Μαγκανάρη Στέλλα

Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

1.1 Η έννοια πρόβλημα

Πρόβλημα είναι μια κατάσταση που χρειάζεται αντιμετώπιση, απαιτεί λύση η οποία δεν είναι γνωστή ούτε προφανής.

Η αντιμετώπιση ενός προβλήματος προϋποθέτει ότι ο λύτης έχει διαθέσιμες συγκεκριμένες πληροφορίες σχετικά με το ποιο είναι το επιθυμητό αποτέλεσμα, με ποιες προϋποθέσεις, με τη χρήση ποιων εργαλείων και ενεργειών, από ποιες αρχικές πληροφορίες ξεκινάει και σε ποια μέσα έχει πρόσβαση.

Αναφέρετε παραδείγματα προβλημάτων
Σε ποιες κατηγορίες θα τα κατατάσσατε

Δεδομένο είναι οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις του

Πληροφορία είναι οποιοδήποτε στοιχείο προέρχεται από επεξεργασία δεδομένων

Επεξεργασία δεδομένων είναι η διαδικασία όπου ένας μηχανισμός δέχεται δεδομένα, τα επεξεργάζεται με ένα προκαθορισμένο τρόπο και αποδίδει πληροφορία

- **Διατυπώστε με σαφήνεια ένα πρόβλημα. Αναφέρετε τα δεδομένα και τα ζητούμενα του.**

1.2 Κατανόηση προβλήματος

Κατανόηση του προβλήματοςΗ κατανόηση ενός προβλήματος ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων, τη ***σωστή διατύπωση*** εκ μέρους του δημιουργού (σωστή ορολογία και σύνταξη) και τη ***σωστή ερμηνεία*** από τη μεριά εκείνου που πρόκειται να το αντιμετωπίσει (η έννοια της κατανόησης ποικίλλει ανάλογα το επίπεδο γνώσεων του λύτη).

- **Παράδειγμα βιβλίου (σελ. 7), ΔΤ7 (σελ.15)**

1.3 ΔΟΜΗ ΠΡΟΒΛΗΜΑΤΟΣ

Με τον όρο **δομή** ενός προβλήματος αναφερόμαστε στα συστατικά του μέρη, στα επιμέρους τμήματα που το αποτελούν καθώς επίσης και στον τρόπο που αυτά τα μέρη συνδέονται μεταξύ τους.

Ανάλυση → η πορεία επίλυσης προς τα πίσω. Θεωρούμε ότι αυτό που ψάχνουμε έχει ήδη βρεθεί. Προσπαθούμε να βρούμε τι θα έπρεπε να προηγείται, για να στηριχτούμε σε αυτό και να καταλήξουμε στο επιθυμητό αποτέλεσμα.

Σύνθεση → ξεκινάμε από κάτι γνωστό. Από το σημείο αυτό με λογικά βήματα προσπαθούμε να φτάσουμε στο ζητούμενο.

Για την επίλυση σύνθετων προβλημάτων ακολουθούμε τρεις φάσεις:

- **Ανάλυση προβλήματος**
- **Επίλυση κάθε υποπροβλήματος**
- **Σύνθεση λύσεων.**

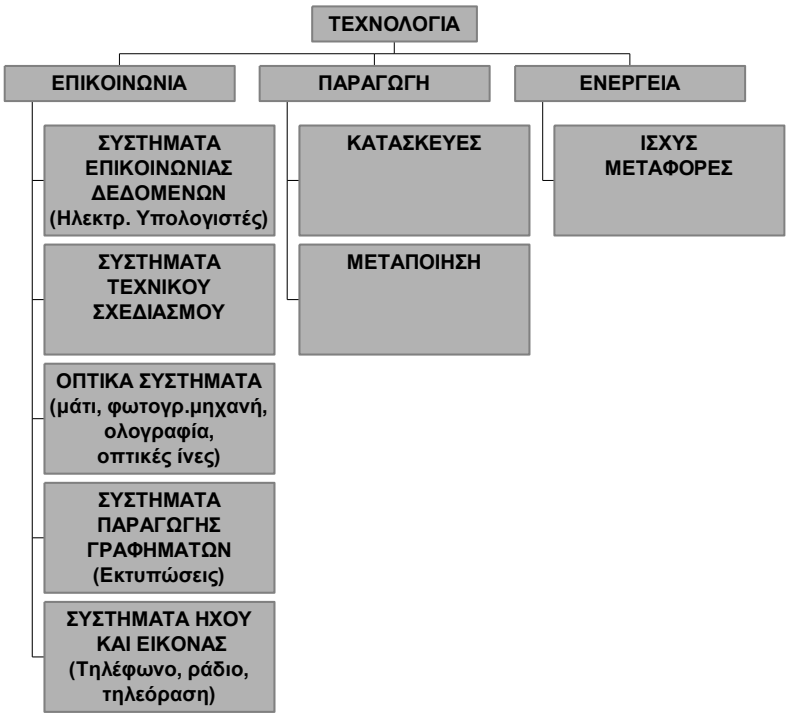
Η παρουσίαση ανάλυσης ενός προβλήματος μπορεί να γίνει με δύο τρόπους:

- **Φραστικά** – περιγράφουμε με λόγια πως και σε ποια υποπροβλήματα χωρίζεται το πρόβλημα.

Παράδειγμα

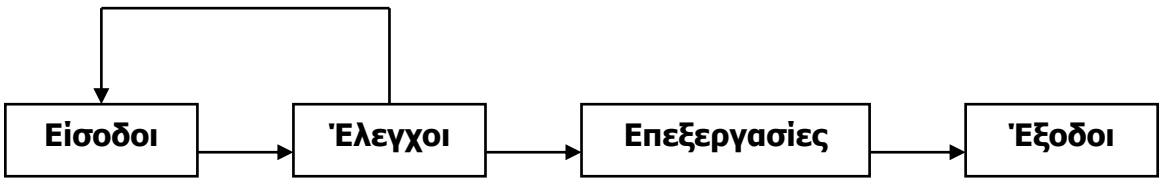
1. Τεχνολογία
- **1.1 Επικοινωνία**
  - **1.1.1 Συστήματα επικοινωνίας δεδομένων**
  - **1.1.2 Συστήματα τεχνικού σχεδιασμού**
  - **1.1.3 Συστήματα ήχου και εικόνας**
  - **1.1.4 Οπτικά συστήματα**
  - **1.1.5 Συστήματα παραγωγής γραφημάτων**
- **1.2 Παραγωγή**
  - **1.2.1 Κατασκευες**
  - **1.2.2 Μεταποίηση**
- **1.3 Μεταφορές**

➢ **Διαγραμματική αναπαράσταση** – η περιγραφή γίνεται σε σχήμα δένδρου.



ΔΣ3 (σελ. 15) Παρουσιάστε την Φραστική αναπαράσταση της ανάλυσης του προβλήματος *(ομάδες – 5 λεπτά)*

Σε γενικές γραμμές η αντιμετώπιση ενός προβλήματος αποτελείται από τις ενέργειες του παρακάτω σχήματος:



- Οι είσοδοι αφορούν καταχώρηση των δεδομένων του προβλήματος.
- Τα δεδομένα ελέγχονται και ίσως χρειαστεί επανακαταχώρηση.
- Γίνονται οι απαραίτητες επεξεργασίες ή υπολογισμοί προκειμένου να βρεθούν τα ζητούμενα αποτελέσματα (πληροφορίες).
- Οι έξοδοι αφορούν εξαγωγή των επιθυμητών αποτελεσμάτων.

1.4 ΚΑΘΟΡΙΣΜΟΣ ΑΠΑΙΤΗΣΕΩΝ

Αφορά πρώτα τον ακριβή προσδιορισμό των ***δεδομένων*** ή ***εισόδων*** που υπάρχουν σ' ένα πρόβλημα και ύστερα τον ακριβή προσδιορισμό των ***ζητούμενων*** ή ***εξόδων*** (δηλ. των πληροφοριών ή αποτελεσμάτων που επιλύουν το πρόβλημα). Π.χ. σε μια άσκηση ο καθορισμός των απαιτήσεων μπορεί να περιλαμβάνει ορισμό δεδομένων-μεταβλητών, πινάκων, κλπ. Πόσες και ποιες μεταβλητές θα χρησιμοποιήσω Μήπως χρειάζομαι δομή δεδομένων (π.χ. πίνακα) Τι πρέπει να εκτυπώνεται τελικά Αυτά είναι μερικά παραδείγματα ερωτημάτων που πρέπει να μας απασχολούν. Επομένως ο καθορισμός των απαιτήσεων πρέπει να γίνεται ***πάντα***.

Ιδιότητες δεδομένων
- ✓ Ορθότητα
- ✓ Πληρότητα
- ✓ Σαφήνεια

Ιδιότητες ζητουμένων
- ✓ Πληρότητα
- ✓ Σαφήνεια

Σαφήνεια στη διατύπωση του προβλήματος → Πόσο περιορισμένες είναι οι επιλογές που έχει ή πόσες διευκρινιστικές ερωτήσεις πρέπει να κάνει αυτός που πρόκειται να λύσει το πρόβλημα.
Ζητούμενα ενός προβλήματος είναι το σημείο στο οποίο πρέπει να φτάσουμε για να λύσουμε το πρόβλημα.
Αποτελέσματα είναι τα στοιχεία που θα προκύψουν από την επίλυση του προβλήματος.
Στην επίλυση ενός προβλήματος είναι σημαντικό η παρουσίαση των αποτελεσμάτων να γίνει με τρόπο ώστε να είναι εύκολα κατανοητά από κάποιον άλλον.

- **ΔΤ3**

Ένα γνωσιακό ποσό δεν μπορεί να χαρακτηριστεί αυτόματα ως δεδομένο ή πληροφορία. Το ίδιο γνωσιακό στοιχείο μπορεί να είναι πληροφορία σε μια κατάσταση και δεδομένο σε κάποια άλλη, ανάλογα την διαδικασία επεξεργασίας από την οποία προέρχεται.

- **Ανάλυση ΔΤ1 (α) – (τεστ σελ 29 Β.Κ.)**

- **Εργασίες ΔΤ1 (β), ΔΤ2, ΔΣ1**

Τεστ σελ. 29 Β.Κ. – 3 λεπτά

1.5 ΕΙΔΗ ΠΡΟΒΛΗΜΑΤΩΝ

Πρακτικά η πληροφορική δεν παρέχει ιδιαίτερη υποστήριξη για τα άλυτα ούτε για τα αδόμητα προβλήματα. Είναι όμως σημαντικός ο ρόλος της στα **επιλύσιμα**, στα **δομημένα** (αυτοματοποιημένα) και στα προβλήματα **βελτιστοποίησης.**

➢ ***Είδη προβλημάτων***

1. **Ως προς την επιλυσιμότητα τους:**
   - ✓ **Επιλύσιμα** → η λύση τους είναι ήδη γνωστή και έχει διατυπωθεί.
   - ✓ **Ανοικτά** → η λύση τους δεν έχει βρεθεί, αλλά δεν έχει αποδειχθεί η μη ύπαρξη της.
   - ✓ **Άλυτα** → έχει αποδειχθεί ότι δεν υπάρχει λύση. (τετραγωνισμός του κύκλου)
2. **Ως προς το βαθμό δόμησης των λύσεων τους τα επιλύσιμα προβλήματα διακρίνονται σε τρεις κατηγορίες:**
   - ✓ **Δομημένα →** η λύση προέρχεται από αυτοματοποιημένη διαδικασία (δευτεροβάθμια εξίσωση).
   - ✓ **Ημιδομημένα →** η λύση μπορεί να επιλεγεί ανάμεσα από ένα εύρος πιθανών λύσεων, αφήνοντας στον ανθρώπινο παράγοντα περιθώρια επιλογής της. (συμπλήρωση του πεδίου οικογενειακή κατάσταση)
   - ✓ **Αδόμητα →** οι λύσεις δεν μπορούν να δομηθούν ή δεν έχει διερευνηθεί σε βάθος η δυνατότητα δόμησης τους. (η οργάνωση ενός πάρτι είναι αδόμητο πρόβλημα εφόσον το πλήθος των επιλογών είναι απεριόριστο).
3. **Με κριτήριο το είδος της επίλυσης, που επιζητούν τα προβλήματα διακρίνονται σε τρεις κατηγορίες:**
   - ✓ **Απόφασης →** η λύση του προβλήματος απαντά σε ένα ερώτημα
   - ✓ **Υπολογιστικά →** απαιτεί τη διενέργεια υπολογισμών. (δίνεται αριθμός v και ζητείται να βρεθεί πόσες διαφορετικές παραγοντοποιήσεις του v υπάρχουν).
   - ✓ **Βελτιστοποίησης →** το πρόβλημα που τίθεται επιζητά το βέλτιστο αποτέλεσμα για τα συγκεκριμένα δεδομένα που τίθενται. (δίνεται αριθμός v και ζητείται να βρεθεί ποια είναι η πραγοντοποίηση του v με το μεγαλύτερο πλήθος παραγόντων).

Χαρακτηρίστε τα προβλήματα: ΔΤ2, ΔΤ6, ΔΤ8 (σελ. 22-23) (ομαδικά)

Οι υπολογιστές δεν επιλύουν προβλήματα με `μαγικό` τρόπο. Προβλήματα λύναμε και πριν την εμφάνιση των υπολογιστών. Οι υπολογιστές (και η επιστήμη της πληροφορικής) δρουν επικουρικά ή υποστηρικτικά στην επίλυση προβλημάτων.
Ο άνθρωπος υπερέχει ποιοτικά σε σχέση με τον υπολογιστή. Ο υπολογιστής μπορεί να εκτελεί μόνο **αριθμητικές πράξεις** (πρόσθεση), **λογικές πράξεις** (σύγκριση) και **μεταφορά δεδομένων**. Οι πράξεις βέβαια μπορούν να εκτελούνται ταχύτατα στον υπολογιστή. Επίσης ο υπολογιστής διαθέτει κύρια μνήμη (RAM) και περιφερειακές μνήμες (π.χ. δίσκοι). Άρα ο υπολογιστής διαθέτει τρομακτικά ισχυρούς μηχανισμούς **για αποθήκευση και μεταφορά δεδομένων**.
Επομένως μπορούμε να αναθέσουμε ένα πρόβλημα σε έναν υπολογιστή όταν:
- ✓ Υπάρχουν πολύπλοκοι υπολογισμοί.
- ✓ Υπάρχουν διαδικασίες που επαναλαμβάνονται .
- ✓ Υπάρχουν πολλές πράξεις (αριθμητικές ή λογικές).
- ✓ Υπάρχει μεγάλο πλήθος (όγκος) δεδομένων.

Τεστ (σελ 31 Β.Κ.) – *3 λεπτά*
Εργασίες 1.5 τεστ αυτοαξιολόγησης (σελ. 16)

2.1 ΤΙ ΕΙΝΑΙ ΑΛΓΟΡΙΘΜΟΣ

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

Κάθε αλγόριθμος πρέπει να ικανοποιεί τα κριτήρια:

- ✓ **Είσοδος** → καμία (ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων τυχαίων αριθμών), μία ή περισσότερες είσοδοι

Προσδιορίστε την/τις εισόδους των παρακάτω δραστηριοτήτων ΔΣ9 (σελ. 26), ΔΣ1, ΔΣ4 (σελ. 34-35)

- ✓ **Έξοδος** → τουλάχιστον μία.
- ✓ **Καθοριστικότητα** → κάθε εντολή να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσης της.
- ✓ **Περατότητα** → ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης.
  1. Διάβασε 1^{ο} αριθμό Α
  2. Max← Α
  3. Διάβασε επόμενο αριθμό Β
  4. Αν Β>max τότε Β←max
  5. Αν υπάρχει άλλος αριθμός, πήγαινε στο βήμα 1
  6. Εμφάνισε max

Τι παρατηρείτε στο παραπάνω σχέδιο λύσης Κάνετε τις απαραίτητες διορθώσεις.

- ✓ **Αποτελεσματικότητα →** Κάθε μεμονωμένη εντολή να είναι εκτελέσιμη.

Extra

- *Εφόσον ο τελικός σκοπός ενός αλγορίθμου είναι η κωδικοποίηση του (μετατροπή) σε μία κατάλληλη διαδικαστική γλώσσα προγραμματισμού[1] (υπολογιστή) τα χαρακτηριστικά που ζητάμε από τον τρόπο παρουσίασης ενός αλγορίθμου είναι:*
- ***Κατανοησιμότητα** → ο αλγόριθμος να γίνεται εύκολα κατανοητός από κάποιον που τον διαβάζει ώστε να μπορεί εύκολα να τον κωδικοποιήσει σε μια γλώσσα προγραμματισμού.*
- ***Εποπτικότητα** → με μια ματιά να καταλαβαίνουμε τι κάνει.*
- ***Ακρίβεια** → να λύνει το πρόβλημα χωρίς να χρειάζονται διευκρινίσεις από αυτόν που τον σχεδίασε.*
- ***Εύκολη υλοποίηση** → εύκολη μετατροπή του αλγορίθμου (σχεδίου λύσης) σε μια γλώσσα προγραμματισμού.*
- ***Δυνατότητα έκφρασης δομών δεδομένων[2] →** όσο ποιο εύκολο είναι να εκφραστούν δομές δεδομένων σε έναν αλγόριθμο τόσο ποιο εύκολη είναι η λύση σύνθετων προβλημάτων.*

2.2 ΣΠΟΥΔΑΙΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ

- ✓ **Υλικού**
- ✓ **Γλωσσών προγραμματισμού**
- ✓ **Θεωρητική**
- ✓ **Αναλυτική**

2.3 Περιγραφή και αναπαράσταση αλγορίθμων

- ✓ **Ελεύθερο κείμενο** → Αδόμητος τρόπος αναπαράστασης αλγορίθμου. Υπάρχει κίνδυνος να παραβιάσει το κριτήριο της αποτελεσματικότητας. *(Παράδειγμα)*
- ✓ **Διαγραμματικές τεχνικές** (π.χ. διαγράμματα ροής). → Χρησιμοποιούμε σχήματα για την αναπαράσταση των εντολών του αλγορίθμου.

Δραστηριότητα ΔΤ1 (σελ. 22) με διάγραμμα ροής.

- ✓ **Φυσική γλώσσα κατά βήματα →** Υπάρχει κίνδυνος να παραβιάσει το κριτήριο της καθοριστικότητας.

[1] Pascal, Qbasic, C

[2] Δομή δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών. Παραδείγματα δομών δεδομένων είναι οι πίνακες τα αρχεία οι εγγραφές, δένδρα κ.α.

- ✓ **Κωδικοποίηση** → ένα πρόγραμμα γραμμένο είτε με μία ψευδογλώσσα είτε σε κάποιο προγραμματιστικό περιβάλλον που όταν εκτελεστεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Επιπλέον στοιχεία κωδικοποίηση:

- Στην πρώτη γραμμή ενός αλγορίθμου βάζουμε πάντα **Αλγόριθμος** όνομα_αλγορίθμου όπου όνομα_αλγορίθμου είναι ένα λεκτικό που περιγράφει (εν συντομία) τον αλγόριθμο. Στην τελευταία γραμμή ενός αλγορίθμου βάζουμε πάντα **Τέλος** όνομα_αλγορίθμου.
- Στη δεύτερη γραμμή ενός αλγορίθμου μπορούμε να ορίσουμε (αν χρειάζεται) τα δεδομένα εισόδου εντός των συμβόλων //……// (π.χ. **Δεδομένα** // x, y, table //). Αυτό το κάνουμε αν π.χ. έχουμε δεδομένη την τιμή μιας μεταβλητής ή δεδομένο έναν πίνακα (γεμάτο με τιμές) και δε χρειάζεται να χρησιμοποιήσουμε εντολές εισόδου (δηλ. **διάβασε**). Στην προτελευταία γραμμή ενός αλγορίθμου βάζουμε τα αποτελέσματα εξόδου εντός των συμβόλων //……// (π.χ. **Αποτελέσματα** // min, max, table //). Αυτό το κάνουμε αν π.χ. σ' ένα αλγόριθμο έχουμε στην έξοδο πληροφορίες όπως την τιμή μιας μεταβλητής ή ενός πίνακα και δε θέλουμε να χρησιμοποιήσουμε εντολές εξόδου (δηλ. **εμφάνισε**).

Δομή αλγορίθμου σε κωδικοποίηση:

Αλγόριθμος όνομα_αλγορίθμου
Δεδομένα // Ορισμός δεδομένων //
    Εντολή 1
    Εντολή 2
    …….
    Εντολή ν
Αποτελέσματα // Ορισμός αποτελεσμάτων //
Τέλος όνομα_αλγορίθμου

Δραστηριότητα ΔΤ2 (σελ. 22) με κωδικοποίηση.

2.4 Βασικές συνιστώσες/ εντολές ενός αλγορίθμου

- ✓ **Σταθερές** (π.χ. 205, 15, "ψηλός", "ανύπαρκτη τιμή", "English", αληθής (true), ψευδής (false)).
- ✓ **Μεταβλητές** (ακέραιες, πραγματικές, αλφαριθμητικές, λογικές).
- ✓ **Αριθμητικοί τελεστές** (+, -, *, /,div, mod, ^).
- ✓ **Συγκριτικοί τελεστές** (=, <, <=, >, >=, <>).
- ✓ **Λογικοί τελεστές** (ΚΑΙ (AND), Ή (OR), ΟΧΙ (NOT)).
- ✓ **Εκφράσεις** (π.χ. 500, a, a=0, x+y, (i<=50 ή done=ψευδής), κλπ).

2.4.1 Δομή ακολουθίας

Χρησιμοποιείται για την επίλυση 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/ εξόδου** (δηλ. **διάβασε/ εμφάνισε**) κι οι εντολές **ανάθεσης ή απόδοσης τιμής** (δηλ. Μεταβλητή ← Έκφραση).

Δραστηριότητα: Δίνονται οι αριθμοί α, β ,γ, δ. Να βρεθεί η τιμή της παράστασης α(β+γ)/δ.

2.4.2 Δομή επιλογής

Γενικά η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (αληθής ή ψευδής) και ακολουθεί η απόφαση εκτέλεσης κάποιας ενέργειας με βάση την τιμή της λογικής αυτής έκφρασης.

Α) ΑΝ…ΤΟΤΕ
ΑΝ συνθήκη ***ΤΟΤΕ***
    Ομάδα εντολών 1
ΤΕΛΟΣ ΑΝ

Β) ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ
ΑΝ συνθήκη ***ΤΟΤΕ***
    Ομάδα εντολών 1
ΑΛΛΙΩΣ
    Ομάδα εντολών 2
ΤΕΛΟΣ ΑΝ

Δραστηριότητα: Δίνονται οι αριθμοί α, β ,γ, δ. Να βρεθεί η τιμή της παράστασης α(β+γ)/δ (δ=0).

Λογικές πράξεις

Πίνακας αληθείας των 3 βασικών λογικών πράξεων **(λογική σύζευξη (ΚΑΙ), λογική διάζευξη (Ή), λογική**

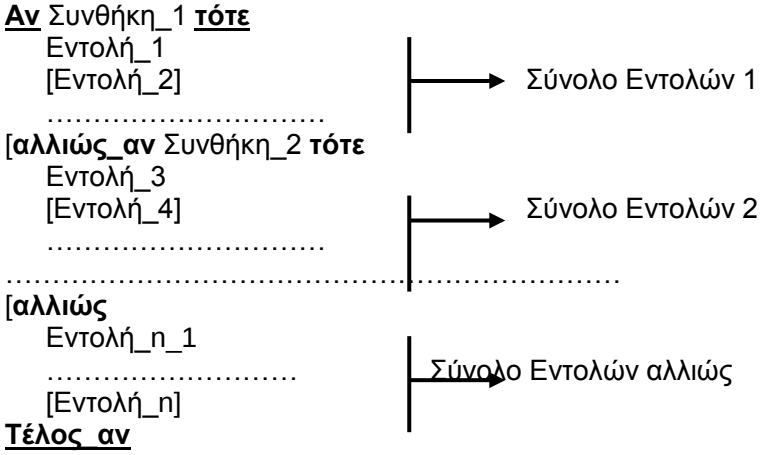
άρνηση (ΟΧΙ)).

Συνθήκη Α	Συνθήκη Β	Α ΚΑΙ Β	Α Ή Β	ΟΧΙ Α
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής

2.4.3 Δομή πολλαπλής επιλογής

Οι διαδικασίες των πολλαπλών επιλογών εφαρμόζονται στα προβλήματα όπου μπορεί να ληφθούν διαφορετικές αποφάσεις ανάλογα την τιμή που παίρνει μια έκφραση. Στη δομή αυτή ανήκουν οι εντολή **Αν..τότε...αλλιώς_αν** κι η εντολή **Επίλεξε**.

Γενική Μορφή Αν..Τότε...Αλλιώς_Αν



Οι λέξεις που είναι έντονες λέγονται **δεσμευμένες** λέξεις. Αυτό σημαίνει ότι δεν μπορούν να αντικατασταθούν με άλλη λέξη. Δηλ. δεν επιτρέπεται να χρησιμοποιήσουμε δικές μας λέξεις αντί γι' αυτές ούτε μπορούμε να τις παραλείψουμε. Οι συνθήκες είναι λογικές εκφράσεις (π.χ. a=0, x+y>10, (j>=5 και found=αληθής), κλπ).

Σε περίπτωση που η συνθήκη ισχύει (δηλ. είναι αληθής) εκτελείται το αντίστοιχο σύνολο εντολών (που περιέχει τουλάχιστον μία εντολή). Στην πολλαπλή επιλογή, κάθε φορά εκτελείται ένα ή κανένα από τα σύνολα εντολών κατά περίπτωση. Η μοναδική περίπτωση στην οποία δεν εκτελείται κανένα σύνολο εντολών είναι να μην ισχύει καμία συνθήκη και να μην υπάρχει καθόλου το τμήμα **αλλιώς**. Εντολή μπορεί να είναι μία οποιαδήποτε αποδεκτή εντολή (π.χ. εισόδου/ εξόδου, ανάθεσης τιμής, επιλογής, επανάληψης, κλπ). Σε περίπτωση που μία από τις εντολές είναι άλλη εντολή **Αν** τότε έχουμε **Εμφωλευμένο Αν**. Προφανώς μπορούμε να εμφωλεύσουμε πολλές εντολές **Αν** μέσα σε μία εντολή **Αν** (κάθε μία πρέπει να έχει δικό της **Τέλος_αν**). Το εμφωλευμένο **Αν** για να μην οδηγήσει σε πιθανά λάθη πρέπει να χρησιμοποιείται για να καλύψει μία ξεκάθαρη υποπερίπτωση μιας περίπτωσης (π.χ. **Αν** city="Πάτρα" **τότε** …… και μετά εμφώλευση του **Αν** class = "Γ Λυκείου" **τότε** ..).

Δραστηριότητα ΔΤ7 (σελ. 23)

*Συνηθισμένα τεχνικά λάθη στη χρήση της εντολής **Αν**:*

*α) **Αν** a>0 **τότε***
*   **εμφάνισε** "α θετικός"*
*   **αλλιώς_αν** b<=0 **τότε***
*   **εμφάνισε** "β αρνητικός"*
*   **Τέλος_αν***

*Εδώ δεν υπάρχει κανένα λάθος στη σύνταξη της εντολής η οποία είναι αποδεκτή. Σε περίπτωση όμως που a<=0 και b>0 δεν εκτυπώνεται τίποτα! Αυτό συμβαίνει διότι υπάρχουν δύο συνθήκες στην ίδια εντολή που ελέγχουν όμως τελείως διαφορετικά πράγματα (το a και το b). Απαιτούνται λοιπόν δύο απλές και ξεχωριστές εντολές **Αν**. Επομένως για την αποφυγή λαθών καλό είναι οι συνθήκες σε **μία** εντολή να μην ελέγχουν διαφορετικά (σε νόημα) πράγματα.*

*β) **Αν** a>0 **τότε***
*   **Αν** a<=0 **τότε***
*   **εμφάνισε** "α θετικός"*
*   **Τέλος_αν***
*   **Τέλος_αν***

*Ούτε εδώ δεν υπάρχει κανένα λάθος στη σύνταξη αλλά υπάρχει αλόγιστη χρήση εμφωλευμένου **Αν**. Το δεύτερο **Αν** δεν πρόκειται να εκτελεστεί ποτέ! Το παραπάνω τμήμα αλγορίθμου λοιπόν δεν έχει κανένα νόημα. Επομένως για την αποφυγή λαθών καλό είναι τα εμφωλευμένα **Αν** να ελέγχουν διαφορετικά (σε νόημα) πράγματα. Πιο συγκεκριμένα πρέπει να ελέγχουν συγκεκριμένες (διακριτές) υποσυνθήκες της συνθήκης στην οποία περιέχονται (π.χ. για συνταξιοδότηση γυναικών χρησιμοποιώ τις εντολές **Αν** gender="female" **τότε** … και μετά εμφώλευση του **Αν** age >= 63 **τότε** …).*

- ***Αν οι διαφορετικές περιπτώσεις είναι πολλές, τότε είναι προτιμότερο να χρησιμοποιηθεί το σχήμα πολλαπλής επιλογής ΕΠΙΛΕΞΕ.***
- ***Η εντολή Επίλεξε συντάσσεται (στη γενική της μορφή) ως εξής:***

Επίλεξε έκφραση
Περίπτωση Λίστα_Τιμών_1
    Εντολή_1_1
    [Εντολή_1_2]
    ……
[**Περίπτωση** Λίστα_Τιμών_2
    Εντολή_2_1
    [Εντολή_2_2]
    ……
[**Περίπτωση_Αλλιώς**
    Εντολή_n_1
    [Εντολή_n_2]
    ……
Τέλος_επιλογών

→ Σύνολο Εντολών 1
→ Σύνολο Εντολών 2
→ Σύνολο Εντολών αλλιώς

Οι λέξεις που είναι έντονες είναι πάλι **δεσμευμένες** λέξεις. Η έκφραση συνήθως είναι μία μεταβλητή ή γενικότερα μία έκφραση. Η λίστα τιμών μπορεί να είναι μία απλή σταθερά ή μεταβλητή (π.χ. 5, 100, a), ένα εύρος τιμών (>=30, <200, {1,2,3}, 10-20 κλπ). Σε περίπτωση που η έκφραση επιβεβαιώνει τη λίστα τιμών εκτελείται το αντίστοιχο σύνολο εντολών (που περιέχει τουλάχιστο μία εντολή).

Σε μία εντολή **Επίλεξε** κάθε φορά εκτελείται ένα ή κανένα από τα σύνολα εντολών κατά περίπτωση. Η μοναδική περίπτωση στην οποία δεν εκτελείται κανένα σύνολο εντολών είναι να μην επιβεβαιώνεται καμία λίστα τιμών και να μην υπάρχει καθόλου το τμήμα **Περίπτωση Αλλιώς**. Το τμήμα **Περίπτωση Αλλιώς** ερμηνεύεται ως `σε κάθε άλλη περίπτωση`. Είναι προφανές πως η εντολή **Επίλεξε** είναι παρόμοια με την εντολή **Αν**. Επομένως είναι δυνατόν να μετατρέψουμε μία εντολή **Αν** σε εντολή **Επίλεξε** κι αντίστροφα.

Δραστηριότητα ΔΤ9 (σελ. 23)

Εμφωλευμένη επιλογή

Εμφωλευμένη επιλογή έχουμε όταν μία δομή επιλογής περιέχεται σε μία άλλη δομή επιλογής.

Επίλυση δευτεροβάθμιας εξίσωσης με διάγραμμα ροής.

- **Γενική μορφή ΟΣΟ .. ΓΙΑ.. (σε διάγραμμα ροής)**
- **Σαββάλας 1.56, 1.59 (σελ. 102) ή *παραδείγματα (σελ. 24-25)***

2.4.5 Δομή επανάληψης

Αποτελείται από ένα σύνολο εντολών που εκτελούνται πολλές φορές (αυτοματοποιημένα). Εφαρμόζεται όταν μια σειρά εντολών πρέπει να εκτελεστεί σε ένα σύνολο περιπτώσεων, που έχουν κάτι κοινό. Στη δομή αυτή ανήκουν η εντολή **Όσο…**, η εντολή **Αρχή_επανάληψης** κι η εντολή **Για…**.

Η εντολή Όσο συντάσσεται (στη γενική της μορφή) ως εξής:

Όσο Συνθήκη **επανάλαβε**
Εντολή_1
[Εντολή_2]
……
Τέλος_επανάληψης

→ Σύνολο Εντολών

- Οι λέξεις που είναι έντονες είναι πάλι **δεσμευμένες** λέξεις. Η συνθήκη είναι μία λογική έκφραση (που είναι είτε αληθής είτε ψευδής). Πρέπει να προσέχουμε οι μεταβλητές που υπάρχουν στη συνθήκη να έχουν πάρει τιμές (π.χ. με χρήση εντολών εισόδου ή ανάθεσης τιμής) πριν την εκτέλεση της εντολής **Όσο**. Το σύνολο εντολών είναι δυνατό να εκτελεστεί από 0 έως πολλές φορές. Βέβαια πρέπει να εξασφαλίσουμε ότι η συνθήκη είναι τέτοια που δεν θα υπάρχει πρόβλημα περατότητας (βλ. παρ. 2.1).
- Η εντολή **Όσο** λειτουργεί ως εξής: Αρχικά ελέγχεται η συνθήκη. Αν είναι **αληθής** τότε εκτελείται το σύνολο εντολών. Στη συνέχεια ελέγχεται εκ νέου η συνθήκη κι αν είναι αληθής τότε εκτελείται πάλι το σύνολο εντολών. Όταν η συνθήκη γίνει ψευδής τότε τελειώνει η επανάληψη κι ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το **Τέλος_επανάληψης**.

Σαββάλας 1.60, 1.73, 1.70, 1.71 (σελ. 102-104)

Η εντολή Αρχή_επανάληψης συντάσσεται (στη γενική της μορφή) ως εξής:

Αρχή_επανάληψης
Εντολή_1
[Εντολή_2]
……
Μέχρις_ότου Συνθήκη

→ Σύνολο Εντολών

- Η συνθήκη είναι μία λογική έκφραση (που είναι είτε αληθής είτε ψευδής). Το σύνολο εντολών είναι δυνατό να εκτελεστεί από **1** έως πολλές φορές. Βέβαια πρέπει να εξασφαλίσουμε ότι η συνθήκη είναι τέτοια που δεν θα υπάρχει πρόβλημα περατότητας (βλ. παρ. 2.1).
- Η εντολή **Αρχή_επανάληψης** λειτουργεί ως εξής: Αρχικά εκτελείται το σύνολο εντολών. Στη συνέχεια ελέγχεται η συνθήκη κι αν είναι ψευδής τότε εκτελείται πάλι το σύνολο εντολών (υπάρχει αντίθεση με την εντολή **Όσο**!). Όταν η συνθήκη γίνει αληθής τότε τελειώνει η επανάληψη κι ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το **Μέχρις_ότου**.

Σαββάλας 1.60, 1.73, 1.78 (σελ. 102-104)

- ***Η εντολή Για συντάσσεται (στη γενική της μορφή) ως εξής:***

Για μετρητής **από** Τ1 **μέχρι** Τ2 [**με_βήμα** Τ3]
Εντολή_1
[Εντολή_2]
……
Τέλος_επανάληψης

→ Σύνολο Εντολών

- Ο μετρητής είναι μία μεταβλητή. Τ1 είναι μία σταθερά ή μεταβλητή που καθορίζει την αρχική τιμή του μετρητή. Τ2 είναι μία σταθερά ή μεταβλητή που καθορίζει την τελική τιμή του μετρητή. Το βήμα είναι μία σταθερά ή μεταβλητή (διάφορη του μηδέν), που καθορίζει πως θα μεταβάλλεται ο μετρητής. Το βήμα είναι προαιρετικό κι αν δε σημειώνεται είναι ίσο με 1 (δηλ. κάθε φορά ο μετρητής αυξάνεται κατά 1). Το σύνολο εντολών είναι δυνατό να εκτελεστεί τόσες φορές όσες καθορίζει ο μετρητής (0 ή περισσότερες φορές).
- Η εντολή **Για** λειτουργεί ως εξής: Αρχικά ο μετρητής παίρνει αρχική τιμή ίση με Τ1. Αν ο μετρητής είναι μικρότερος ή ίσος από Τ2 εκτελείται το σύνολο εντολών. Στη συνέχεια αυξάνεται ο μετρητής κατά Τ3. Αν ο μετρητής είναι μικρότερος ή ίσος από Τ2 εκτελείται πάλι το σύνολο εντολών. Όταν ο μετρητής γίνει μεγαλύτερος από Τ2 τότε τελειώνει η επανάληψη κι ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το **Τέλος_επανάληψης**.

Σαββάλας 1.70, 1.71, 1.80, ΔΤ10 Τετ. μαθητή (σελ. 102-106)

- Η γενικότερη εντολή επανάληψης είναι η εντολή **Όσο**. Με αυτήν μπορούμε να πραγματοποιήσουμε **ο-**

ποιαδήποτε επαναληπτική διαδικασία. Αυτό βέβαια δε σημαίνει ότι η εντολή Όσο είναι πάντα η ευκολότερη στη χρήση. Η εντολή **Αρχή_επανάληψης** εξυπηρετεί καλύτερα αν θέλουμε ένα σύνολο εντολών να εκτελεστεί τουλάχιστο μία φορά. Η εντολή **Για** εξυπηρετεί όταν ο αριθμός των επαναλήψεων είναι προκαθορισμένος.

- Τεστ αξιολόγησης βιβλία καθηγητή (σελ. 50)
- Δραστηριότητες για το σπίτι: **ΔΤ10, ΔΣ3 Τετ. μαθητή** Σαββάλας

Πολλαπλασιασμός αλά ρωσικά

Είναι η μέθοδος με την οποία ο υπολογιστής πολλαπλασιάζει δύο αριθμούς. Συγκεκριμένα, αν έχουμε δύο αριθμούς Χ και Υ και θέλουμε το Χ*Υ τότε αυτό γίνεται ως εξής:
Βήμα 1 Ο Χ διπλασιάζεται και ο Υ υποδιπλασιάζεται.
Βήμα 2 Συνεχίζουμε το Βήμα 1 μέχρι να φτάσει ο Υ σε 1.
Βήμα 3 Το γινόμενο θα είναι το άθροισμα των στοιχείων Χ στα βήματα που το Υ είναι περιττός.
Π.χ. Χ = 45, Υ = 19

X	Y	
45	19	45
90	9	90
180	4	
360	2	
720	1	720
	Άθροισμα=	855

```
Πρόγραμμα Πολλ/σμός_αλά_ρωσικά
Μεταβλητές
        Ακέραιες:  X, Y    !οι αριθμοί
                   P       !το αποτέλεσμα-γινόμενο
Αρχή
     P ← 0    !Αρχικοποίηση γινομένου
      Οσο Y>0 επανέλαβε
           Αν (Y MOD 2) = 1    τότε   !αν το υπόλοιπο της διαίρεσης του Υ με το 2 είναι 1
                P ← P + X             !πρόσθεσε το Χ στο άθροισμα
           τέλος_αν
           X ← X * 2           !διπλασιασμός του Χ
           Y ←  Y DIV 2          !υποδιπλασιασμός του Υ. Παίρνουμε το πηλίκο του Υ με το 2
      τέλος_επανάληψης
      Τύπωσε  ''Το γινόμενο είναι '', P
τέλος_προγράμματος Πολλ/σμός_αλά_ρωσικά
```

Ποια η πρακτική σημασία της μεθόδου αυτής ; Διότι στους Η/Υ υλοποιείται, σε επίπεδο κυκλωμάτων, πολύ απλά και ταχύτατα με την εντολή της ολίσθησης (shift).

Ολίσθηση κατά μία θέση αριστερά → πολλαπλασιασμός επί 2

Ολίσθηση κατά μία θέση δεξιά → ακέραια διαίρεση διά 2

- **Δραστηριότητα: μετατροπή σε διάγραμμα ροής – πίνακας τιμών για χ=2, ψ=3.**

Αλγοριθμική δομή είναι ο τρόπος με το οποίο εκτελείται ένα συγκεκριμένο πλήθος εντολών ενός αλγορίθμου. Οι βασικές Αλγοριθμικές δομές είναι:
1. Ακολουθία ή διαδοχή, 2. Επιλογής, 3. Επανάληψης

Ακολουθία → οι εντολές εκτελούνται η μία μετά την άλλη. Αποτελεί την ποιο απλή μορφή εκτέλεσης εντολών.

Ψευδοκώδικας

Εντολή 1

Εντολή 2

…..

Εντολή n

Αλγόριθμος εύρεσης_δύναμης

! εύρεση 2ης δύναμης αριθμού

P<-- 1

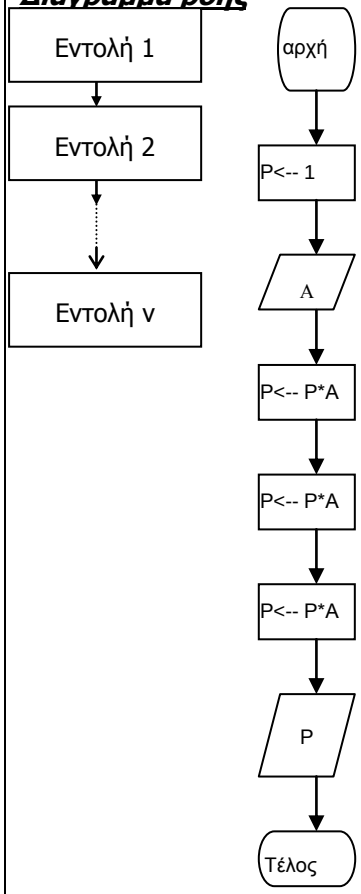
Διάβασε α

P<-- P * a P<-- P * a

Εμφάνισε P

Τέλος εύρεσης_δύναμης

Διάγραμμα ροής



Επιλογή → Κατά την επίλυση πολλών προβλημάτων, σε κάποιο στάδιο χρειάζεται να αποφασιστεί ποια θα είναι η επόμενη πράξη. Η απόφαση αυτή λαμβάνεται με βάση κάποιας συνθήκης.

Ψευδοκώδικας — **Απλή επιλογή** — ***Διάγραμμα ροής***

…

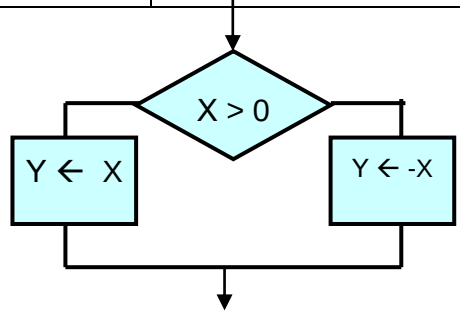
ΑΝ συνθήκη ***ΤΟΤΕ***

Ομάδα εντολών 1

ΑΛΛΙΩΣ

Ομάδα εντολών 1

ΤΕΛΟΣ ΑΝ



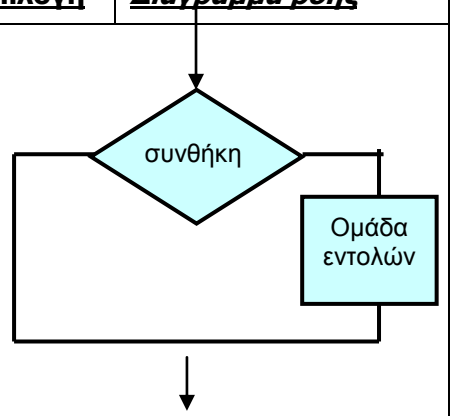
Ψευδοκώδικας — **Περιορισμένη επιλογή** — ***Διάγραμμα ροής***

…

ΑΝ συνθήκη ***ΤΟΤΕ***

Ομάδα εντολών 1

ΤΕΛΟΣ ΑΝ



Δομή επανάληψης : Υπάρχουν 3 μορφές επαναληπτικής δομής οι οποίες διαφέρουν μεταξύ τους στον τρόπο με το οποίο ελέγχεται το πλήθος των επαναλήψεων.

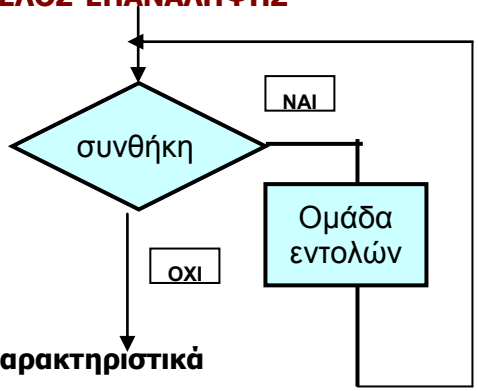
Οι 3 επαναληπτικές δομές είναι : **ΟΣΟ - ΕΠΑΝΑΛΑΒΕ, ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ – ΜΕΧΡΙ_ΟΤΟΥ, ΓΙΑ – ΑΠΟ - ΜΕΧΡΙ.**

1η επαναληπτική ΟΣΟ - ΕΠΑΝΑΛΑΒΕ

ΟΣΟ (συνθήκη) ΕΠΑΝΑΛΑΒΕ

Ομάδα εντολών

ΤΕΛΟΣ-ΕΠΑΝΑΛΗΨΗΣ



Χαρακτηριστικά

- Η ομάδα εντολών εκτελείται όσο η συνθήκη είναι αληθής

-Απαιτείται στην ομάδα εντολών να υπάρχουν εντολές μεταβολής της τιμής της συνθήκης

- Στον μετρητή δίνουμε μια αρχική τιμή πριν την εντολή επανάληψης.

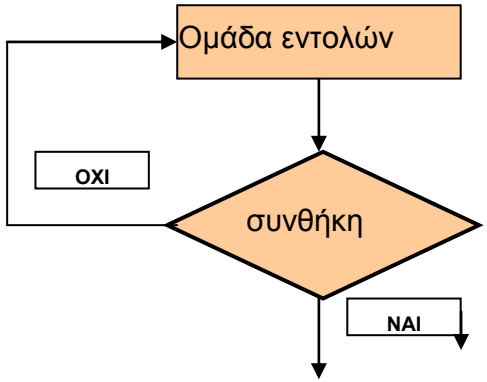
2η επαναληπτική ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

Ομάδα εντολών

ΜΕΧΡΙ_ΟΤΟΥ (συνθήκη)

Λογικό διάγραμμα



Χαρακτηριστικά

- Η ομάδα εντολών εκτελείται όσο η συνθήκη είναι ψευδής

-Απαιτείται στην ομάδα εντολών να υπάρχουν εντολές μεταβολής τιμής της συνθήκης.

Στον μετρητή δίνουμε μια αρχική τιμή πριν την εντολή επανάληψης

3η επαναληπτική ΓΙΑ – ΑΠΟ - ΜΕΧΡΙ

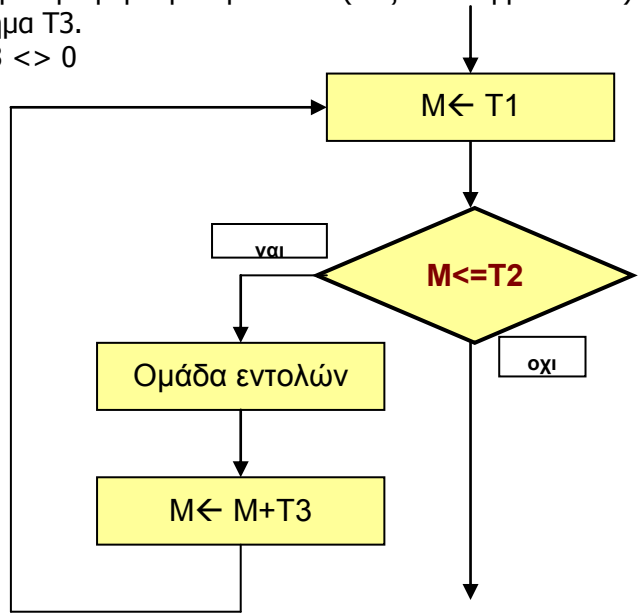
ΓΙΑ Μ ΑΠΟ Τ1 ΜΕΧΡΙ Τ2 ΜΕ_ΒΗΜΑ Τ3

Ομάδα εντολών

ΤΕΛΟΣ-ΕΠΑΝΑΛΗΨΗΣ

χαρακτηριστικά

1. Χρησιμοποιείται μόνο όταν είναι γνωστός ο αριθμός των επαναλήψεων
2. Η μεταβλητή Μ μεταβάλλεται (αυξάνεται ή μειώνεται) κατά βήμα Τ3.
3. Τ3 <> 0



Αν Τ3<0 η συνθήκη γίνεται: **Μ>=Τ2

2.4.4 ΕΜΦΩΛΕΥΜΕΝΕΣ ΔΙΑΔΙΚΑΣΙΕΣ

- **Όταν μια εντολή επιλογής (πλήρης ή περιορισμένη) εμπεριέχεται σε μία άλλη εντολή επιλογής, τότε η αλγοριθμική δομή καλείται εμφωλευμένη επιλογή.**
- Πολλαπλές επιλογές μπορούν να αναπαρασταθούν με μία εμφωλευμένη δομή.

Παράδειγμα:

```
Αλγόριθμος max (χ,ψ.z,μ ακέραιοι)
Δεδομένα // χ, ψ, Z //
        Αρχή
            Μ <- χ
        Αν ψ > μ τότε
            μ <- ψ
        Αλλιώς
                αν Z > μ τότε
                μ <- Z
                Τέλος αν
        Τέλος αν
            Εμφάνισε μ
        Τέλος.
```

Εμφωλευμένη επανάληψη έχουμε όταν στην ομάδα εντολών μιας επαναληπτικής δομής υπάρχει μία ή περισσότερες επαναληπτικές δομές.

```
ΟΣΟ συνθήκη ΕΠΑΝΑΛΑΒΕ
        Εντολή 1
        Εντολή 2
        ΟΣΟ συνθήκη ΕΠΑΝΑΛΑΒΕ
                Εντολή 3
                Εντολή 4
        ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
```

Παράδειγμα (εμφωλευμένης επανάληψης)

Να εκπονηθεί ένας αλγόριθμος για την εύρεση όλων των ακέραιων λύσεων της εξίσωσης: *3x+2y-7z=5*

```
ΑΛΓΟΡΙΘΜΟΣ διοφαντική_ανάλυση
ΔΕΔΟΜΕΝΑ  // χ, y, z //
ΓΙΑ x ΑΠΟ 0 ΜΕΧΡΙ 100 ΕΠΑΝΑΛΑΒΕ
   ΓΙΑ Y ΑΠΟ 0 ΜΕΧΡΙ 100 ΕΠΑΝΑΛΑΒΕ
        ΓΙΑ z ΑΠΟ 0 ΜΕΧΡΙ 100 ΕΠΑΝΑΛΑΒΕ
           ΑΝ (3*X+2*Y-7*Z) = 5 ΤΟΤΕ
                ΕΚΤΥΠΩΣΕ X, Y, Z
           ΤΕΛΟΣ ΑΝ
        ΤΕΛΟΣ  ΕΠΑΝΑΛΗΨΗΣ
   ΤΕΛΟΣ  ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ  ΕΠΑΝΑΛΗΨΗΣ
ΑΠΟΤΕΛΕΣΜΑΤΑ //Χ,Ψ,Ζ //
ΤΕΛΟΣ διοφαντική_ανάλυση
```

Δραστηριότητα: μετατροπή σε διάγραμμα ροής.

3.1 ΔΕΔΟΜΕΝΑ

- Τα δεδομένα είναι γεγονότα, μηνύματα, κωδικοποιημένα ή όχι και αποτελούν ακατέργαστο πληροφοριακό υλικό.
- Πληροφορία είναι το αποτέλεσμα που προκύπτει από την επεξεργασία δεδομένων. Ο αλγόριθμος είναι το μέσο για την παραγωγή πληροφορίας από τα δεδομένα.

Η πληροφορική μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

- Υλικού. Το υλικό επιτρέπει στα δεδομένα να αποθηκεύονται στην Κ.Μ. και στις περιφερειακές μονάδες αποθήκευσης, με διάφορες αναπαραστάσεις (ASCII. EBCDIC, UNICODE, ΕΛΟΤ 928 κ.λπ.).
- Γλωσσών προγραμματισμού. Τύποι μεταβλητών.
- Δομών δεδομένων. Δομή δεδομένων είναι ένα σύνολο υποθηκευμένων δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών (εγγραφή, αρχείο, πίνακας κ.τλ.).
- Ανάλυσης δεδομένων. Τρόποι καταγραφής και αλληλοσυσχέτισης, μελετώνται ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα (Βάσεις Δεδομένων, Μοντελοποίηση Δεδομένων, Αναπαράσταση Γνώσης).

3.2 ΑΛΓΟΡΙΘΜΟΙ + ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ = ΠΡΟΓΡΑΜΜΑ

- *Η αποθήκευση δεδομένων στην κύρια ή δευτερεύουσα μνήμη του υπολογιστή δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλ. σε μία δομή δεδομένων.*
- *Δομή δεδομένων* είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

Λειτουργίες επί των δομών δεδομένων:

- ✓ **Προσπέλαση** (access) είναι η πρόσβαση σε κάποιο κόμβο (μέλος της δομής) με σκοπό να εξετασθεί, να εκτυπωθεί ή να μεταβληθεί το περιεχόμενό του. Π.χ. αν σε ένα πίνακα Α αν αναφερθώ στον κόμβο Α[3] τότε έχω προσπέλαση στο τρίτο στοιχείο του πίνακα.
- ✓ **Εισαγωγή** (insertion) είναι η προσθήκη νέων κόμβων σε μια δομή δεδομένων. Π.χ. αν σε ένα πίνακα Α δώσω την εντολή Α[10] ← 100 τότε εισάγω ένα νέο κόμβο με τιμή εκατό στο δέκατο στοιχείο του πίνακα (υποθέτω ότι είναι κενό). Αν ο κόμβος υπάρχει ήδη (δεν είναι κενός) τότε μιλάμε για τροποποίηση (update).
- ✓ **Διαγραφή** (deletion) είναι η αφαίρεση ενός κόμβου από μια δομή δεδομένων.
- ✓ **Αναζήτηση** (search) είναι η προσπέλαση των κόμβων μίας δομής δεδομένων με σκοπό να βρεθούν ένας ή περισσότεροι κόμβοι με συγκεκριμένη ιδιότητα. Μετά την αναζήτηση μπορεί να εφαρμοστούν άλλες πράξεις (δηλ. εισαγωγή, τροποποίηση, διαγραφή, κλπ). Π.χ. είναι δυνατό να αναζητηθεί μία συγκεκριμένη τιμή με σκοπό αν υπάρχει να διαγραφεί.
- ✓ **Ταξινόμηση** (sort) δηλαδή αναδιάταξη των κόμβων μίας δομής κατά αύξουσα ή φθίνουσα σειρά.
- ✓ **Αντιγραφή**
- ✓ **Συγχώνευση**
- ✓ **Διαχωρισμός**

Οι πιο συνηθισμένες δομές είναι ο πίνακας (table ή array), η εγγραφή (record), η ουρά (queue), η στοίβα (stack), η λίστα (list), το δένδρο (tree) κλπ. Πρακτικά για να υλοποιήσουμε ένα πρόγραμμα σε μία γλώσσα προγραμματισμού (ή έστω ψευδογλώσσα) χρειαζόμαστε τόσο αλγορίθμους όσο και δομές δεδομένων (δηλ. ισχύει το Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα). Κάποιες δομές δεδομένων (π.χ. πίνακας, εγγραφή) υποστηρίζονται άμεσα (προσφέρονται έτοιμες προς δήλωση) από τις περισσότερες γλώσσες προγραμματισμού γενικής χρήσης (Pascal, Basic, C). Άλλες (στοίβα, ουρά, λίστα, δένδρο) υλοποιούνται έμμεσα μέσω δηλώσεων των δομών και μεταβλητών που υποστηρίζονται άμεσα. Φυσικά απαιτείται κι η ανάπτυξη κατάλληλων αλγορίθμων για κάθε λειτουργία (πράξη) επί των δομών αυτών.

Κατηγορίες δομών δεδομένων

Στατικές δομές δεδομένων.

☆Τα στοιχεία τους αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

☆Το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και όχι κατά την εκτέλεση του προγράμματος.

Δυναμικές δομές δεδομένων.

☆Τα στοιχεία τους δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης, αλλά στηρίζονται στην τεχνική δυναμικής παραχώρησης μνήμης.

☆Δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων του μεγαλώνει ή μικραίνει καθώς στη δομή εισάγονται ή διαγράφονται στοιχεία.

3.3 ΠΙΝΑΚΕΣ

Σε πολλά προβλήματα το πλήθος των υπό επεξεργασία δεδομένων είναι μεγάλο, οπότε και η επεξεργασία τους πολύπλοκη. Χρησιμοποιούμε πίνακες, όταν θέλουμε να επεξεργαστούμε ένα πλήθος ομοειδών στοιχείων. Τα στοιχεία ενός πίνακα είναι πάντα του ίδιου τύπου, π.χ. ακέραιοι αριθμοί, πραγματικοί αριθμοί, συμβολοσειρές κλπ.

Ας θεωρήσουμε του ακόλουθους δύο ορισμούς πίνακα:

πραγματικός πίνακας A[10]

πραγματικός πίνακας B[5, 7]

Η πρώτη δήλωση ορίζει ένα μονοδιάστατο πίνακα 10 θέσεων με το όνομα Α, ενώ η δεύτερη δήλωση ορίζει έναν πίνακα πραγματικών αριθμών δύο διαστάσεων, συνολικά 5x7=35 θέσεων.

Κατά την εκτέλεση ενός αλγόριθμου επεξεργαζόμαστε συνήθως ένα στοιχείο του πίνακα κάθε φορά. Στις εντολές λοιπόν των αλγορίθμων εμφανίζονται, εν γένει, τα στοιχεία του πίνακα και όχι όλος ο πίνακας. Για να συμβολίσουμε το στοιχείο με διατακτικό αριθμό ν ενός μονοδιάστατου πίνακα Α, γράφουμε Α[ν]. Αναφέρουμε δηλαδή το όνομα του πίνακα και το διατακτικό αριθμό της θέσης μέσα σε αγκύλες. Όμοια, με Β[κ,λ] συμβολίζουμε το στοιχείο με διατακτικούς αριθμούς κ,λ ενός δισδιάστατου πίνακα Β.

- Οι πίνακες αποτελούν **στατική δομή δεδομένων**.

Ορισμός **Πίνακας είναι ένα σύνολο στοιχείων ίδιου τύπου, τα οποία αναφέρονται με ένα κοινό όνομα. Η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες.**
Ένας πίνακας μπορεί να είναι μονοδιάστατος, δισδιάστατος, και γενικά ν-διάστατος.

Πλεονεκτήματα πινάκων

- Η χρήση πινάκων είναι ένας πολύ καλός τρόπος για τη διαχείριση πολλών δεδομένων του ίδιου τύπου.

Μειονεκτήματα πινάκων

- Απαιτούν μνήμη. Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης.
- Είναι στατικές δομές δεδομένων.

Στέλλα	Κατερίνα	Γιάννης	Γιώργος	Μαρία

Μονοδιάστατος αλφαριθμητικός πίνακας **NAME[5]**

	1η στήλη	*2η στήλη*	*3η στήλη*	*4η στήλη*	*5η στήλη*	*6η στήλη*	*7η στήλη*	*8η στήλη*	*9η στήλη*	*10η στήλη*
1η γραμμή	18	15	14	25	44	45	47	74	58	47
2η γραμμή	478	46	35	458	158	458	25	45	57	35
3η γραμμή	14	54	125	487	547	698	154	258	478	145

Δυσδιάστατος ακέραιος πίνακας Β[3,10]

Παράδειγμα χρήσης δομής δεδομένων «πίνακα».

Α) Έστω ότι θέλουμε να καταγράψουμε της πωλήσεις μιας εταιρείας.

	1	2	3	4	5	6	7	8	9	10	11	12

Β) Γενικεύουμε το πρόβλημα και θεωρούμε ότι η εταιρεία έχει 4 υποκαταστήματα και θέλουμε να μελετήσουμε τις πωλήσεις ενός 6μήνου.

	1	2	3	4	5	6
1						
2						
3						
4						

Γ) Έστω ότι θέλουμε να επεξεργαστούμε τους βαθμούς 5 μαθητών στο μάθημα της Πληροφορικής.

1	2	3	4	5

1	2	3	4	5

- **Εισαγωγή στοιχείων στους παραπάνω πίνακες.**
- **Ασκήσεις: 3.1.1, 1.3.2, 3.1.5, 3.1.4, ΔΤ1 σελ. 33.**
- **Υπολογισμός min, max σε δισδιάστατο πίνακα.**
- **Ασκήσεις για το σπίτι: 3.1.6, 3.1.7, 3.1.8, 3.1.9, 3.3.2, 3.3.3**

Περιγράψτε τις σκοπιές από τις οποίες μελετά τα δεδομένα η Πληροφορική
Δώστε τον ορισμό της δομής δεδομένων. Ποιες είναι οι βασικές λειτουργίες επί των δομών δεδομένων
Τι είναι οι δυναμικές δομές δεδομένων Τι είναι οι στατικές δομές δεδομένων

Μονοδιάστατοι Πίνακες - Άλυτες ασκήσεις

3.1.1. Να αναπτυχθεί αλγόριθμος που θα διαβάζει Ν αριθμούς και θα τους τοποθετεί στις αντίστοιχες θέσεις ενός πίνακα. Στη συνέχεια να υπολογίζεται και να εκτυπώνεται το πλήθος και το ποσοστό που καταλαμβάνουν στον πίνακα οι θετικοί, οι αρνητικοί αριθμοί καθώς και τα μηδενικά

3.1.2. Να αναπτυχθεί αλγόριθμος με δεδομένο έναν πίνακα Ν στοιχείων θα εκτυπώνει το πλήθος των άρτιων

3.1.3. Να αναπτυχθεί αλγόριθμος που θα διαβάζει έναν αριθμό μεγαλύτερο του 10, να διαβάζει αντίστοιχες ρίψεις ζαριού και να εμφανίζει την συχνότητα εμφάνισης κάθε αριθμού, ποιά ζαριά εμφανίστηκε περισσότερες φορές και ποιά λιγότερες

3.1.4. Να αναπτυχθεί αλγόριθμος που θα διαβάζει Ν αριθμούς σε έναν μονοδιάστατο πίνακα και στη συνέχεια θα υπολογίζει και θα εκτυπώνει το μέγιστο καθώς και τις θέσεις του πίνακα που αυτό εντοπίζεται. Μπορείτε να υλοποιήσετε τον αλγόριθμο με μια προσπέλαση του πίνακα

3.1.5. Να αναπτυχθεί αλγόριθμος που θα δημιουργεί μονοδιάστατο πίνακα τοποθετώντας την τιμή -1 στις περιττές θέσεις και 1 στις άρτιες

3.1.6. Ένας συλλέκτης γραμματοσήμων αποφάσισε να χρησιμοποιήσει τις εξής δομές δεδομένων για την διαχείριση της συλλογής του: πίνακα με όνομα ΘΕΜΑ που περιέχει το θέμα του κάθε γραμματοσήμου, πίνακα ΕΤΟΣ που. Να αναπτυχθεί αλγόριθμος που θα εκτυπώνει τα συλλεκτικά κομμάτια που υπάρχουν στη συλλογή και το πλήθος τους

3.1.7. Η εταιρεία "INFO" καταγράφει τα μηνιαία έσοδά της σε πίνακα ΕΣΟΔΑ 12 θέσεων και τα έξοδά της σε

πίνακα ΕΞΟΔΑ αντίστοιχα, ταυτόχρονα υπάρχει πίνακας ΜΗΝΑΣ 12 θέσεων που περιέχει τα ονόματα των μηνών. Να αναπτυχθεί αλγόριθμος που θα εκτυπώνει τον μήνα που:
i. Εμφανίστηκαν τα ελάχιστα έσοδα
ii. Εμφανίστηκαν τα μέγιστα έξοδα
iii. Εμφανίστηκαν τα μέγιστα κέρδη
Η χρονιά ήταν κερδοφόρα για την εταιρεία ή όχι
3.1.8. Να αναπτύξετε αλγόριθμο που θα καταγράφει σε δισδιάστατο πίνακα ΖΑΡΙΕΣ τις 500 ρίψεις ενός παιχνιδιού τάβλι. Στη συνέχεια ο αλγόριθμος πρέπει να υπολογίζει και να εκτυπώνει: α) πόσες διπλές καταγράφηκαν β) Πόσες ζαριές είχαν άθροισμα 11 γ) Τι ποσοστό των ζαριών ήταν ντόρτια (τεσσάρες)
3.1.9. Ο διαχειριστής μιας πολυκατοικίας χρησιμοποιεί τις ακόλουθες δομές δεδομένων για την διαχείριση των οικονομικών: μονοδιάστατος πίνακας ΟΝΟΜΑ με τα ονόματα των 25 ιδιοκτητών διαμερισμάτων της πολυκατοικίας και επίσης μονοδιάστατος πίνακας ΤΜ που περιέχει αντίστοιχα τα τετραγωνικά μέτρα του κάθε διαμερίσματος. Να αναπτύξετε αλγόριθμο που θα διαβάζει το συνολικό ποσό κοινοχρήστων και θα επιμερίζει τα έξοδα στους ιδιοκτήτες διαμερισμάτων με βάση τα τετραγωνικά μέτρα

Συνδυαστικές ασκήσεις Πίνακες - Άλυτες ασκήσεις

3.3.1. Η εταιρεία ΤΤΤ με 1000 εργαζομένους προέβει σε εξαγορά της εταιρείας ΡΤΡ που απασχολεί 250 άτομα. Για κάθε μια από τις δυο εταιρείες στη διάθεσή σας υπάρχουν δυο πίνακες: ο πίνακας ΟΝΟΜΑΤΑ με δυο στήλες με τα επώνυμα και τα ονόματα των εργαζομένων αντίστοιχα και ο πίνακας ΑΠΟΔΟΧΕΣ που περιέχει τους μισθούς τους αντίστοιχα. Να αναπτυχθεί αλγόριθμος όπου:
i. Θα δημιουργεί τους ενοποιημένους πίνακες ΟΝΟΜΑΤΑ_ΟΛΟΙ και ΑΠΟΔΟΧΕΣ_ΟΛΟΙ
ii. Θα εκτυπώνει τα ονόματα των 100 πιο καλά αμοιβομένων υπαλλήλων με φθίνουσα διάταξη
iii. Θα εκτυπώνει τα ονοματεπώνυμα όσων γιορτάζουν σήμερα (σήμερα π.χ. είναι της Αγ. Ειρήνης) και να τους αποδίδει μπόνους 5% επί του μισθού τους
3.3.2. Η εταιρεία ΔΣΣΔ με 450 εργαζομένους έχει καταχωρήσει τα στοιχεία τους σε δισδιάστατο πίνακα όπου στην 1η στήλη έχει καταχωρήσει τα ονόματά τους, στη 2η στήλη τις διευθύνσεις και στην 3η στήλη σε ποιο από τα 2 τμήματα της εταιρείας απασχολούνται. Να αναπτύξετε αλγόριθμο όπου, θα επεξεργάζεται τα στοιχεία και στη συνέχεια:
i. Να εκτυπώνει ταξινομημένα αλφαβητικά τα ονόματα και τις διευθύνσεις των υπαλλήλων της εταιρείας που εργάζονται στο 2ο τμήμα
ii. Να διαβάζει το όνομα ενός υπαλλήλου και να εκτυπώνει το τμήμα που απασχολείται
3.3.3. Για την εκπόνηση μιας εργασίας ένας φοιτητής στατιστικής καλείται να συγκεντρώσει από το διαδίκτυο τις θερμοκρασίες του περασμένου μήνα για 10 πόλεις στην Ελλάδα σε πίνακα ΘΕΡΜΟΚΡΑΣΙΕΣ[10,30], επιπρόσθετα υπάρχει πίνακας ΠΟΛΗ[10] που περιέχει τα ονόματα των πόλεων. Να αναπτύξετε αλγόριθμο που θα διαβάζει τα προαναφερθέντα στοιχεία και θα υπολογίζει και θα εκτυπώνει:
i. Την πιο θερμή πόλη το μήνα που πέρασε
ii. Ποιά ήταν η πόλη με τη χαμηλότερη θερμοκρασία την τελευταία μέρα του μήνα
iii. Ποιά μέρα και σε ποια πόλη σημειώθηκε η μεγαλύτερη αλλά και η μικρότερη θερμακρασία
iv. Ποιά μέρα σημειώθηκε η μεγαλύτερη % αύξηση θερμοκρασίας
3.3.4. Να αναπτυχθεί αλγόριθμος όπου θα αποθηκεύει σε έναν μονοδιάστατο πίνακα ΜΟΥΣΕΙΟ 120 θέσεων τα ονόματα ισάριθμων μουσείων και σε πίνακα ΕΠΙΣΚΕΨΕΙΣ[120,12] το πλήθος των επισκεπτών που δέχτηκαν τους περασμένους 12 μήνες. Να αναπτύξετε αλγόριθμο που:
i. θα διαβάζει το όνομα ενός μουσείου και να εκτυπώνει το πλήθος ημερήσιων επισκέψεων
ii. θα εκτυπώνει τα 10 μουσεία με τις περισσότερες επισκέψεις
3.3.5. Το Υπουργείο Πολιτισμού χρησιμοποιεί τις εξής δομές δεδομένων για την διαχείριση των Ελληνικών θεάτρων: τον πίνακα ΟΝΟΜΑ που περιέχει το όνομα του θεάτρου, τον πίνακα ΠΟΛΗ που περιέχει αντίστοιχα την πόλη που το κάθε θέατρο βρίσκεται καθώς και τον πίνακα ΧΩΡΗΤΙΚΟΤΗΤΑ που περιέχει το πλήθος των καθισμάτων κάθε θεάτρου. Να αναπτύξετε αλγόριθμο που με δεδομένα τα παραπάνω στοιχεία θα:
i. Έκτυπώνει την συνολική χωρητικότητα των θεάτρων στην Ελλάδα
ii. Διαβάζει το όνομα ενός θεάτρου και θα εκτυπώνει σε ποιες πόλεις υπάρχει ομώνυμο θέατρο (αν υπάρχει)
iii. Θα εκτυπώνει κατά αύξουσα διάταξη τα θέατρα με βάση το ονομά τους (αν υπάρχουν θέατρα με το ίδιο όνομα θα λαμβάνεται υπόψη το όνομα της πόλης)
3.3.6. Το δημοτικό parking διαθέτει 300 θέσεις για τα αυτοκίνητα και 50 για μοτοσικλέτες. Να αναπτύξετε αλγόριθμο που θα χρησιμοποιεί πίνακες ΑΥΤΟΚΙΝΗΤΑ, ΜΗΧΑΝΕΣ με αντίστοιχες θέσεις και να καταχωρεί στον κάθε έναν τον αριθμό κυκλοφορίας τους οχήματος/δικύκλου που εισέρχεται, σε θέση του πίνακα που αντιστοιχεί στη θέση που καταλαμβάνει το όχημα στο parking. Ο αλγόριθμος στη συνέχεια θα διαβάζει έναν αριθμό κυκλοφορίας και θα εκτυπώνει το είδος του οχήματος και τη θέση του στο parking και θα «αδειάζει» τη θέση διαγράφοντας τη θέση του πίνακα. Αν το όχημα δεν υπάρχει στο parking τότε θα ζητάει αριθμό θέσης, τύπο οχήματος και θα καταχωρεί τα στοιχεία στον αντίστοιχο πίνακα
3.3.7. Στο σχολικό πρωτάθλημα στίβου διεξάγεται το αγώνισμα του τριάθλου ως εξής: Οι αθλητές την ίδια

μέρα αγωνίζονται στον αγώνα 100 μέτρων, στα 1000 μέτρα και στο άλμα εις μήκος. Σε κάθε αγώνισμα ο αθλητής συγκεντρώνει βαθμούς ως εξής: Για τα 100 και τα 1000 μέτρα κάθε δευτερόλεπτο παίρνει "ποινή" 25 βαθμούς και για το άλμα εις μήκος κάθε εκατοστό αφαιρεί 15 βαθμούς. Να αναπτυχθεί αλγόριθμος που θα διαβάζει τα ονόματα των 120 μαθητών που συμμετείχαν στο πρωτάθλημα σε μονοδιάστατο πίνακα ΟΝΟΜΑ[120] και τις επιδόσεις τους σε τρισδιάστατο πίνακα ΕΠΙΔΟΣΕΙΣ[120, 3] όπου κάθε στήλη αντιστοιχεί σε ένα άθλημα με τη σειρά που αναφέρθηκαν παραπάνω. Ο αλγόριθμος στη συνέχεια πρέπει να εκτυπώνει τα ονόματα των αθλητών που έλαβαν τα μετάλλια
3.3.8. Στον τελικό της άρσης βαρών συμμετέχουν 20 αθλητές για τους οποίους καταχωρούμε τα ακόλουθα στοιχεία στους κάτωθι πίνακες: Πίνακας ΑΘΛΗΤΗΣ[20, 2] στην πρώτη στήλη του οποίου καταχωρούμε το όνομα και στη δεύτερη τη χώρα του, επίσης χρησιμοποιούμε πίνακα ΒΑΡΟΣ[20] με το βάρος του αντίστοιχου αθλητή καθώς και πίνακας ΕΠΙΔΟΣΗ[20, 3] με τις 3 επιδόσεις κάθε αθλητή. Να αναπτυχθεί αλγόριθμος που θα εκτυπώνει τα ονόματα και τις χώρες των αθλητών που θα πάρουν τα μετάλλια καθώς και την υπόλοιπη κατάταξη. Πρέπει να σημειωθεί ότι σε περίπτωση που κάποιοι αθλητές έχουν την ίδια επίδοση τότε τα κιλά τους καθορίζουν την τελική κατάταξη
3.3.9. Καταχωρούμε σε δισδιάστατο πίνακα ΧΩΡΙΑ τα ονόματα όλων των Ελληνικών χωριών (πρώτη στήλη) καθώς και το νομό που βρίσκονται (δεύτερη στήλη). Να αναπτύξετε αλγόριθμο που α) θα εκτυπώνει όλα τα χωριά ταξινομημένα με αύξουσα σειρά κατά όνομα – αν κάποια χωριά έχουν το ίδιο όνομα να λαμβάνεται υπόψη ο νομός, β) να διαβάζει το όνομα ενός χωριού και να εκτυπώνει πόσες φορές εντοπίζεται στην Ελλάδα
3.3.10. Η δανειστική βιβλιοθήκη του δήμου Τενεούπολης χρησιμοποιεί τις εξής δομές: Δισδιάστατος πίνακας ΒΙΒΛΙΑ του οποίου οι γραμμές αντιστοιχούν στα 25.000 βιβλία που υπάρχουν, η πρώτη στήλη περιέχει τον τίτλο κάθε βιβλίου, η δεύτερη στήλη περιέχει τον συγγραφέα του κάθε βιβλίου και η τρίτη το θέμα - περιγραφή του βιβλίου. Να αναπτύξετε αλγόριθμου που α) θα διαβάζει το όνομα ενός συγγραφέα και θα εκτυπώνει τα βιβλία του που υπάρχουν (αν υπάρχουν) στη βιβλιοθήκη , β) θα διαβάζει ένα θέμα και θα επιστρέφει τα βιβλία που υπάρχουν (αν υπάρχουν) με αυτό το θέμα στη βιβλιοθήκη
3.3.11. Οι καθηγητές πληροφορικής του ιδιωτικού εκπαιδευτηρίου "Ακαδημία Πλάτωνα" της Τενεούπολης συλλέγουν υλικό από το διαδίκτυο για το μάθημα ΑΕΠΠ της Γ' Λυκείου. Για να διατηρήσουν ένα αρχείο για το υλικό αυτό χρησιμοποιούν τους εξής μονοδιάστατους πίνακες: ΕΚΦΩΝΗΣΗ που περιέχει τις εκφωνήσεις των ασκήσεων που έχουν εντοπίσει, ΚΕΦΑΛΑΙΟ που περιέχει αντίστοιχα το κεφάλαιο κάθε άσκησης και τον πίνακα ΠΡΟΕΛΕΥΣΗ που περιέχει την URL διεύθυνση απ' όπου προέρχεται η άσκηση. Να αναπτυχθεί αλγόριθμος που:
i. Να διαβάζει τον αριθμό των ασκήσεων που υπάρχουν (μέγιστος αριθμός 10.000)
ii. Να διαβάζει τα απαραίτητα στοιχεία και να τα αποθηκεύει στους παραπάνω πίνακες
iii. Να ταξινομεί τις ασκήσεις με βάση την προέλευση και να εκτυπώνει τις εκφωνήσεις και το κεφάλαιο iv. Να διαβάζει μια διεύθυνση "προέλευσης" της άσκησης και το επιθυμητό κεφάλαιο και να δημιουργεί νέο πίνακα με τις εκφωνήσεις αυτές.

ΕΡΓΑΣΤΗΡΙΟ

1. Να αναπτυχθεί αλγόριθμος που θα διαβάζει άγνωστο πλήθος αριθμών και θα εντοπίζει και εκτυπώνει το ποσοστό αυτών που είναι πολλαπλάσια του 5. Ο αλγόριθμος θα τερματίζεται όταν εισαχθεί ο αριθμός 0.

2. Ο μισθός του κύριου Χ είναι 1250 €, ενώ σύμφωνα με το μισθολόγιο αυξάνεται κατά 11% ετησίως. Κάθε μήνα έχει αποφασίσει να αποταμιεύει 9% του μισθού για το όνειρό του που είναι η αγορά φουσκωτού σκάφους. Να αναπτύξετε αλγόριθμο που θα υπολογίζει και θα εκτυπώνει σε πόσους μήνες θα κατορθώσει να προβεί στην αγορά του φουσκωτού αξίας 7000 €.

3.4 ΣΤΟΙΒΑ

Η στοίβα είναι μια δομή δεδομένων που υλοποιείται με χρήση μονοδιάστατου πίνακα.
- Η στοίβα χρησιμοποιεί τη μέθοδο επεξεργασίας LIFO τελευταίο μέσα – πρώτο έξω.
- Η βοηθητική μεταβλητή ή δείκτης top χρησιμοποιείται για να δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας.

Οι κύριες λειτουργίες της είναι δύο:
- **Ώθηση στοιχείου στη στοίβα (PUSH).**
  1. Γίνεται έλεγχος αν η στοίβα είναι γεμάτη αλλιώς συμβαίνει **<u>υπερχείλιση</u>** (overflow).
  2. Η μεταβλητή top αυξάνεται κατά ένα.
  3. Εισάγεται το στοιχείο.

- **Απώθηση στοιχείου από τη στοίβα (POP).**
  1. Γίνεται έλεγχος αν υπάρχει τουλάχιστον ένα στοιχείο στη στοίβα αλλιώς συμβαίνει **<u>υποχείλιση</u>** (underflow).
  2. Εξέρχεται το στοιχείο που δείχνει η μεταβλητή top.
  3. Η μεταβλητή top μειώνεται κατά ένα.

3.5 ΟΥΡΑ

Η ουρά είναι μια δομή δεδομένων που υλοποιείται με χρήση μονοδιάστατου πίνακα.

- Η ουρά χρησιμοποιεί τη μέθοδο FIFO πρώτο μέσα – πρώτο έξω.
- Στην ουρά χρησιμοποιούνται δύο δείκτες ο **εμπρός (front)** που δίνει τη θέση του στοιχείου που θα εξαχθεί από την ουρά, και ο δείκτης **πίσω (rear)** που μας δείχνει το στοιχείο που μόλις εισήλθε στην ουρά.

Οι κύριες λειτουργίες της είναι δύο:

- **Εισαγωγή στοιχείου στο πίσω άκρος της ουράς (Enqueue).**
  1. Γίνεται έλεγχος υπάρχουν κενές θέσεις στον πίνακα.
  2. Αυξάνεται ο δείκτης rear κατά ένα.
  3. Εισάγεται το στοιχείο.
- **Εξαγωγή στοιχείου από την ουρά (Dequeue).**
  1. Γίνεται έλεγχος αν υπάρχει τουλάχιστον ένα στοιχείο στην ουρά.
  2. Εξέρχεται το στοιχείο που δείχνει ο δείκτης front.
  3. Ο δείκτης front αυξάνεται κατά ένα, για να δείχνει το επόμενο στοιχείο που πρόκειται να εξαχθεί.

3.1 Αναζήτηση

Υπάρχουν δύο βασικοί αλγόριθμοι αναζήτησης ενός στοιχείου ή κόμβου (με συγκεκριμένο περιεχόμενο) σ' ένα πίνακα. Η **σειριακή** (sequential) ή γραμμική αναζήτηση κι η **δυαδική** (binary) αναζήτηση. Απλούστερη αλλά και πιο χρονοβόρα είναι η σειριακή αναζήτηση (παρ. 3.6 σελ. 64) αφού απαιτεί αναζήτηση των στοιχείων με τη σειρά προσπέλαση των κόμβων του πίνακα μέχρι να εντοπιστεί ο κόμβος που θέλουμε. Αν το στοιχείο που αναζητούμε δεν υπάρχει τότε πρέπει να προσπελαστούν όλοι οι κόμβοι του πίνακα.

3.2 Ταξινόμηση

Υπάρχουν αρκετοί αλγόριθμοι ταξινόμησης (συνήθως κατά αύξουσα σειρά) των κόμβων ενός πίνακα. Απλούστερη αλλά και πιο χρονοβόρα είναι η **ταξινόμηση φυσαλίδας** (παρ. 3.7 σελ. 68). Η λογική αυτού του αλγορίθμου είναι η εξής: Αρχικά το μικρότερο στοιχείο όλων τοποθετείται στην πρώτη θέση του πίνακα. Στη συνέχεια, το δεύτερο μικρότερο στοιχείο τοποθετείται στην δεύτερη θέση του πίνακα, κοκ μέχρις ότου ταξινομηθεί όλος ο πίνακας.

Για καλύτερη κατανόηση του αλγορίθμου έστω μη ταξινομημένος πίνακας 9 θέσεων (σελ. 65 σχολικού βιβλίου). <u>Για κάθε</u> κόμβο του πίνακα εκτελείται η παρακάτω επαναληπτική διαδικασία. Ξεκινάμε την επαναληπτική διαδικασία από το **δεύτερο** κόμβο του πίνακα (θα δούμε παρακάτω γιατί). Αρχικά συγκρίνουμε μεταξύ τους τα περιεχόμενα του $9^{ου}$ και του $8^{ου}$ κόμβου. Αν το περιεχόμενο του $9^{ου}$ κόμβου είναι μικρότερο τότε αντιμεταθέτουμε τα περιεχόμενα των δύο κόμβων. Κατόπιν συγκρίνουμε μεταξύ τους τα περιεχόμενα του $8^{ου}$ και του $7^{ου}$ κόμβου. Αν το περιεχόμενο του $8^{ου}$ κόμβου είναι μικρότερο τότε αντιμεταθέτουμε τα περιεχόμενα των δύο κόμβων, κοκ. Είναι προφανές ότι κάθε φορά προωθούνται μπροστά τα μικρότερα στοιχεία. Τελικά συγκρίνουμε μεταξύ τους τα περιεχόμενα του $2^{ου}$ και του $1^{ου}$ κόμβου. Αν το περιεχόμενο του $2^{ου}$ κόμβου είναι μικρότερο τότε αντιμεταθέτουμε τα περιεχόμενα των δύο κόμβων. Με τον τρόπο αυτό το μικρότερο στοιχείο όλων τοποθετείται στην πρώτη θέση του πίνακα. Να γιατί η επαναληπτική διαδικασία ξεκίνησε από το δεύτερο κόμβο! Συνεχίζουμε την επαναληπτική αυτή διαδικασία από τον τρίτο κόμβο του πίνακα. Εξυπακούεται ότι τα περιεχόμενα των κόμβων έχουν αλλάξει με σκοπό την ταξινόμηση όλων των κόμβων. Συγκρίνουμε λοιπόν πάλι μεταξύ τους τα περιεχόμενα του $9^{ου}$ και του $8^{ου}$ κόμβου, κοκ. Τελικά συγκρίνουμε μεταξύ τους τα περιεχόμενα του $3^{ου}$ και του $2^{ου}$ κόμβου. Προφανώς με τον τρόπο αυτό το δεύτερο μικρότερο στοιχείο όλων τοποθετείται στη δεύτερη θέση του πίνακα, κοκ.

Η **αντιμετάθεση** δύο μεταβλητών είναι μία λεπτή λειτουργία. Π.χ. έστω δύο μεταβλητές x, y όπου x = 10 και y = 20. Η αντιμετάθεσή τους θα πρέπει να έχει ως αποτέλεσμα x = 20 και y = 10. Θα υπέθετε κανείς ότι με τις απλές εντολές x ← y και μετά y ← x θα μπορούσε να γίνει η αντιμετάθεση. Όμως με την πρώτη εντολή το x θα πάρει την τιμή 10 και με τη δεύτερη εντολή το y θα πάρει κι αυτό την τιμή 10 αφού άλλαξε η τιμή

του x! Γι' αυτό μας χρειάζεται να χρησιμοποιήσουμε μια βοηθητική μεταβλητή έστω temp. Με τη σειριακή εκτέλεση των εντολών temp ← x, x ← y, y ← temp η αντιμετάθεση γίνεται σωστά αφού η temp `κρατάει` την αρχική τιμή του x.

ΕΡΩΤΗΣΕΙΣ

1. Περιγράψτε τη δομή της στοίβας, καθώς και τις κύριες λειτουργίες της. Τι σημαίνει δομή LIFO
2. Περιγράψτε τη δομή της ουράς, καθώς και τις κύριες λειτουργίες της. Τι σημαίνει δομή FIFO
3. Τι είναι η σειριακή (γραμμική) μέθοδος αναζήτησης Σε ποιες περιπτώσεις δικαιολογείται η χρήση της Δώστε τον αντίστοιχο αλγόριθμο
4. Δώστε τον ορισμό της ταξινόμησης (διάταξης)
5. Τι σημαίνει δομή δεδομένων δευτερεύουσας μνήμης
6. Δώστε τον αλγόριθμο της ταξινόμησης ευθείας ανταλλαγής

4.1 Ανάλυση προβλημάτων

Η λύση σε ένα πρόβλημα μπορεί να προέλθει από ποικίλες και διαφορετικές προσεγγίσεις, τεχνικές και μεθόδους. Έτσι είναι απαραίτητο να γίνεται μια καλή ανάλυση του κάθε προβλήματος και να προτείνεται συγκεκριμένη μεθοδολογία και ακολουθία βημάτων.

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

- ✓ Καταγραφή των **δεδομένων** του προβλήματος
- ✓ Την αναγνώριση των ιδιαιτεροτήτων του προβλήματος
- ✓ Την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησης του προβλήματος
- ✓ Πρόταση επίλυσης με χρήση κάποιας μεθόδου
- ✓ Την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

Κατά την ανάλυση ενός προβλήματος θα πρέπει να δοθεί απάντηση στις επόμενες ερωτήσεις:

- ✓ Ποια τα **δεδομένα** και το **μέγεθος** του προβλήματος
- ✓ Ποιες οι συνθήκες που πρέπει να πληρούνται
- ✓ Ποια η πλέον αποδοτική **μέθοδος** επίλυσης
- ✓ Πως θα καταγραφεί η λύση
- ✓ Ποιος ο τρόπος **υλοποίησης** στο συγκεκριμένο υπολογιστικό σύστημα

Οι μέθοδοι ανάλυσης και επίλυσης προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον, για τους παρακάτω λόγους:

- ✓ Παρέχουν ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας
- ✓ Μπορούν να αναπαρασταθούν με κοινές δομές δεδομένων και ελέγχου
- ✓ Παρέχουν τη δυνατότητα καταγραφής των χρονικών και χωρικών απαιτήσεων της μεθόδου επίλυσης έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

4.1 Μέθοδοι σχεδίασης αλγορίθμων

Πρόκειται για θεωρητικές προσεγγίσεις που χρησιμοποιούνται κατά τη διάρκεια της φάσης της ανάλυσης.

Κάθε τεχνική χρειάζεται να υποστηρίζει τα εξής:

- ✓ Να αντιμετωπίζει με το δικός τρόπο τα δεδομένα
- ✓ Να έχει τη δική της ακολουθία εντολών
- ✓ Να διαθέτει τη δική της αποδοτικότητα

Υπάρχουν πολλές τεχνικές σχεδίασης αλγορίθμων. Οι πιο βασικές είναι:

- ✓ **Μέθοδος Διαίρει και βασίλευε**
- ✓ **Μέθοδος Δυναμικού προγραμματισμού**
- ✓ **Άπληστη μέθοδος.**

6.1 Η έννοια του προγράμματος

Η επίλυση ενός προβλήματος με Η/Υ περιλαμβάνει τρία στάδια:

- ✓ Τον ακριβή προσδιορισμό του προβλήματος.
- ✓ Την ανάπτυξη του αντίστοιχου αλγορίθμου.
- ✓ Τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον Η/Υ.

- **Ο προγραμματισμός**, ασχολείται με τη δημιουργία του συνόλου των εντολών που πρέπει να δοθούν στον Η/Υ, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος.
- Βασικό στοιχείο του προγράμματος είναι τα ***δεδομένα*** και οι ***δομές δεδομένων*** επί των οποίων ενεργεί.
- **Ο προγραμματισμός** είναι αυτός που δίνει την εντύπωση ότι οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν πολύπλοκα προβλήματα. Αυτό βέβαια, είναι μια ψευδαίσθηση μιας και ο Η/Υ μπορεί μόνο να:
  - ✓ Να αποθηκεύει ακολουθίες δυαδικών ψηφίων.
  - ✓ Να ανακτά τις ακολουθίες δυαδικών ψηφίων.
  - ✓ Να κάνει στοιχειώδεις αριθμητικές πράξεις.
  - ✓ Να κάνει συγκρίσεις.

6.2 Ιστορική αναδρομή

Μία πρώτη ταξινόμηση των γλωσσών προγραμματισμού είναι ανάλογα με το κριτήριο της γενιάς στην οποία ανήκουν (επίπεδο γλώσσας). Έτσι έχουμε:

- ✓ **Γλώσσες μηχανής** (machine languages).
- ✓ χαμηλού επιπέδου που είναι οι **συμβολικές γλώσσες** (assembly).
- ✓ **Γλώσσες 3ης γενιάς** ή **υψηλού επιπέδου** που είναι οι γλώσσες Fortran, Cobol, Basic, Algol, Pascal, PL/1, C, C++, Java, κλπ.
- ✓ **Γλώσσες 4ης γενιάς** που είναι γλώσσες ερωταπαντήσεων, γλώσσες εφαρμογών βάσεων δεδομένων (π.χ. SQL) κλπ.

6.2.1 Γλώσσα μηχανής

ορισμός: **Ένα πρόγραμμα σε γλώσσα μηχανής** είναι μια ακολουθία από δυαδικά ψηφία, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες.

- Η γλώσσα μηχανής είναι άμεσα αντιληπτή από τον υπολογιστή και απ' ευθείας εκτελέσιμη. Αποτελείται από μία ακολουθία δυαδικών ψηφίων (δηλ. 0 και 1). Προφανώς είναι πολύ δύσχρηστη και επίπονη στη χρήση γι' αυτό χρησιμοποιούνταν στους πρώτους υπολογιστές πριν ακόμα αναπτυχθούν γλώσσες υψηλότερου επιπέδου. Η γλώσσα μηχανής είναι άμεσα **εξαρτημένη** από την αρχιτεκτονική (υλικό ή μηχανή) του υπολογιστή στον οποίο εκτελείται. Γράφοντας προγράμματα σε γλώσσα μηχανής «μιλάμε» απ' ευθείας με το υλικό του συγκεκριμένου υπολογιστή . Δύο διαφορετικοί υπολογιστές (δηλ. με διαφορετικό επεξεργαστή) δεν έχουν την ίδια γλώσσα μηχανής. Επομένως τα προγράμματα που έχουν γραφεί σε γλώσσα μηχανής δεν είναι καθόλου εύκολα μεταφέρσιμα σε άλλους υπολογιστές.

00110010 00000011 10000001	→	Βάλε τον αριθμό 3, σε μία συγκεκριμένη θέση μνήμης
00110010 00000100 10000010	→	Βάλε τον αριθμό 4, σε μία συγκεκριμένη θέση μνήμης
01011100 00000011 10000001	→	Πρόσθεσε τις δύο θέσεις μνήμης

6.2.2 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου

- Οι εντολές σε **συμβολική γλώσσα** αποτελούνται από συμβολικά ονόματα (π.χ ADD) που αντιστοιχούν σε εντολές τις γλώσσας μηχανής. Το έργο της μετάφρασης αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής.

Μειονεκτήματα:

- ✓ Παραμένουν συνδεδεμένες με την αρχιτεκτονική κάθε Η/Υ.
- ✓ Δεν διέθεταν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμ-

ματα, που ήταν δύσκολο να γραφούν και να συντηρηθούν.
- ✓ Τα προγράμματα δεν μπορούσαν να μεταφερθούν σε άλλο Η/Υ ακόμα και του ίδιου κατασκευαστή (χαμηλού επιπέδου).

Οι **συμβολικές γλώσσες** είναι μία εξέλιξη των γλωσσών μηχανής. Χρησιμοποιούν συντομογραφίες και σύμβολα (της αγγλικής γλώσσας) που είναι πιο κατανοητές στον άνθρωπο. Έχουν τη δυνατότητα να μετατρέπουν εσωτερικά τις εντολές στην ισοδύναμη ακολουθία δυαδικών ψηφίων (δηλ. 0 και 1) αφού διαθέτουν **συμβολομεταφραστή** (assembler). Μία εντολή συμβολικής γλώσσας αντιστοιχεί σε μία αρκετά μεγάλη ακολουθία δυαδικών αριθμών. Η συμβολική γλώσσα παραμένει άμεσα **εξαρτημένη** από την αρχιτεκτονική (υλικό ή μηχανή) του υπολογιστή στον οποίο εκτελείται. Γράφοντας προγράμματα σε συμβολική γλώσσα πάλι 'μιλάμε' απ' ευθείας με το υλικό του συγκεκριμένου υπολογιστή (π.χ. με τη μνήμη). Δύο διαφορετικοί υπολογιστές (δηλ. με διαφορετικό επεξεργαστή) δεν έχουν κατ' ανάγκη την ίδια συμβολική γλώσσα. Επομένως τα προγράμματα που έχουν γραφεί σε συμβολική γλώσσα δεν είναι εύκολα μεταφέρσιμα σε άλλους υπολογιστές. Π.χ.

LDA 3 $01 → Βάλε τον αριθμό 3, σε μία συγκεκριμένη θέση μνήμης (π.χ. θέση μνήμης 1)
LDA 4 $02 → Βάλε τον αριθμό 4, σε μία συγκεκριμένη θέση μνήμης (π.χ. θέση μνήμης 2)
ADD $01 $02 → Πρόσθεσε τις δύο θέσεις μνήμης (ουσιαστικά τα περιεχόμενά τους)

6.2.3 Γλώσσες υψηλού επιπέδου

Οι γλώσσες υψηλού επιπέδου είναι πολύ πιο απλές και κατανοητές στον άνθρωπο. Έχουν επίσης τη δυνατότητα να μετατρέπουν εσωτερικά τις εντολές στην ισοδύναμη ακολουθία δυαδικών ψηφίων (δηλ. 0 και 1) αφού διαθέτουν **μεταγλωττιστή** (compiler) ή **διερμηνευτή** (interpreter). Οι γλώσσες υψηλού επιπέδου είναι **ανεξάρτητες** από την αρχιτεκτονική (υλικό) του υπολογιστή στον οποίο εκτελούνται.

Χρονολογία δημιουργίας	**Εταιρεία**	**Όνομα γλώσσας προγραμματισμού**	**Χαρακτηριστικά**
1957	IBM	FORTRAN (FORmula TRANslation)	Για την επίλυση μαθηματικών και επιστημονικών προβλημάτων.
1960		COBOL (Common Business Oriented Language)	Για την ανάπτυξη εμπορικών εφαρμογών.
1960		ALGOL (ALGOrithmic Language)	Για την ανάπτυξη γενικής φύσης εφαρμογών.
Στα μέσα της δεκαετίας του '60		PL/1 (Program Language 1)	Προσπάθησε να καλύψει επιστημονικούς και εμπορικούς τομείς, χωρίς επιτυχία.
Στα μέσα της δεκαετίας του '60	MIT	LISP (List Processor)	Για το χειρισμό λιστών από σύμβολα. (προβλήματα τεχνητής νοημοσύνης)
Αρχές δεκαετίας του '70	MIT	PROLOG (PROgramming LOGic)	Προβλήματα τεχνητής νοημοσύνης (έμπειρα συστήματα, παιχνίδια, επεξεργασία φυσικών γλωσσών κ.α.)
	MICROSOFT	BASIC (Beginner' s All purpose Symbolic Instruction Code)	Συμβολικός κώδικας εντολών γενικής χρήσης για αρχάριους. Σχεδιάστηκε για τη συγγραφή γρήγορων προγραμμάτων τα οποία εκτελούνται με διερμηνευτή. Η τυποποίηση της από τη Microsoft με τις εκδόσεις Quick Basic και Visual Basic την καθιέρωσε ως πρότυπο για την ανάπτυξη εφαρμογών στους προσωπικούς υπολογιστές.
1970	Wirth (καθηγητής)	PASCAL	Γλώσσα γενικής χρήσης, κατάλληλη για την εκπαίδευση και τη δημιουργία ισχυρών προγραμμάτων. Χαρακτηριστικό της είναι η καταλληλότητα της για τη δημιουργία δομημένων προγραμμάτων. Αποτέλεσε τη βάση για την ADA KAI Modula-2m

	BELL	C	Αρχικά χρησιμοποιήθηκε για τη δημιουργία του λειτουργικού συστήματος UNIX. Είναι κατάλληλη για τη δημιουργία δομημένων προγραμμάτων, παράλληλα διαθέτει πολλές δυνατότητες γλώσσας χαμηλού επιπέδου.
		C++	Αντικειμενοστραφής γλώσσα προγραμματισμού.
	SUN	JAVA	Αντικειμενοστραφής γλώσσα προγραμματισμού. Χρησιμοποιείται για την ανάπτυξη εφαρμογών που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλ. σε διαφορετικούς Η/Υ οι οποίοι είναι συνδεδεμένοι στο Internet. Αυτά τα προγράμματα εκτελούνται από διαφορετικά PCs ή μεγάλους Η/Υ με διαφορετικά λειτουργικά συστήματα, χωρίς αλλαγές.

Πλεονεκτήματα των γλωσσών υψηλού επιπέδου

- Ο φυσικότερος και ποιο «ανθρώπινος» τρόπος έκφρασης προβλημάτων.
- Η ανεξαρτησία από τον τύπο του Η/Υ. Προγράμματα σε μια γλώσσα υψηλού επιπέδου μπορούν να εκτελεσθούν σε οποιονδήποτε υπολογιστή, με ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της μεταφερσιμότητας είναι σημαντικό προσόν.
- Η ευκολία στην εκμάθηση.
- Η διόρθωση λαθών και η συντήρηση προγραμμάτων γίνεται εύκολα.

6.2.4 Γλώσσες 4ης γενιάς

Οι γλώσσες υψηλού επιπέδου (3ης γενιάς) απευθύνονταν μόνο σε προγραμματιστές. Ο χρήστης δεν είχε τη δυνατότητα να κάνει αλλαγές σε κάποιο πρόγραμμα. Αντίθετα στις Γλώσσες 4ης γενιάς ο χρήστης, έχει τη δυνατότητα, σχετικά εύκολα,

- ✓ να υποβάλλει ερωτήσεις στο σύστημα,
- ✓ να αναπτύξει εφαρμογές που ανακτούν πληροφορίες από βάσεις δεδομένων και
- ✓ να καθορίζει τον ακριβή τρόπο εμφάνισης των πληροφοριών.

Οι γλώσσες 4ης γενιάς είναι ακόμα περισσότερο απλές και κατανοητές στον άνθρωπο, γι' αυτό δεν απευθύνονται αποκλειστικά και μόνο σε προγραμματιστές αλλά ακόμα και σε απλούς χειριστές υπολογιστών.

Ταξινόμηση γλωσσών προγραμματισμού

- Διαδικασιακές ή αλγοριθμικές γλώσσες → Επιτρέπουν την υλοποίηση αλγορίθμων.
- Αντικειμενοστραφείς γλώσσες.
- Συναρτησιακές γλώσσες.
- Μη διαδικασιακές γλώσσες ή γλώσσες πολύ υψηλού επιπέδου π.χ. PROLOG.
- Γλώσσες ερωτοαπαντήσεων π.χ. SQL

Ταξινόμηση γλωσσών με βάση την περιοχή χρήσης τους.

- Γλώσσες γενικής χρήσης.
  - ✓ Γλώσσες επιστημονικής κατεύθυνσης.
  - ✓ Γλώσσες εμπορικής κατεύθυνσης
- Γλώσσες προγραμματισμού συστημάτων.
- Τεχνητής νοημοσύνης
- Γλώσσες ειδικής χρήσης
  - ✓ Χρησιμοποιούνται σε ειδικές περιοχές εφαρμογών, όπως στα γραφικά με Η/Υ, στη ρομποτική, στη σχεδίαση ολοκληρωμένων κυκλωμάτων, στην εκπαίδευση μέσω Η/Υ κ.α.

Ταξινόμηση των γλωσσών προγραμματισμού είναι ανάλογα με το κριτήριο του τρόπου προ-

γραμματισμού (ιδέα-φιλοσοφία στην οποία βασίζονται). Έτσι έχουμε:

- ✓ Γλώσσες **αλγοριθμικές** ή **διαδικασιακές** (π.χ. Fortran, Cobol, Pascal, Basic, C, κλπ). Πρόκειται για τις πιο παλαιές και 'κλασσικές' γλώσσες. Ξεκινάμε τον προγραμματισμό από τις λειτουργίες (ή διαδικασίες) που έχουμε να υλοποιήσουμε. Αφού τις αναλύσουμε, για κάθε διαδικασία φτιάχνεται κι ένας αλγόριθμος. Τα δεδομένα είναι κατά κάποιο τρόπο πιο δευτερεύοντα αφού ορίζονται και χρησιμοποιούνται μέσα στις διαδικασίες.
- ✓ Γλώσσες **αντικειμενοστραφείς** (π.χ. C++, Smalltalk, Java, κλπ). Εδώ ξεκινάμε τον προγραμματισμό από τα δεδομένα που έχουμε στο συγκεκριμένο πρόβλημα. Αφού αναλυθούν τα δεδομένα, μετά την κατάλληλη μορφοποίηση δομούνται ως αντικείμενα. Οι διαδικασίες είναι κατά κάποιο τρόπο πιο δευτερεύουσες αφού ορίζονται και χρησιμοποιούνται μέσα στα αντικείμενα (λειτουργίες μεταξύ αντικειμένων).
- ✓ Γλώσσες **συναρτησιακές** (π.χ. Lisp). Η Lisp, που ανήκει στο χώρο της τεχνητής νοημοσύνης, βασίζεται στη δομή δεδομένων της λίστας. Όλες οι λειτουργίες στη γλώσσα αυτή γίνονται μέσω συναρτήσεων μεταξύ λιστών.
- ✓ Γλώσσες **μη διαδικασιακές** ή **λογικές** (π.χ. Prolog). Συνήθως αυτές είναι γλώσσες πάρα πολύ υψηλού επιπέδου. Πολλές από αυτές ανήκουν στο χώρο της τεχνητής νοημοσύνης. Οι φιλοσοφία τους είναι αντίθετη με τον διαδικασιακό (αλγοριθμικό) προγραμματισμό. Προσπαθούν να προσομοιώσουν την ανθρώπινη λογική στους υπολογιστές. Για παράδειγμα ένα έμπειρο σύστημα (όπως ένα σύστημα πρόγνωσης καιρού) μπορεί να 'μαθαίνει' με την πάροδο του χρόνου όπως ο άνθρωπος.
- ✓ Γλώσσες **ερωταπαντήσεων** (π.χ. SQL). Πολλές από αυτές ανήκουν στο χώρο των βάσεων δεδομένων. Ο χρήστης μπορεί να υποβάλλει ερωτήματα στον υπολογιστή και να παίρνει τις αναγκαίες πληροφορίες με την επιθυμητή μορφή.
- ✓ Γλώσσες **οπτικές** (visual) Π.χ. Visual Basic, Visual C++, java. Οι κλασσικές διαδικασιακές γλώσσες δεν εκμεταλλεύονται τα νέα γραφικά περιβάλλοντα επικοινωνίας μεταξύ ανθρώπου-υπολογιστή (π.χ. Windows). Οι οπτικές γλώσσες διαθέτουν ένα μεγάλο πλούτο από λειτουργίες εισόδου/ εξόδου (δεδομένων/ πληροφοριών) και μας επιτρέπουν να δημιουργούμε όλο το περιβάλλον ενός προγράμματος με εύχρηστο και γραφικό τρόπο (πλαίσια διαλόγου, μενού, παράθυρα, κλπ).
- ✓ Γλώσσες **οδηγούμενες από το γεγονός** όπου λειτουργίες του προγράμματος ενεργοποιούνται με την εκτέλεση ενός γεγονότος, π.χ. την επιλογή μιας εντολής από ένα μενού ή πάτημα ενός κουμπιού.
- ✓ Γλώσσες **παράλληλες** (π.χ. Occam, Parallel C)**.** Εκμεταλλεύονται αρχιτεκτονικές με περισσότερους από έναν επεξεργαστές που λειτουργούν παράλληλα. Η απόδοση (ταχύτητα) στα συστήματα αυτά έτσι αυξάνει κατακόρυφα.

Μία τρίτη ταξινόμηση των γλωσσών προγραμματισμού είναι ανάλογα με το κριτήριο της περιοχής χρήσης (σκοποί, χρήσεις στις οποίες εφαρμόζονται). Έτσι έχουμε:

- ✓ Γλώσσες **γενικής χρήσης** (π.χ. Basic, Pascal). Είναι αρκετά καλές κι αποτελεσματικές για μία ευρεία γκάμα εφαρμογών. Βέβαια εδώ μπορούμε να εντάξουμε (θεωρητικά) τόσο τις γλώσσες **επιστημονικής κατεύθυνσης** (π.χ. Fortran) όσο και τις γλώσσες **εμπορικής κατεύθυνσης** (π.χ. Cobol).
- ✓ Γλώσσες **προγραμματισμού συστημάτων** (π.χ. C). Πολλές συσκευές (π.χ. video) ή τεχνολογικά συστήματα (καρτοτηλέφωνα, μηχανήματα αυτόματων τραπεζικών συναλλαγών, μηχανήματα ΠΡΟΠΟΛΟΤΤΟ, κλπ) εκτελούν προγράμματα όπως περίπου ο υπολογιστής. Επομένως χρειάζονται κάποια γλώσσα προγραμματισμού που να διαθέτει και αρετές χαμηλού επιπέδου για να εκμεταλλεύεται το συγκεκριμένο υλικό (μηχάνημα).
- ✓ Γλώσσες **τεχνητής νοημοσύνης** (π.χ. Prolog, Lisp). Μία αδυναμία των παραδοσιακών γλωσσών είναι ότι διαφέρουν κατά πολύ από την ανθρώπινη λογική. Οι γλώσσες τεχνητής νοημοσύνης προσπαθούν κατά κάποιο τεχνητό τρόπο να προσομοιώσουν την ανθρώπινη νοημοσύνη. Χρησιμοποιούνται σε παιχνίδια (π.χ. σκάκι), έμπειρα συστήματα, κλπ.
- ✓ Γλώσσες **ειδικής χρήσης**. Είναι γλώσσες που χρησιμοποιούνται για πολύ εξειδικευμένες εφαρμογές (π.χ. σχεδίαση-γραφικά, εκπαίδευση, κλπ).

Στον παρακάτω πίνακα φαίνονται οι πιο γνωστές γλώσσες προγραμματισμού με τα πιο βασικά χαρακτηριστικά τους. Η παράθεσή τους είναι περίπου χρονολογική.

ΟΝΟΜΑ ΓΛΩΣΣΑΣ	**ΓΕΝΙΑ Ή ΕΠΙΠΕΔΟ**	**ΤΡΟΠΟΣ ΠΡΟΓΡ/ΣΜΟΥ**	**ΧΡΗΣΗ**	**ΠΑΡΑΤΗΡΗΣΕΙΣ**
Μηχανής	$1^{η}$, Πολύ χαμηλό			Χρησιμοποιήθηκε στους πρώτους Η/Υ
Συμβολική	$2^{η}$, Χαμηλό			Μη δομημένη (επιβάλλεται η χρήση GOTO)
Fortran	$3^{η}$,Υψηλό	Αλγοριθμικός	Επιστημονική	Πρώτη γλώσσα υψηλού

				επιπέδου
Cobol	3η,Υψηλό	Αλγοριθμικός	Εμπορική	Χρησιμοποιείται σχεδόν σε κάθε κατηγορία υπολογιστή
Algol	3η,Υψηλό	Αλγοριθμικός	Γενική	Απέτυχε εμπορικά αλλά επηρέασε πολύ άλλες
Lisp	3η – 4η	Συναρτησιακός	Τεχνητή νοημοσύνη	Χρησιμοποιεί τη δομή της λίστας
Prolog	4η, Πολύ Υψηλό	Λογικός (Μη διαδικασιακός)	Τεχνητή νοημοσύνη (έμπειρα συστήματα)	
Logo	3η,Υψηλό	Μη διαδικασιακός ή οπτικός	Ειδικής χρήσης Εκπαιδευτική	Υπάρχει και οπτική υλοποίησή της
PL/1	3η,Υψηλό	Αλγοριθμικός	Γενική	Απέτυχε εμπορικά
Basic	3η,Υψηλό	Αλγοριθμικός	Γενική	Απλή και για αρχαρίους
Pascal	3η,Υψηλό	Αλγοριθμικός	Γενική	Καθιέρωσε το δομημένο προγραμματισμό
C	3η,Υψηλό	Αλγοριθμικός	Γενική, Επιστημονική, Προγραμματισμός συστημάτων	Πολύ ισχυρή, με δυνατότητες χαμηλού επιπέδου
C++	3η,Υψηλό	Αντικειμενοστραφής	Γενική, Επιστημονική, Προγραμματισμός συστημάτων	Πολύ ισχυρή, με δυνατότητες χαμηλού επιπέδου Επέκταση της C
SQL	4η, Πολύ Υψηλό	Ερωταπαντήσεων	Εμπορική Βάσεις Δεδομένων	
dBASE, Clipper, Access	4η, Πολύ Υψηλό	Ερωταπαντήσεων	Εμπορική Βάσεις Δεδομένων	
Occam	3η,Υψηλό	Παράλληλος	Ειδική	
Parallel C	3η,Υψηλό	Παράλληλος	Ειδική	Επέκταση της C
Visual C	3η,Υψηλό	Οπτικός	Γενική, Επιστημονική, Προγραμματισμός συστημάτων	Επέκταση της C
Visual C++	3η,Υψηλό	Οπτικός και Αντικειμενοστραφής	Γενική, Επιστημονική, Προγραμματισμός συστημάτων	Επέκταση της C
Visual Basic	3η, Υψηλό	Οπτικός	Γενική	Επέκταση της Basic
Java	3η,Υψηλό	Αντικειμενοστραφής	Διαδίκτυο	

Ποια είναι η καλύτερη γλώσσα προγραμματισμού

- Μία γλώσσα προγραμματισμού που είναι αντικειμενικά καλύτερη από κάποια άλλη, δεν υπάρχει ούτε πρόκειται να υπάρξει.

Η επιλογή μιας γλώσσας προγραμματισμού για την ανάπτυξη μιας εφαρμογής εξαρτάται από:

- ✓ Το υπολογιστικό περιβάλλον στο οποίο θα εκτελεστεί.
- ✓ Τα προγραμματιστικά περιβάλλοντα που διαθέτουμε.
- ✓ Το είδος της εφαρμογής.
- ✓ Τις γνώσεις του προγραμματιστή.

6.3 Φυσικές και τεχνητές γλώσσες

Μια γλώσσα προσδιορίζεται από το :

- **Το αλφάβητο** → το σύνολο των στοιχείων που χρησιμοποιεί η γλώσσα.
- **Το λεξιλόγιο** → αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία της αλφαβήτου.
- **Τη γραμματική** → **τυπικό**, είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια λέξη είναι αποδεκτή.
- **Συντακτικό**
- **Τη σημασιολογία**

Διαφορές φυσικών και τεχνητών γλωσσών

- Η δυνατότητα εξέλιξης τους.
  Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημιουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν, και αυτό γιατί χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπων, που εξελίσσονται και αλλάζουν ανάλογα τις εποχές και τον κοινωνικό περίγυρο.
- Οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα, αφού κατασκευάζονται για ένα συγκεκριμένο σκοπό.
  Ωστόσο συχνά οι γλώσσες προγραμματισμού **βελτιώνονται και μεταβάλλονται**, *με σκοπό*
  - ✓ να Βελτιώσουν αδυναμίες
  - ✓ να καλύψουν μεγαλύτερο εύρος εφαρμογών
  - ✓ να ακολουθήσουν νέες εξελίξεις

  Οι γλώσσες προγραμματισμού αλλάζουν *σε επίπεδο*
  - ✓ διαλέκτου και
  - ✓ επέκτασης

6.4 τεχνικές σχεδίασης προγραμμάτων

Πρόκειται για προσεγγίσεις που χρησιμοποιούνται στο τέλος της φάσης της ανάλυσης για να αποτυπωθεί ο σχεδιασμός της επίλυσης. Οι πιο βασικές τεχνικές είναι:

- **Ιεραρχική σχεδίαση** προγράμματος. Προκύπτει συνήθως μετά τη φάση της 'Διαίρει και βασίλευε' ανάλυσης ή από επάνω προς τα κάτω – top/ down.
- **Τμηματικός προγραμματισμός**. Υλοποιεί την ιεραρχική σχεδίαση. Κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα.
- **Δομημένος προγραμματισμός**. Βασίζεται στη χρήση των δομών ακολουθίας, επιλογής, επανάληψης και μόνο. Αποθαρρύνει την ανεξέλεγκτη χρήση των εντολών GOTO.

Πλεονεκτήματα δομημένου προγραμματισμού

- ✓ Δημιουργία απλούστερων προγραμμάτων.
- ✓ Άμεση μεταφορά των αλγορίθμων σε πρόγραμμα.
- ✓ 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 αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεί ενός προγράμματος τα δεδομένα από τα οποία με κατάλληλη μορφοποίηση δημιουργούνται τα αντικείμενα.

6.6 Παράλληλος προγραμματισμός

Χρησιμοποιείται σε συστήματα που διαθέτουν παραπάνω από έναν επεξεργαστές οι οποίοι μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα εκτελώντας διαφορετικές εντολές του ίδιου προγράμματος.

6.7 Προγραμματιστικά περιβάλλοντα

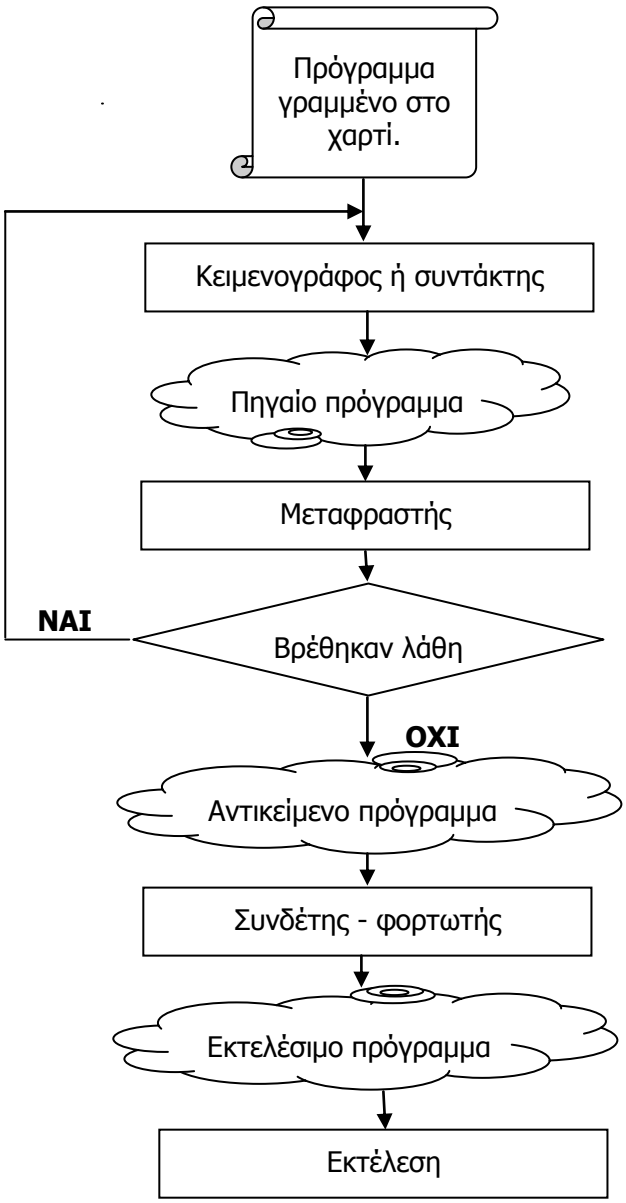
Σε κάθε προγραμματιστικό περιβάλλον υπάρχουν τα κατάλληλα εργαλεία που βοηθούν τον προγραμματιστή στην εργασία του. Η εργασία αυτή αρχίζει με τη **σύνταξη** (γράψιμο) του προγράμματος (κώδικα) και ολοκληρώνεται με την εκτέλεσή του. Πιο συγκεκριμένα σε ένα παραδοσιακό προγραμματιστικό περιβάλλον ισχύουν τα ακόλουθα:

- Σχεδόν όλες οι γλώσσες προγραμματισμού διαθέτουν ένα ειδικό εργαλείο (πρόγραμμα) που ονομάζεται **συντάκτης**. Ο συντάκτης είναι απλά ένας κειμενογράφος που επιτρέπει την αρχική σύνταξη του προγράμματος και τη διόρθωσή του στη συνέχεια (π.χ. σβήσιμο χαρακτήρων-εντολών). Το σύνολο των εντολών της γλώσσας προγραμματισμού (δηλ. ο αρχικός κώδικας) που γράφουμε ονομάζεται **πηγαίο πρόγραμμα** (ή πηγαίος κώδικας).
- το πηγαίο πρόγραμμα δε μπορεί να εκτελεστεί, αφού είναι απαραίτητη η 'μετάφρασή' του σε γλώσσα μηχανής. Για να γίνει αυτή η 'μετάφραση, του **μεταφραστή** (compiler) και του **διερμηνευτή** (interpreter).
  Πολλές γλώσσες χρησιμοποιούν μεταφραστή (π.χ. Pascal). Αρχικά από όλο το πηγαίο (source) πρόγραμμα παράγεται μετά τη μετάφραση ένα ενδιάμεσο πρόγραμμα, που ονομάζεται αντικείμενο (object) πρόγραμμα. Για να δημιουργηθεί το αντικείμενο πρόγραμμα η μετάφραση πρέπει να είναι επιτυχής (δηλ. δεν πρέπει να υπάρχουν συντακτικά λάθη στον πηγαίο κώδικα). Αν υπάρχουν λάθη, κατάλληλα μηνύματα μας βοηθούν να τα διορθώσουμε.
- Το αντικείμενο πρόγραμμα είναι κατανοητό από τον υπολογιστή (σε μορφή δυαδικού συστήματος) αλλά δεν είναι δυνατό να εκτελεστεί. Είναι απαραίτητο να **συνδεθεί** με άλλα τμήματα κώδικα που βρίσκονται στη βιβλιοθήκη της γλώσσας. Αυτή τη λειτουργία την αναλαμβάνει ο **συνδέτης-φορτωτής** (linker-loader) που παράγει και το τελικό **εκτελέσιμο** (executable) πρόγραμμα .

Επομένως σε ένα σύστημα με μεταφραστή, η διαδικασία είναι:

σύνταξη → μετάφραση → σύνδεση → εκτέλεση (σχήμα 1).

Τα απαραίτητα εργαλεία είναι ο **συντάκτης**, ο ***μεταφραστής*** κι ο ***συνδέτης*** και τα αρχεία που δημιουργούνται είναι το **πηγαίο**, το ***αντικείμενο*** και το ***εκτελέσιμο***. Αν γίνουν αλλαγές ή διορθώσεις στον πηγαίο κώδικα, η διαδικασίες της μετάφρασης και σύνδεσης πρέπει να επαναληφθούν μέχρι να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.



Σχήμα1: <u>Διαδικασία μετάφρασης προγράμματος.</u>

Ορισμένες άλλες γλώσσες χρησιμοποιούν **διερμηνευτή ή διερμηνέα** (π.χ. Basic). ***Η διερμηνεία του πηγαίου προγράμματος (κώδικα) γίνεται ως εξής (σχήμα 2)***

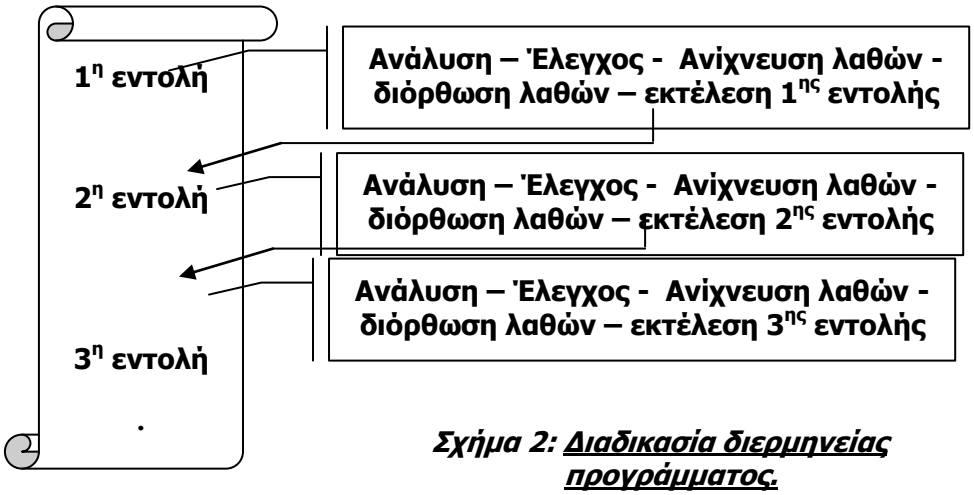
- ✓ **Ανάλυση – Έλεγχος - Ανίχνευση λαθών - διόρθωση λαθών - εκτέλεση 1ης εντολής.**
- ✓ **Ανάλυση – Έλεγχος - Ανίχνευση λαθών - διόρθωση λαθών - εκτέλεση 2ης εντολής.**

……

- ✓ **Ανάλυση – Έλεγχος - Ανίχνευση λαθών - διόρθωση λαθών - εκτέλεση Ν εντολής.**

Μετά από κάθε εντολή του πηγαίου (source) κώδικα γίνεται η 'μετάφραση' . Αν υπάρχουν λάθη στην εντολή, κατάλληλα μηνύματα μας βοηθούν να τα διορθώσουμε. Αφού διορθωθούν όλα τα συντακτικά λάθη η εντολή εκτελείται.

Πηγαίος κώδικας



Σχήμα 2: <u>Διαδικασία διερμηνείας προγράμματος.</u>

.

Διαφορές διερμηνευτή και μεταγλωττιστή

- ✓ Η βασική διαφορά του διερμηνευτή (interpreter) με το μεταγλωττιστή (compiler) είναι ότι η διερμηνεία γίνεται εντολή-εντολή, ενώ η μεταγλώττιση γίνεται για όλο το πρόγραμμα. Για να καταλάβουμε τη διαφορά μεταξύ του μεταφραστή και του διερμηνέα ας δούμε κάτι ανάλογο από τη ζωή μας. Ο άνθρωπος-μεταφραστής παίρνει ένα κείμενο και αφού το μεταφράσει ολόκληρο, μας επιστρέφει το μεταφρασμένο κείμενο στη γλώσσα που του ζητήσαμε. Αντίθετα, ο άνθρωπος-διερμηνέας βρίσκεται μεταξύ δύο συνομιλητών και κάθε πρόταση που ακούει από τον ένα τη μεταγλωττίζει στον άλλο. Έτσι, ακόμα που μερικές προτάσεις ή και ολόκληρη η συζήτηση επαναληφθεί από την αρχή, είναι αναγκασμένος να ξανακάνει τη μεταγλώττιση.
- ✓ Σε συστήματα με διερμηνευτή δε δημιουργείται αντικείμενο πρόγραμμα.
- ✓ Σε συστήματα με διερμηνευτή δεν υπάρχει συνδέτης-φορτωτής και μετά από τη διερμηνεία το πρόγραμμα μπορεί να εκτελεστεί άμεσα. Επομένως σε ένα σύστημα με διερμηνευτή, η διαδικασία είναι σύνταξη → διερμηνεία → εκτέλεση. Τα απαραίτητα εργαλεία είναι ο συντάκτης κι ο διερμηνευτής και το μόνο αρχείο που δημιουργείται είναι το πηγαίο. Δηλαδή δεν παράγεται εκτελέσιμο αρχείο που σημαίνει ότι ακόμα κι αν δεν έχουν γίνει αλλαγές ή διορθώσεις στον πηγαίο κώδικα, **<u>η διαδικασία της διερμηνείας επιβάλλεται πάντα πριν να εκτελεστεί το πρόγραμμα.</u>**
- ➢ Η τεχνική του <u>μεταφραστή έχει το μειονέκτημα</u> των διαδικασιών μετάφρασης και σύνδεσης (χρονική καθυστέρηση). Όμως το παραγόμενο εκτελέσιμο πρόγραμμα είναι ταχύτερο και μάλιστα μεταφέρσιμο.

Σήμερα σε πολλά προγραμματιστικά περιβάλλοντα υπάρχουν και μεικτές υλοποιήσεις όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος, και μεταφραστή για την τελική έκδοση του.

7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ

Στις επόμενες παραγράφους θα χρησιμοποιηθεί ως γλώσσα προγραμματισμού η **ΓΛΩΣΣΑ**. Φυσικά δε πρόκειται για μία υπαρκτή γλώσσα προγραμματισμού. Η ΓΛΩΣΣΑ είναι καθαρά εκπαιδευτική και ελάχιστα διαφέρει από την ψευδογλώσσα που χρησιμοποιήθηκε στις προηγούμενες παραγράφους. Έχοντας μελετήσει την έννοια των φυσικών και τεχνητών γλωσσών (βλ. §6.3) και τα προγραμματιστικά περιβάλλοντα (βλ. §6.7), δεν είναι δύσκολο να κατανοηθεί η ΓΛΩΣΣΑ που είναι πιο 'αυστηρή' από την ψευδογλώσσα ειδικά στη δήλωση των μεταβλητών.

Στο αλφάβητο της ΓΛΩΣΣΑΣ περιλαμβάνονται:

- ✓ όλα τα γράμματα (πεζά, κεφαλαία, ελληνικά, λατινικά),
- ✓ τα ψηφία (0-9)
- ✓ και ορισμένοι ειδικοί χαρακτήρες (+ - * / = κλπ).

7.2 Τύποι δεδομένων

Η ΓΛΩΣΣΑ υποστηρίζει τους βασικούς τύπους δεδομένων δηλαδή **ακεραίους, πραγματικούς, χαρακτήρες (κείμενο-αλφαριθμητικά) και λογικούς (Αληθής, Ψευδής).**

7.3 Σταθερές

Η ΓΛΩΣΣΑ υποστηρίζει την αντιστοίχιση σταθερών τιμών με ονόματα. Βέβαια, τέτοιες σταθερές πρέπει να δηλωθούν στην αρχή του προγράμματος. Π.χ.

ΣΤΑΘΕΡΕΣ

ΠΙ = 3.14

7.4 Μεταβλητές

Η ΓΛΩΣΣΑ υποστηρίζει τη χρήση μεταβλητών των τεσσάρων τύπων που αναφέρθηκαν (βλ. §7.2). Βέβαια, επισημαίνεται ότι στη ΓΛΩΣΣΑ **όλες οι μεταβλητές πρέπει να δηλώνονται υποχρεωτικά** στην αρχή του προγράμματος στο τμήμα δηλώσεων. Αυτή είναι μία βασική διαφορά από ην ψευδογλώσσα. Ασφαλώς στη ΓΛΩΣΣΑ δε χρησιμοποιούνται τα επιπλέον (πρόσθετα) στοιχεία της ψευδογλώσσας δηλαδή οι λέξεις αλγόριθμος, δεδομένα, αποτελέσματα. Π.χ. οι δηλώσεις μεταβλητών στη ΓΛΩΣΣΑ γίνονται ως εξής:

ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: ποσό
ΑΚΕΡΑΙΕΣ: τιμή
ΧΑΡΑΚΤΗΡΕΣ: όνομα, επίθετο
ΛΟΓΙΚΕΣ: έγγαμος

Σημειώνεται ότι στη ΓΛΩΣΣΑ οι λέξεις ΣΤΑΘΕΡΕΣ, ΜΕΤΑΒΛΗΤΕΣ, ΠΡΑΓΜΑΤΙΚΕΣ, ΑΚΕΡΑΙΕΣ ΧΑΡΑΚΤΗΡΕΣ, ΛΟΓΙΚΕΣ είναι δεσμευμένες.

7.5 Αριθμητικοί τελεστές

Η ΓΛΩΣΣΑ υποστηρίζει τους αριθμητικούς τελεστές **+, -, *, /, ^ (ύψωση σε δύναμη), DIV, (ακέραια διαίρεση), MOD (υπόλοιπο ακέραιας διαίρεσης)**. Π.χ. 3^2 είναι τρία στο τετράγωνο δηλ. 9. Ακόμα 7 DIV 2 = 3 (προσοχή όχι 3,5!) και 7 MOD 2 = 1 αφού το πηλίκο της ακέραιας διαίρεσης είναι 3 και το υπόλοιπο 1.

7.6 Συναρτήσεις

Η ΓΛΩΣΣΑ υποστηρίζει πολλές μαθηματικές συναρτήσεις:

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

7.7 Αριθμητικές εκφράσεις

Στη ΓΛΩΣΣΑ οι αριθμητικές εκφράσεις δε διαφέρουν από αυτές της ψευδογλώσσας. Σε μία τέτοια έκφραση μπορεί να υπάρχουν **αριθμητικές σταθερές, μεταβλητές, συναρτήσεις, τελεστές και παρενθέσεις**. Π.χ μία αριθμητική έκφραση μπορεί να είναι:

(((2 * x) + (9 * y)) ^ 2) / Τ_Ρ(a)

Μεγάλη προσοχή πρέπει να δοθεί στην **προτεραιότητα** των πράξεων.

7.8 Εντολή εκχώρησης

ΒΙΒΛΙΟ ΚΑΘΗΓΗΤΗ ΣΕΛ 62 Στη ΓΛΩΣΣΑ η εντολή εκχώρησης ή απόδοσης τιμής δε διαφέρει από αυτή της ψευδογλώσσας (δηλ. χρησιμοποιείται πάλι το σύμβολο ←). Βέβαια σε καμία περίπτωση η εντολή αυτή δε πρέπει να εκλαμβάνεται ως εξίσωση. Π.χ.

x ← y ^ 2 + 7
όνομα ← 'Ιωάννου' παντρεμένος ← αληθής

7.9 Εντολές εισόδου-εξόδου

Στη ΓΛΩΣΣΑ οι εντολές εισόδου εξόδου ελάχιστα διαφέρουν από αυτές της ψευδογλώσσας. Για την εισαγωγή δεδομένων χρησιμοποιείται η εντολή **ΔΙΑΒΑΣΕ**, ενώ για την εμφάνιση των αποτελεσμάτων η εντολή **ΓΡΑΨΕ** (αντί ΕΜΦΑΝΙΣΕ της ψευδογλώσσας). Π.χ.

ΔΙΑΒΑΣΕ όνομα, ηλικία
ΓΡΑΨΕ όνομα, ' είσαι ', ηλικία, 'ετών'

7.10 Δομή προγράμματος

Στη ΓΛΩΣΣΑ <u>κάθε</u> πρόγραμμα έχει την παρακάτω δομή:

<u>ΠΡΟΓΡΑΜΜΑ</u> όνομα_προγράμματος
[**<u>ΣΤΑΘΕΡΕΣ</u>**
.....]
[**<u>ΜΕΤΑΒΛΗΤΕΣ</u>**
.....]
<u>ΑΡΧΗ</u>
.....
.....
<u>ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ</u>

Οι λέξεις που είναι έντονες και με κεφαλαία είναι **δεσμευμένες** λέξεις. Στο τμήμα των σταθερών δηλώνουμε τις σταθερές τιμές με ονόματα αν υπάρχουν. Στο τμήμα των μεταβλητών δηλώνουμε τις μεταβλητές του προγράμματος. Στο τμήμα μεταξύ των λέξεων **ΑΡΧΗ** και **ΤΕΛΟΣ** γράφουμε τις εντολές του προγράμματος.

8.1 Εντολές επιλογής

Στη ΓΛΩΣΣΑ υποστηρίζονται οι εντολές επιλογής **ΑΝ** και **ΕΠΙΛΕΞΕ** όπως ακριβώς και στην ψευδογλώσσα.

Λογική έκφραση

Για τη σύνταξη μιας λογικής έκφραση ή συνθήκης χρησιμοποιούνται:
Σταθερές - μεταβλητές
αριθμητικές πράξεις → (^ , +, -, *, /, DIV, MOD)
συγκριτικοί τελεστές → (=,>, <, >=, <=, <>)
λογικοί τελεστές → (ΚΑΙ (AND), Ή (OR), ΟΧΙ (NOT))

Παραδείγματα απλών λογικών εκφράσεων ή συνθηκών :
Name = 'Maρία' → όπου το Name είναι μια αλφαριθμητική μεταβλητή.
AGE >= 18
Κόστος_1 <= Τιμή → όπου Κόστος_1 και Τιμή είναι αριθμητικές μεταβλητές.
Ποσοστό >= ((Χ+500)*0.18)→ όπου Ποσοστό και Χ είναι αριθμητικές μεταβλητές.

Σύνθετες εκφράσεις

Όταν χρειάζεται να συνδυαστούν δύο ή περισσότερες συνθήκες τότε χρησιμοποιούμε τις λογικές πράξεις **ΚΑΙ (AND), Ή (OR), ΟΧΙ (NOT).**

Παραδείγματα σύνθετων λογικών εκφράσεων ή συνθηκών :

Name = 'Maρία' **ΚΑΙ** AGE >= 18
Κόστος_1 <= Τιμή **Ή** Ποσοστό >= ((Χ+500)*0.18)

8.1.1 Η Εντολή ΑΝ

Η εντολή χρησιμοποιεί τέσσερις βοηθητικές λέξεις (***αν, τότε, αλλιώς, τέλος_αν***). Αρχίζει με τη βοηθητική λέξη ***αν*** και τελειώνει με τη βοηθητική λέξη ***τέλος_αν***. Μετά το ***αν*** ακολουθεί η ***συνθήκη***, η οποία τελειώνει πριν από τη δεσμευμένη λέξη ***τότε***.

αν <συνθήκη> **τότε**
ομάδα εντολών 1
αλλιώς
ομάδα εντολών 2
τέλος_αν

Η δομή επιλογής εκτελείται ως εξής: Αρχικά ελέγχεται η συνθήκη. Στη συνέχεια υπάρχουν δύο δυνατές περιπτώσεις:

- ✓ Αν η συνθήκη είναι αληθής, εκτελείται μόνο η ομάδα εντολών 1. Η ομάδα εντολών 2 αγνοείται και ο έλεγχος του αλγόριθμου βγαίνει από τη δομή επιλογής.
- ✓ Σε αντίθετη περίπτωση, αν δηλαδή η συνθήκη είναι ψευδής, δε λαμβάνεται υπόψη η ομάδα εντολών 1. Εκτελείται μόνο η ομάδα εντολών 2 και η εκτέλεση της δομής επιλογής τελειώνει.

Υπάρχει περίπτωση σε μια εντολή επιλογής, στη περίπτωση που ισχύει η συνθήκη, να εκτελείται μια σειρά εντολών, ενώ στην αντίθετη περίπτωση να μη θέλουμε να εκτελεστεί κάποια εντολή. Τότε χρησιμοποιούμε τη δομή περιορισμένης επιλογής.

αν <συνθήκη> **τότε**

ομάδα εντολών 1

τέλος_αν

Εντολές ΕΠΙΛΕΞΕ & ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ ΑΝ

Η εντολή χρησιμοποιεί τις βοηθητικές λέξεις (**επίλεξε, περίπτωση, περίπτωση αλλιώς και τέλος επιλογών**) αρχίζει με τη βοηθητική λέξη επίλεξε και τελειώνει με τη βοηθητική λέξη τέλος_επιλογών. Μετά το επίλεξε ακολουθεί μια μεταβλητή ή μία έκφραση.

επίλεξε έκφραση **περίπτωση** λίστα_τιμών1 εντολή/ες 1 **περίπτωση** λίστα_τιμών 2 Εντολή/ες 2 … **περίπτωση** λίστα_τιμών ν Εντολή/ες ν **Περίπτωση_αλλιώς** Εντολή/ες_αλλιώς **τέλος_επιλογών**	Με παρόμοιο τρόπο μπορούμε να χρησιμοποιήσουμε την πολλαπλή δομή επιλογής:	**αν** <συνθήκη_1> **τότε** εντολή/ες 1 **αλλιώς_αν** <συνθήκη_2> **τότε** εντολή/ες 2 …… **αλλιώς_αν** <συνθήκη_ν> **τότε** εντολή/ες ν **αλλιώς** εντολή/ες _αλλιώς **τέλος_αν**

ΕΜΦΩΛΕΥΜΕΝΗ ΕΠΙΛΟΓΗ

Όταν μια εντολή της μορφής αν …. Τότε …αλλιώς περιέχεται σε μία άλλη εντολή της μορφής αν …τότε έχουμε εμφωλευμένη επιλογή.

8.2 Εντολές επανάληψης

όσο <συνθήκη> **επανάλαβε**
ομάδα εντολών
Τέλος_επανάληψης

Εκτελείται η ομάδα εντολών όσο η συνθήκη είναι αληθής.

8.2.2 Εντολή Μέχρις_ότου

αρχή_επανάληψης

ομάδα εντολών

μέχρις_ότου <συνθήκη τέλους>

Σε αυτή τη μορφή της εντολής επανάληψης ο έλεγχος γίνεται μετά την εκτέλεση της ομάδας εντολών. **Αυτό σημαίνει ότι η ομάδα εντολών εκτελείται τουλάχιστον μία φορά**.

8.2.3 Εντολή ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ

Όταν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων, μπορεί να χρησιμοποιηθεί η τρίτη μορφή της επαναληπτικής δομής, η οποία έχει την ακόλουθη μορφή:
για <μετρητής> **από** <αρχική τιμή> **μέχρι** <τελική τιμή> ***με βήμα*** <βήμα μεταβολής (του μετρητή)>
ομάδα εντολών
τέλος_επανάληψης

Αυτή η μορφή της επαναληπτικής δομής έχει το χαρακτηριστικό ότι δεν απαιτείται στην ομάδα εντολών να υπάρχουν εντολές μεταβολής της τιμής του μετρητή, η οποία μεταβάλλεται από την ίδια την εντολή σε κάθε επανάληψη κατά βήμα. Όταν το βήμα είναι 1, εννοείται και δεν αναφέρεται καθόλου.

9.1 Μονοδιάστατοι πίνακες

Η μεταβλητές με δείκτες στη ΓΛΩΣΣΑ υλοποιούνται με τη δομή δεδομένων του πίνακα.

Στο τμήμα δηλώσεων, δηλώνεται ο τύπος το όνομα και το μέγεθος του πίνακα. πχ. ένας μονοδιάστατος πίνακας ακεραίων 20 θέσεων, δηλώνεται ως εξής:

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: βαθμοί [20]

Η διάσταση του πίνακα καθορίζεται από το πλήθος των δεικτών που χρησιμοποιούμε για να αναφερθούμε σε κάθε στοιχείο του. Ο δείκτης πίνακα είναι μια ακέραια έκφραση.

Πίνακας είναι ένα σύνολο αντικειμένων του ίδιου τύπου, τα οποία ορίζονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχειά του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο από ένα ή περισσότερους δείκτες.

9.2 Πότε πρέπει να χρησιμοποιούνται πίνακες

Η χρήση πινάκων εξυπηρετεί ή συχνά είναι απαραίτητη όταν θέλουμε να χειριστούμε ***πολλά δεδομένα του ίδιου τύπου,*** τα οποία πρέπει ***να διατηρούνται στη μνήμη μέχρι το τέλος εκτέλεσης του προγράμματος.***

Τα μειονεκτήματα των πινάκων είναι τα εξής:

- ✓ **Οι πίνακες απαιτούν μνήμη**. Με τη δήλωσή του, κάθε πίνακας δεσμεύει στατικά πολλές θέσεις μνήμης. Επομένως, η χρήση πολλών και μεγάλων πινάκων, μπορεί να οδηγήσει και σε ***αδυναμία εκτέλεσης του προγράμματος***.
- ✓ **Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος**. Οι πίνακες ως στατικές δομές έχουν σταθερό μέγεθος που δηλώνεται στην αρχή του προγράμματος.

9.3 Πολυδιάστατοι πίνακες

Στη ΓΛΩΣΣΑ υποστηρίζεται η δομή του πίνακα πολλών διαστάσεων.

ΜΕΤΑΒΛΗΤΕΣ

ΠΡΑΓΜΑΤΙΚΕΣ: Θ[3,20,12], Θ1[20,12]

9.4 Τυπικές επεξεργασίες πινάκων

- ✓ Υπολογισμός αθροισμάτων στοιχείων πίνακα
- ✓ Εύρεση του μέγιστου ή του ελάχιστου στοιχείου
- ✓ Ταξινόμηση στοιχείων πίνακα
- ✓ Αναζήτηση ενός στοιχείου πίνακα
- ✓ Συγχώνευση δύο πινάκων

10.1 Τμηματικός προγραμματισμός

Ο καλύτερος τρόπος για να αντιμετωπισθούν σύνθετα προβλήματα και να γραφούν τα αντίστοιχα προγράμματα, είναι η ιεραρχική προσέγγιση, η ανάπτυξη του προγράμματος ***από πάνω προς τα κάτω.***

- Κάθε πρόβλημα διαιρείται σε μικρότερα επιμέρους προβλήματα και κάθε ένα από αυτά διαιρείται σε ακόμα απλούστερα.
- Στο τέλος τα επιμέρους υποπροβλήματα είναι αρκετά απλά, ώστε οι αντίστοιχοι αλγόριθμοι και τα αντίστοιχα τμήματα προγράμματος να μπορούν να σχεδιασθούν και να γραφούν εύκολα.
- Ο τελικός αλγόριθμος του προβλήματος ανάγεται σε πολλούς απλούστερους επιμέρους αλγορίθμους, και το τελικό πρόγραμμα σε πολλά απλούστερα υποπρογράμματα.
- Η τεχνική του τμηματικού προγραμματισμού είναι ένα από τα βασικότερα χαρακτηριστικά του δομημένου προγραμματισμού.
- **Τμηματικός προγραμματισμός ονομάζεται** η τεχνική σχεδίασης και ανάπτυξης προγραμμάτων, ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.
- Όταν ένα τμήμα προγράμματος επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα, τότε αναφερόμαστε σε **υποπρόγραμμα.**

10.2 Χαρακτηριστικά των υποπρογραμμάτων

- **Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο.**
  Ένα υποπρόγραμμα ενεργοποιείται με την είσοδο σε αυτό, και απενεργοποιείται από την έξοδο που γίνεται από το τέλος.
- **Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα.**
  Μπορεί να σχεδιασθεί, να αναπτυχθεί και να συντηρηθεί αυτόνομα χωρίς να επηρεαστούν άλλα υποπρογράμματα.
- **Κάθε υποπρόγραμμα δεν πρέπει να είναι πολύ μεγάλο.**
  Πρέπει να είναι τόσο, ώστε να είναι εύκολα κατανοητό για να μπορεί να ελέγχεται. Πρέπει να εκτελεί μόνο μία λειτουργία, αν όχι, πρέπει να διασπαστεί σε μικρότερα υποπρογράμματα.

10.3 Πλεονεκτήματα τμηματικού προγραμματισμού.

- **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος**.
  Επιτρέπει την εξέταση και επίλυση απλών προβλημάτων, και όχι στην αντιμετώπιση του συνολικού προβλήματος, με τη σταδιακή επίλυση των υποπροβλημάτων και την δημιουργία των αντίστοιχων υποπρογραμμάτων τελικά επιλύεται το συνολικό πρόβλημα.
- **Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.**
  Επιτρέπει τη γρήγορη διόρθωση ενός συγκεκριμένου τμήματος, χωρίς αυτές οι αλλαγές να επηρεάσουν το υπόλοιπο πρόγραμμα.
  Διευκολύνει οποιονδήποτε θέλει να διαβάσει και να κατανοήσει πως λειτουργεί. Βασικό χαρακτηριστικό για μεγάλα προγράμματα που χρειάζονται συντήρηση.
- **Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος.**
  Πολύ συχνά χρειάζεται η ίδια λειτουργία σε διαφορετικά σημεία του προγράμματος. Από τη στιγμή που αυτό έχει γραφεί, καλείται από πολλά σημεία του προγράμματος. Έτσι έχουμε:
  - ✓ *Μείωση μεγέθους του προγράμματος.*
  - ✓ *Μείωση στο χρόνο συγγραφής.*
  - ✓ *Μείωση πιθανοτήτων λάθους.*

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

Ένα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί και σε άλλα προγράμματα. Η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων, που παρέχει η γλώσσα προγραμματισμού (π.χ. υπολογισμό ημίτονου συνημίτονου).

10.4 Παράμετροι

κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα το οποίο ονομάζεται κύριο πρόγραμμα. Το υποπρόγραμμα είναι αυτόνομο και ανεξάρτητο τμήμα, αλλά συχνά χρειάζεται να επικοινωνεί με το υπόλοιπο πρόγραμμα.

- **Παράμετρος είναι** μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από το ένα τμήμα προγράμματος σε ένα άλλο.

10.5 Διαδικασίες και συναρτήσεις

υπάρχουν δύο τύποι υποπρογραμμάτων:

- **Συναρτήσεις** – Συνάρτηση είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομα της συνάρτησης (όπως οι μαθηματικές συναρτήσεις).
- **Διαδικασίες** – Διαδικασία είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος.
- *Είναι προφανές ότι κάθε συνάρτηση μπορεί να γραφεί ως διαδικασία, ενώ το αντίστροφο δεν ισχύει.*

Παράδειγμα 1ο

Δίνεται ένας ακέραιος αριθμός x και θέλουμε να υπολογίσουμε και να εμφανίσουμε το τετράγωνό του. Ασφαλώς ο αλγόριθμος μπορεί να γραφεί δίχως χρήση υποπρογραμμάτων. Μπορούμε όμως να 'σπάσουμε' τον αλγόριθμο σε τρία επιμέρους μέρη. Την εισαγωγή των δεδομένων (διαδικασία), τον υπολογισμό του τετραγώνου (συνάρτηση) και την εμφάνιση των αποτελεσμάτων (διαδικασία).

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος_δεδομένων(Αριθμός)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Αριθμός
ΑΡΧΗ
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΓΡΑΨΕ 'Δώστε ένα θετικό ακέραιο: '
        ΔΙΑΒΑΣΕ Αριθμός
    ΜΕΧΡΙΣ_ΟΤΟΥ Αριθμός > 0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Οι λέξεις που είναι έντονες και με κεφαλαία είναι **δεσμευμένες** λέξεις. Η παραπάνω διαδικασία επιστρέφει έναν ακέραιο αριθμό (τη μεταβλητή Αριθμός) που αποτελεί παράμετρό της.

```
ΣΥΝΑΡΤΗΣΗ Τετράγωνο(Αριθμός): ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: Αριθμός
ΑΡΧΗ
    Τετράγωνο ← Αριθμός ^ 2
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Οι λέξεις που είναι έντονες και με κεφαλαία είναι **δεσμευμένες** λέξεις. Η παραπάνω συνάρτηση δέχεται ως παράμετρο έναν ακέραιο αριθμό (τη μεταβλητή Αριθμός) και επιστρέφει το τετράγωνό του.

```
ΔΙΑΔΙΚΑΣΙΑ Εμφάνιση_αποτελεσμάτων(Αριθμός)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Αριθμός
ΑΡΧΗ
    ΓΡΑΨΕ 'Το αποτέλεσμα είναι: ', Αριθμός
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Οι λέξεις που είναι έντονες και με κεφαλαία είναι **δεσμευμένες** λέξεις. Η παραπάνω διαδικασία δέχεται ως παράμετρο έναν ακέραιο αριθμό (τη μεταβλητή Αριθμός) και απλά τον εμφανίζει.

Το κυρίως πρόγραμμα έχει ως εξής:

ΠΡΟΓΡΑΜΜΑ Υπολογισμός_τετραγώνου
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: x, y
ΑΡΧΗ
ΚΑΛΕΣΕ Είσοδος_δεδομένων(x)
y ← Τετράγωνο(x)
ΚΑΛΕΣΕ Εμφάνιση_αποτελεσμάτων(y)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Οι λέξεις που είναι έντονες και με κεφαλαία είναι **δεσμευμένες** λέξεις. Οι διαδικασίες καλούνται με χρήση της εντολής **ΚΑΛΕΣΕ**, ενώ οι συναρτήσεις απλά με το όνομά τους που ισούται με την τιμή που επιστρέφουν.

10.5.1 Ορισμός και κλήση συναρτήσεων

ΣΥΝΑΡΤΗΣΗ Όνομα (λίστα παραμέτρων): τύπος συνάρτησης
Τμήμα δηλώσεων
ΑΡΧΗ
Όνομα <-- έκφραση
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗ

10.5.2 Ορισμός και κλήση διαδικασιών

ΔΙΑΔΙΚΑΣΙΑ Όνομα_διαδικασίας (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
Εντολή/ές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Κλήση διαδικασίας
.....
ΚΑΛΕΣΕ όνομα_διαδικασίας (λίστα παραμέτρων)
.....

10.5.3 Πραγματικές και τυπικές παράμετροι

- ***Η λίστα των ΤΥΠΙΚΩΝ παραμέτρων καθορίζει τις παραμέτρους στη ΔΗΛΩΣΗ του υποπρογράμματος.***
- ***Η λίστα των ΠΡΑΓΜΑΤΙΚΩΝ παραμέτρων καθορίζει τις παραμέτρους στη ΚΛΗΣΗ του υποπρογράμματος.***

Οι λίστες των παραμέτρων πρέπει να ακολουθούν τους εξής κανόνες:

- **Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ο ίδιος.**
- **Κάθε πραγματική αντιστοιχεί στην τυπική που βρίσκεται στην αντίστοιχη θέση.**
- **Η τυπική παράμετρος και η αντίστοιχη της τυπική να είναι του ίδιου τύπου.**

ΑΣΚΗΣΕΙΣ

1. α. Χρησιμοποιώντας δομή ακολουθίας γράψτε έναν αλγόριθμο με εντολές ψευδογλώσσας ο οποίος θα δέχεται ως είσοδο έναν αριθμό a, θα υπολογίζει τον αντίθετό του και θα τον εμφανίζει ως πληροφορία εξόδου.

β. Χρησιμοποιώντας δομή ακολουθίας γράψτε έναν αλγόριθμο με εντολές ψευδογλώσσας ο οποίος θα δέχεται ως είσοδο ένα ποσό σε δραχμές, θα υπολογίζει το αντίστοιχό του σε ευρώ και θα τον εμφανίζει.

γ. Σε μία τάξη υπάρχουν 3 τμήματα. Να γράψετε έναν αλγόριθμο τόσο σε ψευδογλώσσα όσο και σε διάγραμμα ροής που αρχικά θα διαβάζει τον αριθμό των αγοριών και τον αριθμό των κοριτσιών για κάθε τμήμα. Κατόπιν θα υπολογίζει και θα εμφανίζει το σύνολο των κοριτσιών της τάξης, το σύνολο των αγοριών της τάξης, το σύνολο των μαθητών της τάξης και το ποσοστό επί τοις εκατό αγοριών και κοριτσιών της τάξης

δ. Χρησιμοποιώντας δομή ακολουθίας γράψτε έναν αλγόριθμο με εντολές ψευδογλώσσας ο οποίος θα διαβάζει έναν αριθμό a, και αφού υπολογίσει τον διπλάσιό του, θα τον εμφανίζει.

2. α. Δίνεται η παρακάτω εντολή Αν.

Αν (βάρος < 60) **και** (ύψος > 1.80) **και** (φύλο = 'ΓΥΝΑΙΚΑ') **τότε**

Εμφάνισε 'ψηλή αδύνατη'
Τέλος_αν

Ξαναγράψτε την εντολή χρησιμοποιώντας εμφωλευμένο Αν ώστε να έχουμε το ίδιο αποτέλεσμα.

β. Δίνεται η παρακάτω εντολή Αν.

Αν **(βάρος < 60) τότε**

Αν (ύψος > 1.80) τότε

Αν (φύλο = 'ΓΥΝΑΙΚΑ') τότε

εμφάνισε 'ψηλή αδύνατη'

αλλιώς_αν (φύλο = 'ΝΔΡΑΣ')

τέλος_αν

τέλος_αν

Τι λάθη υπάρχουν

3. α. Να γίνει αλγόριθμος σε ψευδοκώδικα (ψευδογλώσσα) και διάγραμμα ροής, που θα διαβάζει τον μέσο όρο βαθμολογίας ενός μαθητή, και θα εμφανίζει το αποτέλεσμα φοίτησης ανάλογα με τον μέσο όρο βαθμολογίας του, όπως παρακάτω:

Για μέσο όρο μεγαλύτερο ή ίσο του 18,5, το αποτέλεσμα φοίτησης θα είναι 'Άριστος'.
Για μέσο όρο μεγαλύτερο ή ίσο του 17 και μικρότερο του 18,5, το αποτέλεσμα φοίτησης θα είναι 'Πολύ καλός'.
Για μέσο όρο μεγαλύτερο ή ίσο του 14,5 και μικρότερο του 17, το αποτέλεσμα φοίτησης θα είναι 'Μέτριος'.
Για μέσο όρο μεγαλύτερο ή ίσο του 10 και μικρότερο του 14,5 το αποτέλεσμα φοίτησης θα είναι 'Μέτριος'.
Για μέσο όρο μικρότερο του 10, το αποτέλεσμα φοίτησης θα είναι 'Απορρίπτεται'.

β. Να γίνει αλγόριθμος σε ψευδοκώδικα και διάγραμμα ροής, που θα διαβάζει τον αριθμό των αγοριών και κοριτσιών ενός γονέα, και θα εμφανίζει αν είναι πολύτεκνος Επίσης, σε περίπτωση που ο πολύτεκνος γονέας έχει και πάνω από 3 αγόρια να εμφανίζει και ένα μήνυμα ότι 'Το πρώτο αγόρι θα υπηρετήσει φαντάρος μόνο ένα μήνα'.

4. Ξαναγράψτε τον αλγόριθμο του παραδείγματος 7 (σχολικό βιβλίο, σελ. 40) χρησιμοποιώντας την εντολή Για...από...μέχρι αντί της εντολής Όσο...επανέλαβε ώστε να έχουμε το ίδιο αποτέλεσμα.

5. Που διαφέρει ο αλγόριθμος του παραδείγματος 8 (σχολικό βιβλίο, σελ. 41) από τον αλγόριθμο του παραδείγματος 9 (σελ. 42)

6. Τροποποιήστε τον αλγόριθμο του παραδείγματος 11 (σχολικό βιβλίο, σελ. 44) ώστε να βρεθεί και να εκτυπωθεί το άθροισμα των **περιττών** αριθμών από 1 μέχρι 100.

7. α. Τι θα εμφανιστεί μετά την εκτέλεση της παρακάτω εντολής

Για k **από** 100 **μέχρι** 1 **με_βήμα** −1

Εμφάνισε k
Τέλος_επανάληψης

Ξαναγράψτε το παραπάνω τμήμα αλγορίθμου χρησιμοποιώντας την εντολή Όσο…επανέλαβε ώστε να έ-

χουμε το ίδιο αποτέλεσμα.

β. Να γίνει ένας αλγόριθμος σε ψευδοκώδικα (ψευδογλώσσα) που θα διαβάζει τις ηλικίες μιας ομάδας 50 ατόμων, και θα εμφανίζει τον μέσο όρο των ηλικιών.

γ. Να βρεθεί και να εμφανιστεί το άθροισμα των μαθητών και των μαθητριών 50 σχολείων, επίσης το άθροισμα των μαθητών, το άθροισμα των μαθητριών, και τα ποσοστά μαθητών και μαθητριών. Δεδομένα είναι ο αριθμός των αγοριών για κάθε σχολείο, και ο αριθμός κοριτσιών για κάθε σχολείο.

δ. Δίνεται η παρακάτω εντολή Όσο.

```
διάβασε IQ
    Όσο (IQ > 120) τότε
        Αν (φύλο = "ΓΥΝΑΙΚΑ") τότε
            εμφάνισε "είσαι πανέξυπνη!"
        αλλιώς_αν (φύλο = "ΑΝΔΡΑΣ") τότε
            εμφάνισε "είσαι πανέξυπνος!"
        τέλος_αν
    τέλος_αν
```

Τι λάθη υπάρχουν

8. Στα παρακάτω τμήματα αλγορίθμου γράψτε τι θα εμφανιστεί κάθε φορά:

α.
```
x ← 3
y ← 4
εμφάνισε x * y
```

β.
```
x ← 2
y ← 5
z ← 2
Αν y < z τότε
    εμφάνισε 2 * x + y – z
αλλιώς
    εμφάνισε 3 * x - y + z
τέλος_αν
```

γ.
```
x ← 10
y ← 2
Όσο x >= y επανέλαβε
    εμφάνισε x, y
    y ← y + 3
Τέλος_επανάληψης
```

δ. x ← 6

y ← 2

Αρχή_επανάληψης

Αν x > 2 * y **τότε**

εμφάνισε x + y

αλλιώς

εμφάνισε y - x

τέλος_αν

y ← y + 2

Μέχρις_ότου x >= y

Πόσες φορές θα έχω εμφάνιση (δηλ. πόσες επαναλήψεις)

ε. **Για** k **από** 1 **μέχρι** 7

table[k] ← 0

Τέλος_επανάληψης

Για k **από** 1 **μέχρι** 7 **με_βήμα** 3

table[k] ← 2 * k - 1

εμφάνισε table[k] + 1

Τέλος_επανάληψης

εμφάνισε k, table[1], table[2], table[3], table[4]

9. Σε μία αίθουσα βρίσκονται 15 καπνιστές. Θεωρούμε ότι κάθε τσιγάρο περιέχει 0,15 mg (μικρογραμμάρια) νικοτίνης και 1,2 mg (μικρογραμμάρια) πίσσας. Θεωρούμε επίσης ότι η μέση διάρκεια ζωής για κάθε καπνιστή είναι 70 χρόνια. Να υλοποιήσετε έναν αλγόριθμο ο οποίος:

α. Θα διαβάζει την ηλικία όλων των καπνιστών της αίθουσας.

β. Κατόπιν θα διαβάζει τον αριθμό των τσιγάρων πού καπνίζει ο κάθε καπνιστής της αίθουσας (μόνο για τους καπνιστές που είναι νεώτεροι από 70 ετών-για τους άλλους δεν θα διαβάζεται ο αριθμός των τσιγάρων που καπνίζουν).

γ. Στη συνέχεια θα υπολογίζει το **<u>συνολικό</u>** βάρος (σε γραμμάρια) της νικοτίνης και το συνολικό βάρος (σε γραμμάρια) της πίσσας που θα καταναλωθούν από όλους μαζί τους καπνιστές της αίθουσας που είναι νεώτεροι από 70 ετών από τώρα μέχρι το τέλος της ζωής τους.

δ. Τελικά θα εμφανίζει τις δύο παραπάνω συνολικές τιμές.

Η υλοποίηση να γίνει τόσο σε μορφή ψευδοκώδικα (ψευδογλώσσας) όσο και σε μορφή διαγράμματος ροής.

10. Γράψτε έναν αλγόριθμο σε ψευδογλώσσα ο οποίος αρχικά θα διαβάζει τα ονόματα και τις ηλικίες 20 μαθητών και θα τα καταχωρεί σε δύο μονοδιάστατους πίνακες Α και Β. Κατόπιν αφού θα υπολογίζει το μέσο όσο ηλικίας, θα τον εμφανίζει. Τελικά θα εμφανίζει τα ονόματα των μαθητών με ηλικία μεγαλύτερη από το μέσο όρο.

11. Σε μία αίθουσα βρίσκονται μαζεμένοι πάρα πολλοί μαθητές Γ' Λυκείου (δεν γνωρίζουμε πόσοι). Θέλουμε να διαχωρίσουμε τους μαθητές ανάλογα με την κατεύθυνση (θετική, θεωρητική, τεχνολογική) που έχουν επιλέξει και ορισμένα άλλα κριτήρια. Να υλοποιήσετε **<u>έναν</u>** αλγόριθμο σε μορφή ψευδογλώσσας ο οποίος:

α. Θα διαβάζει το όνομα του κάθε μαθητή. Τα ονόματα θα τοποθετούνται σε έναν πίνακα **ΜΑΘ**. Η διαδικασία ανάγνωσης ονομάτων θα σταματά αμέσως μόλις δοθεί το ειδικό όνομα '**ΤΕΛΕΥΤΑΙΟΣ'**. Εξυπακούεται ότι το ειδικό όνομα αυτό εξυπηρετεί μόνο το σκοπό αυτό και δεν πρέπει να εισαχθεί στον πίνακα ΜΑΘ.

β. Μετά την παραπάνω διαδικασία, θα εμφανίζεται κάθε στοιχείο (όνομα) του πίνακα ΜΑΘ και θα διαβάζεται η κατεύθυνση που έχει επιλέξει ο αντίστοιχος μαθητής. Ανάλογα με ποια από τις τρεις κατευθύνσεις έχει επιλεγεί, τα ονόματα των μαθητών θα εισάγονται σε τρεις πίνακες αντίστοιχα (**ΘΕΤ**, **ΘΕΩΡ** και **ΤΕΧ**).

γ. Στη συνέχεια θα εμφανίζεται κάθε στοιχείο (όνομα) του πίνακα ΤΕΧ και θα διαβάζεται η ηλικία και το

φύλο του αντίστοιχου μαθητή. Σε περίπτωση που η ηλικία είναι μεγαλύτερη ή ίση από **18** και το φύλο είναι **'ΆΡΡΕΝ'** θα εμφανίζεται το μήνυμα 'Θέλεις να καταταγείς εθελοντής στο στρατό '. Αφού διαβαστεί η απάντηση ('ΝΑΙ' ή 'ΟΧΙ') τα ονόματα αυτών που απαντούν '**ΝΑΙ'** θα εισάγονται σε έναν πίνακα **ΣΤΡ**.

δ. Τελικά θα εμφανίζεται ο συνολικός αριθμός όλων των μαθητών (αριθμητικά), ο συνολικός αριθμός των μαθητών θετικής κατεύθυνσης, ο συνολικός αριθμός των μαθητών θεωρητικής κατεύθυνσης, ο συνολικός αριθμός των μαθητών τεχνολογικής κατεύθυνσης, και ο συνολικός αριθμός των εθελοντών στρατιωτών.

Να δοθεί ιδιαίτερη προσοχή στο συλλογισμό, στη σωστή χρησιμοποίηση των μεταβλητών και στην κατάλληλη επιλογή των εντολών.

12. Μετατρέψτε τους παρακάτω αλγορίθμους σε δομημένους ώστε να δίνουν το ίδιο αποτέλεσμα.

α.
```
1:  Διάβασε x
    Αν x > 0 τότε
        εμφάνισε x
        goto 1
    τέλος_αν
```

β.
```
    sum ← 0
    i ← 2
1:  Αν i <= 100 τότε
        sum ← sum + i
        i ← i + 2
        goto 1
    αλλιώς
        goto 2
    τέλος_αν
2:  εμφάνισε sum
```