ποια είναι τα βασικά συστατικά που πρέπει να καταγράψουμε για την επίλυση ενός προβλήματος με βάση τον αντικειμενοστραφή προγραμματισμό;

- ✓ τα **αντικείμενα** που συμμετέχουν με βάση τον ρόλο τους στο συγκεκριμένο σενάριο
- ✓ οι **ιδιότητες** κάθε αντικειμένου, δηλ. τα σχετικά με το συγκεκριμένο πρόβλημα χαρακτηριστικά του
- ✓ 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.

4. Μεθοδολογία Εύρεσης αντικειμένων, ιδιοτήτων, ενεργειών / υπηρεσιών και ειδών συνεργασίας.

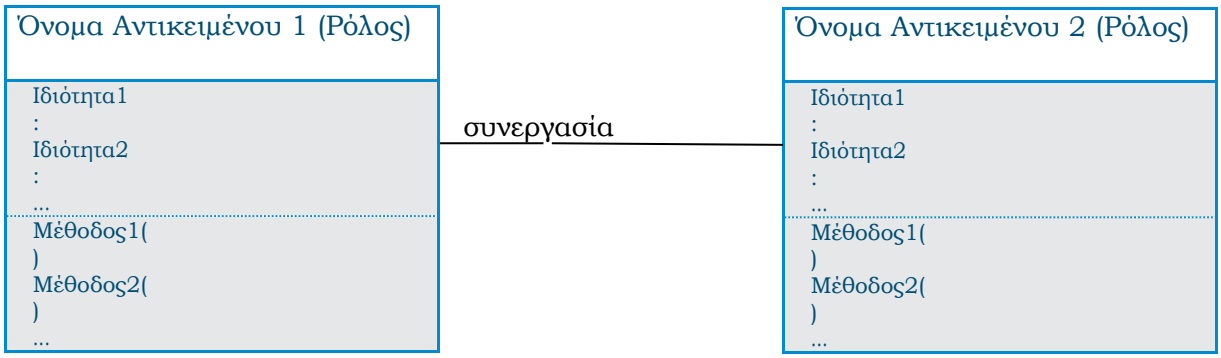
«Η μητέρα σας, η οποία ήταν αδύνατο να στείλει αυτοπροσώπως λουλούδια στη φίλη της την Άννα, η οποία ζει στη Ρώμη, έπρεπε να επισκεφθεί το ανθοπωλείο της γειτονιάς, να δώσει τα προσωπικά της στοιχεία (όνομα, επώνυμο, διεύθυνση, τηλέφωνο, email), να περιγράψει στον κ. Γιώργο τον ανθοπώλη το είδος της ανθοδέσμης που ήθελε να στείλει, να δώσει τη διεύθυνση και τα στοιχεία επικοινωνίας της Άννας στη Ρώμη. Ο κ. Γιώργος, ο οποίος έπρεπε να στείλει ένα μήνυμα στο δίκτυο ανθοπωλών που συμμετείχε για να εντοπίσει συνεργαζόμενο ανθοπωλείο στη Ρώμη και να αποστείλει τα στοιχεία της παραγγελίας της μαμάς, ο τοπικός ανθοπώλης κ. Τζιοβάνι να αποδεχτεί τη συνεργασία, να αναθέσει στον ανθοδέτη του κ. Αντόνιο να φτιάξει την ανθοδέσμη και στη συνέχεια να καλέσει τον ταχυμεταφορέα κ. Πέπε να παραδώσει την ανθοδέσμη στην κα Άννα.» Με βάση το παραπάνω σενάριο, να εντοπίσετε τα αντικείμενα, τις ιδιότητες, τις ενέργειες / υπηρεσίες και τα είδης συνεργασίας.

- ✓ **Αντικείμενα:** Οι οντότητες που συμμετέχουν στο σενάριο μας είναι: Μαμά (Πελάτης), Γιώργος (Ανθοπώλης), Τζιοβάνι (Ανθοπώλης), Αντώνιο (Ανθοδέτης), Πέπε (Ταχυμεταφορέας), Άννα (Πελάτης).
  - Στα αντικείμενα προσδιορίσαμε και τον ρόλο τους, κάτι που θα μας βοηθήσει στην δημιουργία κλάσεων στη συνέχεια.
- ✓ **Ιδιότητες:** Τα χαρακτηριστικά κάθε αντικειμένου, δηλαδή οι πληροφορίες που πρέπει να γνωρίζουμε για αυτό είναι (στο παράδειγμα μας δεν αναφέρονται αναλυτικά, θεωρούμε τα παρακάτω):
  - **Μαμά (Πελάτης):** όνομα, επώνυμο, διεύθυνση, τηλέφωνο, email
  - **Άννα (Πελάτης):** όνομα, επώνυμο, διεύθυνση, τηλέφωνο, email
  - **Γιώργος (Ανθοπώλης):** επωνυμία εταιρίας, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, τραπεζικός λογαριασμός, κωδικός δικτύου συνεργασίας.
  - **Τζιοβάνι (Ανθοπώλης):** επωνυμία εταιρίας, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, τραπεζικός λογαριασμός, κωδικός δικτύου συνεργασίας.
  - **Αντόνιο (Ανθοδέτης):** επωνυμία εταιρίας, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, ειδικότητα, ωριαία αμοιβή.
  - **Πέπε (Ανθοδέτης):** επωνυμία εταιρίας, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, τύπος.
- ✓ **Ενέργειες / υπηρεσίες:** Θα καταγράψουμε τις ενέργειες ή τις υπηρεσίες (μέθοδοι) που πραγματοποιεί κάθε αντικείμενο. Τα ονόματα των μεθόδων παρουσιάζονται με βάση την ονοματοδοσία των υποπρογραμμάτων (δηλαδή όνομα_μεθόδου()).
  - **Μαμά (Πελάτης):** ΚάνειΠαραγγελία()
  - **Άννα (Πελάτης):** ΠαραλαμβάνειΑνθοδέσμη()
  - **Γιώργος (Ανθοπώλης):** ΔέχεταιΠαραγγελία(), ΖητάΣυνεργασία()
  - **Τζιοβάνι (Ανθοπώλης):** ΑποδέχεταιΣυνεργασία(), ΑναθέτειΑνθοσεσία(), ΑναθέτειΠαράδοση()
  - **Αντόνιο (Ανθοδέτης):** ΕτοιμάζειΑνθοδέσμη()
  - **Πέπε (Ανθοδέτης):** ΠαραδίδειΑνθοδέσμη()
- ✓ **Είδος Συνεργασίας:** θα καταγράψουμε τον τρόπο με τον οποίο συνεργάζονται τα αντικείμενα μεταξύ τους
  - **Παραγγελία:** Μαμά (Πελάτης) – Γιώργος (Ανθοπώλης)
  - **Συνεργασία:** Γιώργος (Ανθοπώλης) – Τζιοβάνι (Ανθοπώλης)
  - **Ανάθεση ανθοδεσίας:** Τζιοβάνι (Ανθοπώλης) – Αντόνιο (Ανθοδέτης)
  - **Ανάθεση παράδοσης:** Τζιοβάνι (Ανθοπώλης) – Πέπε (Ταχυμεταφορέας)
  - **Παράδοση:** Πέπε (Ταχυμεταφορέας) – Άννα (Πελάτης)

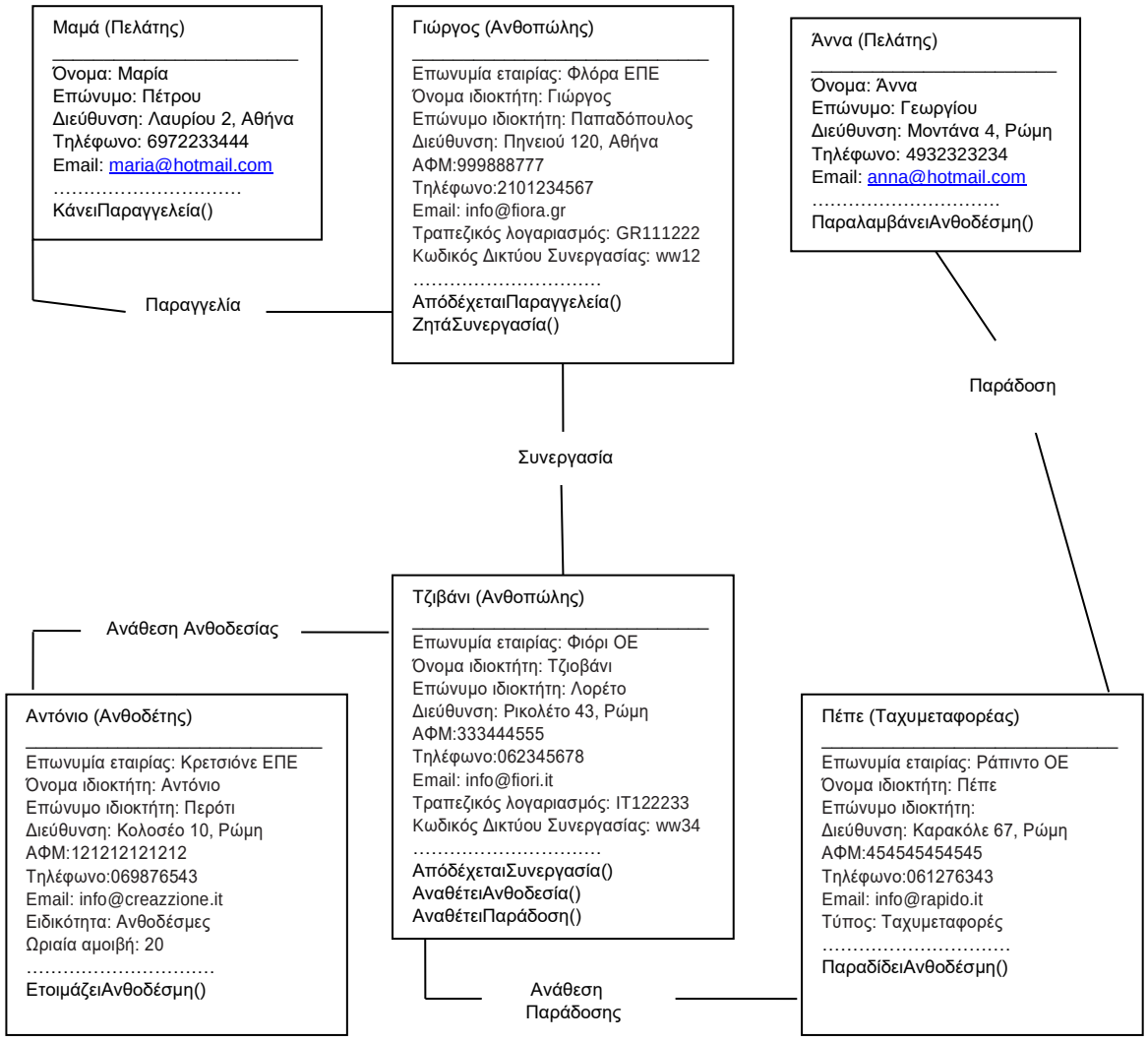
5. Διαγραμματική Αναπαράσταση των συστατικών ενός προβλήματος.

Αφού εντοπίσαμε τα συστατικά επίλυσης του προβλήματος, μπορούμε να τα αναπαραστήσουμε διαγραμματικά, ώστε να έχουμε μία εικόνα των συνεργαζόμενων οντοτήτων. Αποτελεί ουσιαστικά το σχέδιο επίλυσης του με βάση την αντικειμενοστραφής προσέγγιση. Χρησιμοποιούμε:

- ✓ **Παραλληλόγραμμα**, για την αποτύπωση των αντικειμένων, των ιδιοτήτων και των μεθόδων τους.
- ✓ **Γραμμές σύνδεσης,** για την περιγραφή του είδους συνεργασίας τους



Ολοκληρωμένο το παραπάνω παράδειγμα αποστολής λουλουδιών:



6. Πως 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

7. Τι ονομάζουμε ενθυλάκωση;

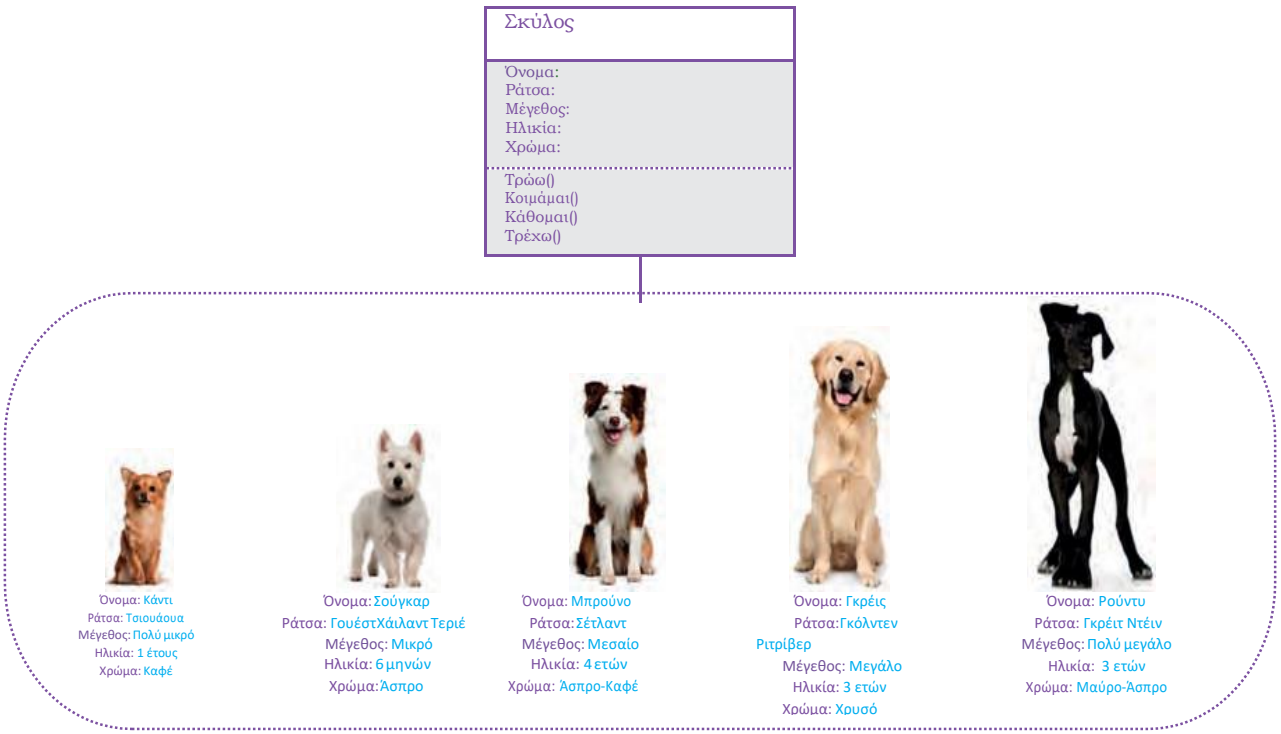
Σε μια αντικειμενοστραφή εφαρμογή κάθε αντικείμενο αποτελεί ξεχωριστή οντότητα και περιέχει ενσωματωμένες τις ιδιότητες (δεδομένα) και τους κανόνες συμπεριφοράς του (μεθόδους). Η δυνατότητα ενός αντικειμένου να συνδυάζει εσωτερικά τα δεδομένα και τις μεθόδους χειρισμού του καλείται **ενθυλάκωση** (encapsulation).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

8. Τι ονομάζουμε κλάση;

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.

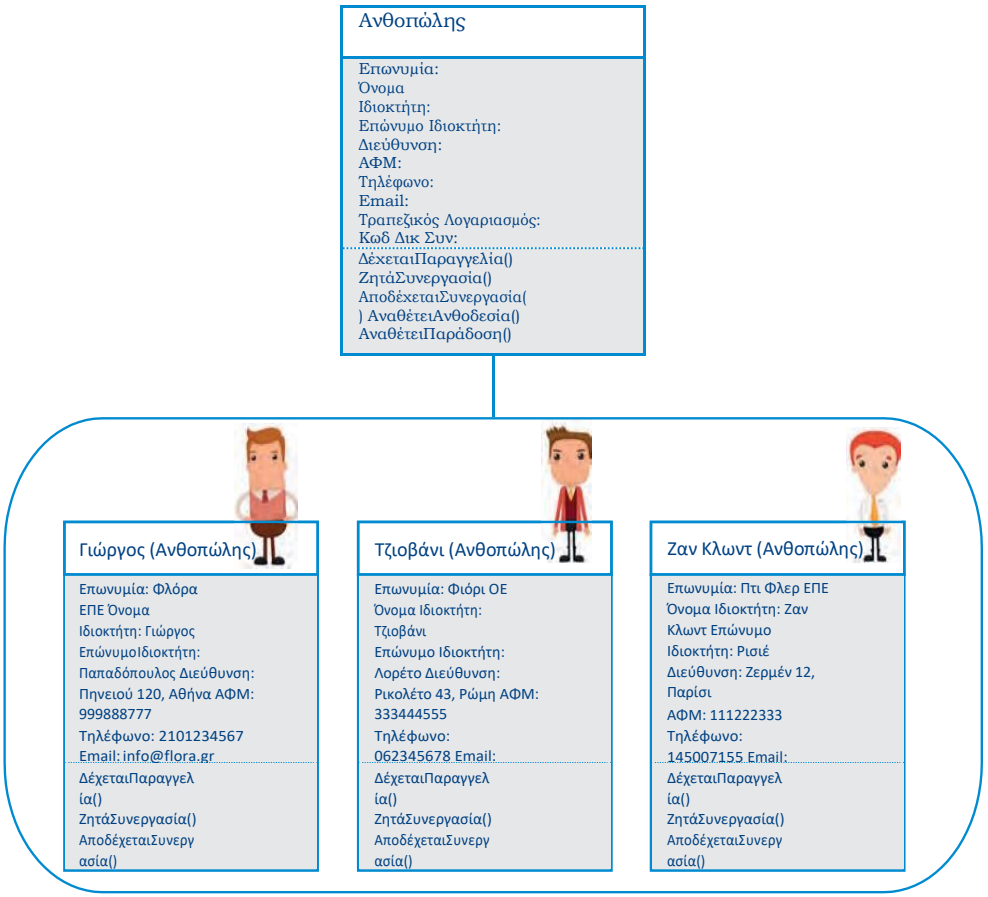
9. Δώστε παραδείγματα κλάσεων.

Ένα παράδειγμα από τον πραγματικό κόσμο είναι τα αυτοκίνητα. Όλα τα αυτοκίνητα ενός συγκεκριμένου μοντέλου παράγονται με βάση το ίδιο σχέδιο που καθορίζει τις προδιαγραφές του οχήματος, π.χ. διαστάσεις αμαξώματος, διαστάσεις τροχών, κυβισμός, είδος κιβωτίου ταχυτήτων, είδος καυσίμου, χρώμα αμαξώματος, επένδυση καθισμάτων, κ.λπ. Με βάση αυτό το κοινό σχέδιο παράγονται από το εργοστάσιο πολλά διαφορετικά οχήματα του ίδιου μοντέλου. Κάθε όχημα διαφοροποιείται από τα υπόλοιπα στις τιμές κάποιων ιδιοτήτων. Ακόμα όμως και αν παραχθούν δύο οχήματα με τις ίδιες ακριβώς τιμές για τις ιδιότητές τους (κάτι που είναι συνηθισμένο), τα οχήματα συνεχίζουν να αποτελούν διαφορετικές οντότητες. Μπορούμε λοιπόν να θεωρήσουμε το σχέδιο του συγκεκριμένου μοντέλου αυτοκινήτου ως κλάση και τα οχήματα που κατασκευάζονται με βάση το σχέδιο ως αντικείμενα της κλάσης. Ένα δεύτερο παράδειγμα, το οποίο απεικονίζεται διαγραμματικά παρακάτω, είναι η κλάση «Σκύλος» και τα αντικείμενα της.



Να δημιουργήσετε την κλάση «Ανθοπώλης» για το πρόβλημα αποστολή λουλουδιών του ερωτήματος 4.

Ο «Γιώργος», ο «Τζιοβάνι» και το νέο μέλος από το Παρίσι, ο Ζαν Κλωντ, είναι όλα αντικείμενα της κλάσης «Ανθοπώλης», άρα έχουν τις ίδιες ιδιότητες και μεθόδους. Αυτή είναι μία πλήρης απεικόνιση των αντικειμένων, καθώς στην διαγραμματική απεικόνιση που είχαμε κάνει στο ερώτημα 4, ο «Γιώργος» και ο «Τζιοβάνι» είχαν διαφορετικές μεθόδους. Όμως θα μπορούσε και κάποιος από τη Ρώμη να στείλει λουλούδια στην Αθήνα, οπότε οι ρόλοι θα αντιστρεφόταν.

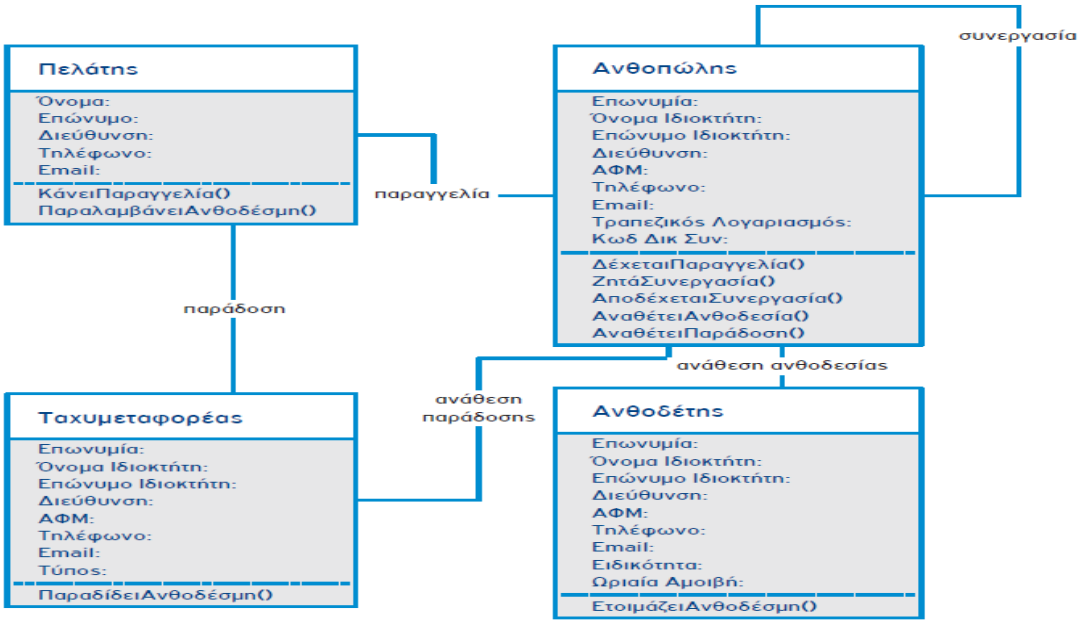


10. Να παρουσιάσετε μία διαγραμματική απεικόνιση των κλάσεων του προβλήματος αποστολής λουλουδιών του ερωτήματος 4.

Αναλύοντας το πρόβλημα, προκύπτουν οι ακόλουθες κλάσεις:

- ✓ **Ανθοπώλης:** με τις ιδιότητες και μεθόδους που αναλύσαμε στο προηγούμενο ερώτημα. Θα αναπαραστήσουμε την συνεργασία μεταξύ δύο αντικειμένων της ίδιας κλάσης, με ένα βέλος (μέθοδος) που θα αρχίζει και θα επιστρέφει στην ίδια την κλάση.
- ✓ **Πελάτης:** με ιδιότητες όνομα, επώνυμο, διεύθυνση, τηλέφωνο, email και μεθόδους ΚάνειΠαραγελία() και ΠαραλαμβάνειΑνθοδέση().
- ✓ **Ταχυμεταφορέας:** Με ιδιότητες επωνυμία, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, τύπος και μεθόδους ΠαραδίδειΑνθοδέσμη().
- ✓ **Ανθοδέτης:** Με ιδιότητες επωνυμία, όνομα ιδιοκτήτη, επώνυμο ιδιοκτήτη, διεύθυνση, ΑΦΜ, τηλέφωνο, email, ειδικότητα, ωριαία αμοιβή και μεθόδους ΠαραδίδειΑνθοδέσμη().

Η διαγραμματική απεικόνιση κλάσεων είναι η ακόλουθη:

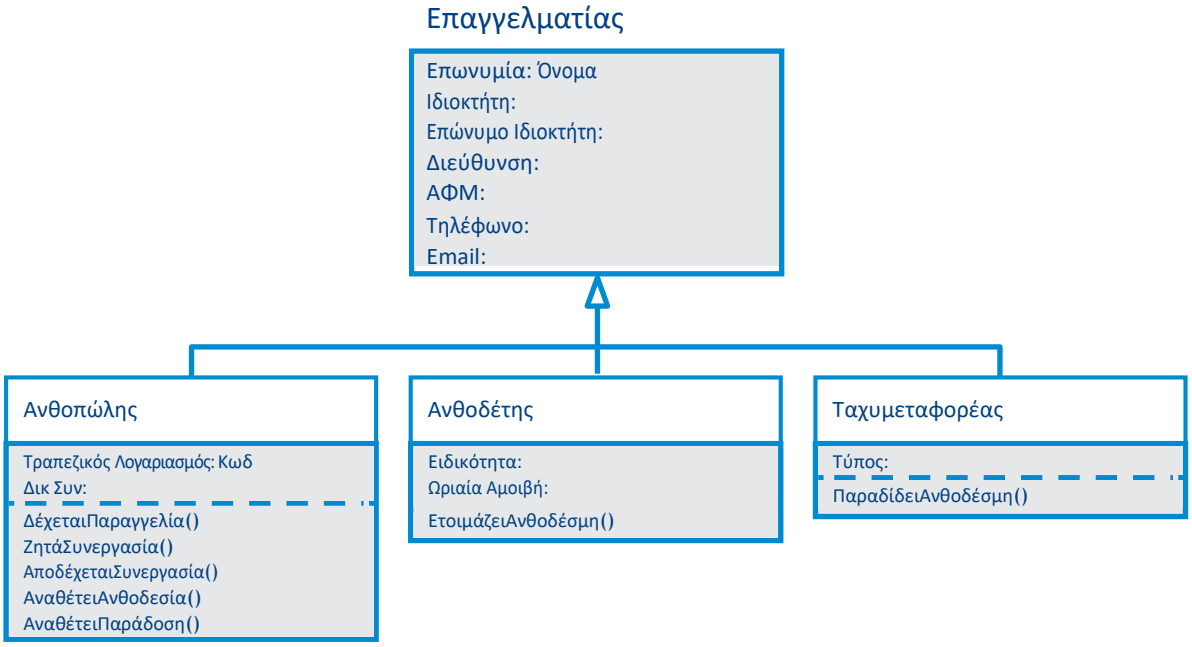


11. Τι είναι η κληρονομικότητα;

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).

12. Να δημιουργήσετε σχέσεις κληρονομικότητας στο παράδειγμα 4 με το πρόβλημα αποστολής λουλουδιών.

- ✓ Μπορούμε να δημιουργήσουμε μία κλάση «Επαγγελματίας», η οποία διαθέτει τις κοινές ιδιότητες των κλάσεων «Ανθοπώλης», «Ανθοδέτης» και «Ταχυμεταφορέας».
- ✓ Σε μια σχέση κληρονομικότητας, η κλάση-πρόγονος περιλαμβάνει τις κοινές ιδιότητες και μεθόδους όλων των κλάσεων-απογόνων της, ενώ οι κλάσεις-απόγονοι εμφανίζουν μόνο τις διαφορετικές τους ιδιότητες και μεθόδους αφού τις κοινές τις κληρονομούν από τη «μητέρα» τους.
- ✓ Η διαγραμματική αναπαράσταση της σχέσης κληρονομικότητας που μόλις περιγράψαμε γίνεται με τη βοήθεια του ειδικού συμβόλου γενίκευσης .
- ✓ Οι ιδιότητες της κλάσης «Επαγγελματίας» δεν επαναλαμβάνονται στην κλάση «Ανθοπώλης», όμως το αντικείμενο «Γιώργος» θα έχει τις ιδιότητες και των δύο.



13. Να εξηγήσετε πότε μία κλάση Α μπορεί να είναι έγκυρη υποκλάση της Β.

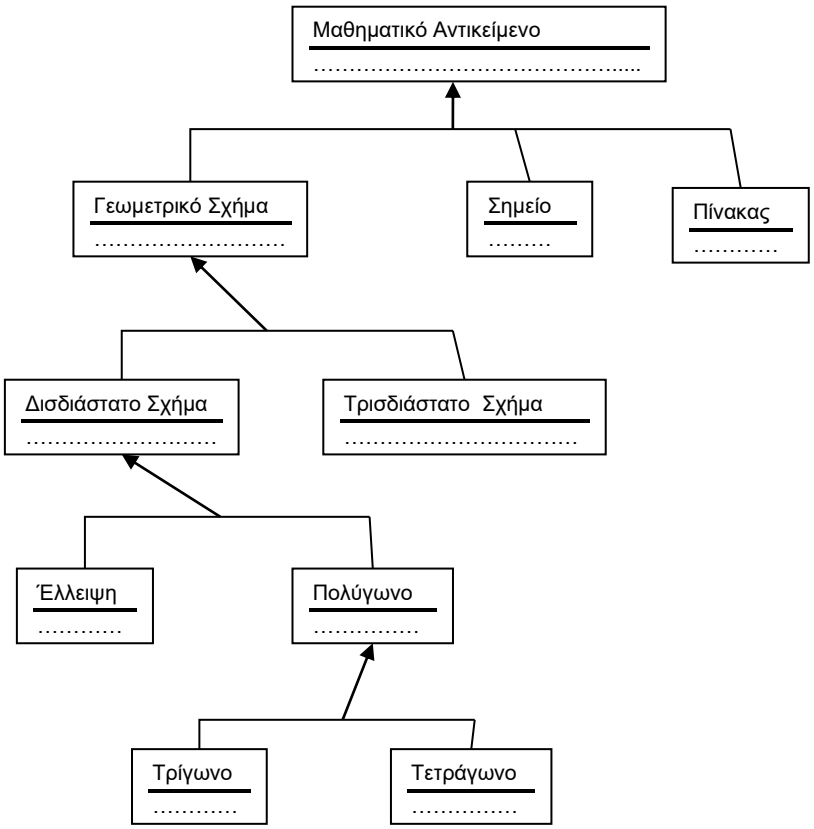
Μία κλάση Α μπορεί αν είναι έγκυρη υποκλάση της Β όταν ισχύει ο κανόνας «το Α είναι ένα (is_a) Β». Για παράδειγμα:

- ✓ «Το αυτοκίνητο είναι μέσο μεταφοράς»: ισχύει ο κανόνας, «Αυτοκίνητο is_a Μέσο μεταφοράς» οπότε η κλάση «Αυτοκίνητο» θα μπορούσε να είναι υποκλάση της κλάσης «Μέσο μεταφοράς».
- ✓ «Ο άνθρωπος είναι μέσο μεταφοράς»: δεν ισχύει ο κανόνας, «Άνθρωπος is_a Μέσο μεταφοράς» οπότε η κλάση «Άνθρωπος» δεν θα μπορούσε να είναι υποκλάση της κλάσης «Μέσο μεταφοράς».

14. Να οργανώσετε σε ιεραρχία κλάσεων με σχέση κληρονομικότητας τις κλάσεις: Σημείο, Πολύγωνο, Πίνακας, Κύκλος, Δισδιάστατο σχήμα, Τετράγωνο, Κύλινδρος, Μαθηματικό αντικείμενο, Σφαίρα, Τρίγωνο.

- ✓ Θα εντοπίσουμε τα επίπεδα ιεραρχίας από το πιο γενικό στο πιο ειδικό και θα κατατάξουμε κάθε κλάση στο αντίστοιχο επίπεδο ορίζοντας τις κατάλληλες σχέσεις κληρονομικότητας.
- ✓ Η πιο γενική έννοια είναι το «Μαθηματικό αντικείμενο», άρα θα είναι η υπερκλάση ανωτέρου επιπέδου.
- ✓ Κατεβαίνοντας ένα επίπεδο, μαθηματικά αντικείμενα είναι ο «Πίνακας», το «Σημείο» και το «Δισδιάστατο σχήμα».
- ✓ Δισδιάστατα σχήματα είναι το «Πολύγωνο», ο «Κύκλος», το «Τρίγωνο» και το «Τετράγωνο».
  - Το «Τετράγωνο» και το «Τρίγωνο» είναι πολύγωνα.
- ✓ Παρατηρούμε πως ο «Κύλινδρος» και η «Σφαίρα» είναι γεωμετρικά σχήματα αλλά όχι δισδιάστατα. Μία λύση είναι να εισάγουμε δύο κλάσεις:
  - «Τρισδιάστατο σχήμα», που θα είναι υπερκλάση των κλάσεων «Κύλινδρος» και «Σφαίρα».
  - «Γεωμετρικό σχήμα», που θα είναι υπερκλάση των κλάσεων «Δισδιάστατο σχήμα » και «Τρισδιάστατο σχήμα».

Με βάση την παραπάνω ανάλυση καταλήγουμε στο ακόλουθο διάγραμμα:



15. Τι είναι ο πολυμορφισμος;

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

- ✓ Ο πολυμορφισμός μας επιτρέπει να επαναπροσδιορίσουμε τον τρόπο με τον οποίο λειτουργούν κάποια πράγματα, είτε αλλάζοντας τον τρόπο λειτουργίας τους είτε αλλάζοντας τα εργαλεία τα οποία χρησιμοποιούνται για την επίτευξη του στόχου.
- ✓ Μια μορφή πολυμορφισμού έχουμε όταν, αντί για το αυτόματο σύστημα κεντρικού κλειδώματος, χρησιμοποιήσουμε τα κλειδιά για να κλειδώσουμε το αυτοκίνητό μας. Και στις δύο περιπτώσεις το αποτέλεσμα είναι το ίδιο. Με βάση όμως το εργαλείο που ενεργοποιούμε κάθε φορά αλλάζει και ο τρόπος υλοποίησης της λειτουργίας.

16. Παράδειγμα πολυμορφισμού: αριθμητικός τελεστής «+».

Σε κάποιες γλώσσες προγραμματισμού, ο τελεστής «+» δεν χρησιμοποιείται για αριθμητικά δεδομένα, αλλά διαφοροποιείται η λειτουργία του ες εξής:

- ✓ Αν δοθούν δύο αριθμοί, για παράδειγμα 60 και 20, τότε θα τους προσθέσει και θα βγάλει ως αποτέλεσμα το 60.
- ✓ Αν δοθούν δύο συμβολοσειρές, για παράδειγμα «Γεια» και «σας», ο τελεστής «+» θα προσαρμοστεί στα νέα δεδομένα και θα εκτελέσει μία πράξη που έχει νόημα στο χώρο των συμβολοσειρών, αυτή της «συνένωσης αλφαριθμητικών» και θα δώσει ως αποτέλεσμα την συμβολοσειρά «Γεια σας».
- ✓ Αν δοθούν η συμβολοσειρά «Καλώς ήρθες» και ο αριθμός 2020, ο τελεστής «+» θα αλλάξει ξανά συμβολοσειρά, θα μετατρέψει τον αριθμό 2020 στην συμβολοσειρά «2020», θα συνενώσει τις δύο συμβολοσειρές και θα δώσει ως αποτέλεσμα την συμβολοσειρά «Καλώς ήρθες 2020».
- ✓ Αν υποστηριζόταν από την γλώσσα προγραμματισμού, θα μπορούσαμε να δημιουργήσουμε μία συνάρτηση με το όνομα «Πρόσθεση» και συμπεριφορά ανάλογη με τα δεδομένα εισόδου:

Συνάρτηση Πρόσθεση (a, b) ΑΡΧΗ … ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ	
Είσοδος	**Έξοδος**
Πρόσθεση(20,40)	60
Πρόσθεση(«Γεια », «σας!»)	«Γεια σας!»
Πρόσθεση(«Καλώς ήρθες », 2020)	«Καλώς ήρθες 2020»

- ✓ Τρεις είναι οι γενικές περιπτώσεις κατά τις οποίες αναγκάζεται η συνάρτηση «Πρόσθεση» να προσαρμοστεί και να αλλάξει συμπεριφορά σύμφωνα με τον τύπο των παραμέτρων της: **1)** αριθμοί **2)** συμβολοσειρές **3)** αριθμοί και συμβολοσειρές.

17. Παράδειγμα υλοποίησης μεθόδου.

Έστω πως έχουμε τρία γεωμετρικά σχήματα σε κλάσεις, «Τρίγωνο», «Παραλληλόγραμμο» και «Κύκλος». Και οι τρεις κλάσεις έχουν την ίδια μέθοδο «ΥπολογισμόςΕμβαδού()», ο υπολογισμός όμως γίνεται από διαφορετικό μαθηματικό τύπο, επομένως η λειτουργία είναι πολυμορφική.

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

Οι μέθοδοι είναι στην ουσία υποπρογράμματα, όπως αυτά που έχετε ήδη μάθετε. Μόνο που ενώ οι γνωστές σας διαδικασίες και συναρτήσεις εντάσσονταν απευθείας στο κύριο πρόγραμμα, η κάθε αντικειμενοστραφής μέθοδος εντάσσεται σε μια κλάση και «περιορίζεται» στα δεδομένα που αυτή περιέχει. Αν θεωρήσουμε λοιπόν ότι η γλώσσα προγραμματισμού υποστηρίζει το παραπάνω μοντέλο, ο κώδικας υλοποίησης της πολυμορφικής μεθόδου «ΥπολογισμόςΕμβαδού()» για κάθε ένα από τα τρία γεωμετρικά σχήματα του παραδείγματός μας παρουσιάζεται παρακάτω:

Τρίγωνο	**Παραλληλόγραμμο**	**Κύκλος**
ΣΥΝΑΡΤΗΣΗ ΥπολογισμόςΕμβαδού(): **ΠΡΑΓΜΑΤΙΚΗ** **ΜΕΤΑΒΛΗΤΕΣ** **ΠΡΑΓΜΑΤΙΚΕΣ:**Εμ **ΑΡΧΗ** Εμ <- Βάση*Υψος/2 ΥπολογισμόςΕμβαδού <- Εμ **ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ**	**ΣΥΝΑΡΤΗΣΗ** ΥπολογισμόςΕμβαδού(): **ΠΡΑΓΜΑΤΙΚΗ** **ΜΕΤΑΒΛΗΤΕΣ** **ΠΡΑΓΜΑΤΙΚΕΣ:**Εμ **ΑΡΧΗ** Εμ <- Μήκος*Πλάτος ΥπολογισμόςΕμβαδού<-Εμ **ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ**	**ΣΥΝΑΡΤΗΣΗ** ΥπολογισμόςΕμβαδού(): **ΠΡΑΓΜΑΤΙΚΗ** **ΣΤΑΘΕΡΕΣ** Π=3.14 **ΜΕΤΑΒΛΗΤΕΣ** **ΠΡΑΓΜΑΤΙΚΕΣ**:Εμ **ΑΡΧΗ** Εμ <- Π*Ακτίνα*Ακτίνα ΥπολογισμόςΕμβαδού <- Εμ **ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ**

Ενότητα 5. Εκσφαλμάτωση Προγράμματος

5.1 Κατηγορίες Λαθών

1) Ποιες είναι οι κατηγορίες λαθών σε ένα πρόγραμμα;

• 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.

5.2 Εκσφαλμάτωση

1. Που μπορεί να εμφανίζονται λογικά λάθη στις δομές επιλογής;
- τη συνθήκη ή τις συνθήκες
- τις ομάδες εντολών που εκτελούνται όταν μια συνθήκη είναι αληθής ή ψευδής.

2. Που μπορεί να εμφανίζονται λογικά λάθη στις δομές επανάληψης;
- τη συνθήκη επανάληψης ή τερματισμού,
- την αρχικοποίηση της συνθήκης,
- την ενημέρωση της συνθήκης εντός του βρόχου επανάληψης,
- τις εντολές που περιλαμβάνονται εντός του βρόχου.

Παράδειγμα 8 – μέθοδος ελέγχου «Μαύρο κουτί»: «Η βαθμολογία σε ένα διαγώνισμα στο μάθημα της πληροφορικής, κυμαίνεται στο διάστημα [1,100] σε ακέραιες τιμές. Για να πετύχει ο μαθητής στο διαγώνισμα, θα πρέπει να συγκεντρώσει τουλάχιστον 60 μονάδες. Να αναπτύξετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο: **1)** θα διαβάζει την βαθμολογία που συγκέντρωσε ένας μαθητής **2)** θα εμφανίζει «επιτυχία» αν πέτυχε στο διαγώνισμα, ή «αποτυχία» σε αντίθετη περίπτωση **3)** αν δοθεί βαθμολογία εκτός [1,100] να εμφανίζει «λάθος δεδομένα»». Με βάση την παραπάνω εκφώνηση, να δημιουργήσετε κατάλληλα σενάρια για να πραγματοποιήσετε έλεγχο ακραίων τιμών.

- ✓ 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.
- ✓ 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.
- ✓ 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.
- ✓ 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 >=).

- ✓ **Βήμα 1:** Δημιουργία ισοδύναμων διαστημάτων.

Υπάρχουν δύο διαστήματα για τις τιμές εισόδου: 1<=βαθμός<60 και 60<=βαθμός<=100.
Υπάρχουν δύο διαστήματα μη έγκυρων τιμών: βαθμός<0 και βαθμός >100.

---------- > | 1 < ---------- > | 60 < ---------- > 100 | < ----------
Λάθος τιμές Αποτυχία Επιτυχία Λάθος τιμές

- ✓ **Βήμα 2:** Καθορισμός ακραίων τιμών των ισοδύναμων διαστημάτων.

---------- > 0 | 1 < ---------- >59 | 60 < ---------- > 100 | 101 < ----------
Λάθος τιμές Αποτυχία Επιτυχία Λάθος τιμές

- ✓ **Βήμα 3:** Δημιουργία σεναρίων ελέγχου για κάθε ακραία τιμή.

Είσοδος→ 0 (άνω άκρο για βαθμός<1) **Αναμενόμενο αποτέλεσμα→**Λάθος δεδομένα
Είσοδος→ 1 (κάτω άκρο για 1<=βαθμός<60) **Αναμενόμενο αποτέλεσμα→**Αποτυχία
Είσοδος→ 59 (άνω άκρο για 1<βαθμός<60) **Αναμενόμενο αποτέλεσμα→**Αποτυχία
Είσοδος→ 60 (κάτω άκρο για 60<=βαθμός<=100) **Αναμενόμενο αποτέλεσμα→**Επιτυχία
Είσοδος→ 100 (άνω άκρο για 60<=βαθμός<=100) **Αναμενόμενο αποτέλεσμα→**Επιτυχία
Είσοδος→ 101 (κάτω άκρο για βαθμός>100) **Αναμενόμενο αποτέλεσμα→**Λάθος δεδομένα