Γ΄ ΛΥΚΕΙΟΥ

Συνοπτική Παρουσίαση της Εξεταζόμενης ύλης

Περιεχόμενα

Κεφάλαιο 1. Ανάλυση Προβλήματος

1.1 Η έννοια του προβλήματος

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

1.2 Κατανόηση προβλήματος

1. Η **κατανόηση** ενός προβλήματος εξαρτάται σε μεγάλο βαθμό από τη διατύπωσή του. Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός είτε ο γραπτός.
2. Με τον όρο **δεδομένο** δηλώνεται οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις του. Με τον όρο **πληροφορία** αναφέρεται οποιοδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων. Ο όρος **επεξεργασία δεδομένων** δηλώνει εκείνη τη διαδικασία κατά την οποία ένας "μηχανισμός" δέχεται δεδομένα, τα επεξεργάζεται σύμφωνα με έναν προκαθορισμένο τρόπο και αποδίδει πληροφορίες.

1.3 Δομή προβλήματος

1. 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.
2. Η **ανάλυση** ενός προβλήματος μπορεί να πραγματοποιηθεί είτε **φραστικά** είτε **διαγραμματικά**.
3. Η **διαγραμματική** αναπαράσταση προσφέρει μια απτή απεικόνιση της δομής του προβλήματος. Η δημιουργία του σχετικού διαγράμματος βοηθάει τόσο στην καλύτερη κατανόηση του ίδιου του προβλήματος, όσο και στη σχεδίαση της λύσης του.
4. Η καταγραφή της δομής ενός προβλήματος σημαίνει αυτόματα ότι έχει αρχίσει η διαδικασία ανάλυσης του προβλήματος σε άλλα απλούστερα.

1.4 Καθορισμός απαιτήσεων

Συμπερασματικά από όλα τα παραπάνω διαφαίνεται πως τα στάδια αντιμετώπισης ενός προβλήματος είναι τρία:

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

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

Η ***ανάλυση-αφαίρεση*** αποτελεί το δεύτερο βήμα στην διαδικασία επίλυσης ενός προβλήματος και προηγείται της επίλυσης και έπεται της κατανόησης ενός προβλήματος. Στόχος της ανάλυσης, είναι η διάσπαση του προβλήματος σε άλλα απλούστερα προβλήματα για να είναι εύκολη η αντιμετώπισή τους.

Για τη σωστή επίλυση ενός προβλήματος είναι σημαντικός ο επακριβής προσδιορισμός των **δεδομένων** που παρέχει το πρόβλημα και η λεπτομερειακή καταγραφή των **ζητούμενων** που αναμένονται σαν αποτελέσματα της επίλυσης του προβλήματος. Για να βρει κάποιος τα ζητούμενα χρειάζεται να επεξεργαστεί τα δεδομένα.

Κεφαλαιο 2: Βασικές έννοιες αλγορίθμων

2.1 Τι είναι αλγόριθμος

{Ορισμός} Είναι μία **πεπερασμένη** σειρά ενεργειών, αυστηρά **καθορισμένων** και **εκτελέσιμων** σε πεπερασμένο χρόνο, που **στοχεύουν** στην επίλυση ενός προβλήματος.
Η σειρά – αλληλουχία των ενεργειών δεν είναι μοναδική. Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά με έννοιες της πληροφορικής.

Απαραίτητα κριτήρια αλγορίθμων

a. **Είσοδος**: **Καμία, μία ή περισσότερες** τιμές δεδομένων πρέπει να δίνονται ως είσοδος. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται όταν: ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες **πρωτογενείς** τιμές με την βοήθεια **συναρτήσεων παραγωγής τυχαίων αριθμών** ή με την βοήθεια άλλων **απλών εντολών**.
b. **Έξοδος**: Ο αλγόριθμος πρέπει να δημιουργεί **τουλάχιστον** ένα αποτέλεσμα προς τον **χρήστη** ή προς ένα άλλο **αλγόριθμο**.
c. **Καθοριστικότητα**: Κάθε εντολή πρέπει να καθορίζεται χωρίς **αμφιβολία** για τον τρόπο εκτέλεσής της. Πχ μία εντολή ζ <-- χ / ψ πρέπει να λαμβάνει υπ' όψιν της το γεγονός ότι μπορεί το ψ να είναι μηδέν.
d. **Περατότητα**: Ο αλγόριθμος πρέπει να τελειώνει μετά από **πεπερασμένα** βήματα. Αν δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν είναι αλγόριθμος αλλά **υπολογιστική** διαδικασία.
e. **Αποτελεσματικότητα**: Κάθε εντολή ενός αλγορίθμου δεν αρκεί να είναι **ορισμένη** αλλά πρέπει να είναι **απλή** και **εκτελέσιμη**. Π.χ.παραβίαση αυτού του κριτηρίου είναι η εντολή «Βρες το μεγαλύτερο από αυτούς τους 100 αριθμούς». Η γλώσσες προγραμματισμού (του επιπέδου που μιλάμε) δεν έχουν τέτοια εντολή. Χρειάζεται σύνολο εντολών για να βρούμε το μεγαλύτερο από 100 αριθμούς. Τέτοιου είδους παραβιάσεις οδηγούν σε μη εκτελέσιμο αλγόριθμο.

2.2 Σπουδαιότητα αλγορίθμων

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

- **Υλικού** (hardware). Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού, δηλαδή από τον τρόπο που είναι δομημένα σε μία ενιαία αρχιτεκτονική τα διάφορα συστατικά του υπολογιστή (δηλαδή ανάλογα με το αν ο υπολογιστής έχει κρυφή μνήμη και πόση, ανάλογα με την ταχύτητα της κύριας και δευτερεύουσας μνήμης κ.ο.κ.).
- **Γλωσσών Προγραμματισμού** (programming languages). Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλαδή, χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου. Γενικά μία γλώσσα που είναι

χαμηλότερου επιπέδου (όπως η assembly ή η γλώσσα C) είναι ταχύτερη από μία άλλη γλώσσα που είναι υψηλοτέρου επιπέδου (όπως η Basic ή Pascal). Ακόμη, σημειώνεται ότι διαφορές συναντώνται μεταξύ των γλωσσών σε σχέση με το πότε εμφανίσθηκαν. Για παράδειγμα, παλαιότερα μερικές γλώσσες προγραμματισμού δεν υποστήριζαν την αναδρομή (έννοια που θα εξετάσουμε σε βάθος αργότερα).

- **Θεωρητική** (theoretical). Το ερώτημα που συχνά τίθεται είναι αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την δύσκολο να σχολιασθεί στα πλαίσια του βιβλίου αυτού, επειδή απαιτεί μεγάλη θεωρητική κατάρτιση. Ωστόσο η προσέγγιση αυτή είναι ιδιαίτερα σημαντική, γιατί προσδιορίζει τα όρια της λύσης που θα βρεθεί σε σχέση με ένα συγκεκριμένο πρόβλημα.
- **Αναλυτική** (analytical). Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για λειτουργίες CPU και για λειτουργίες εισόδου/εξόδου κ.λπ.

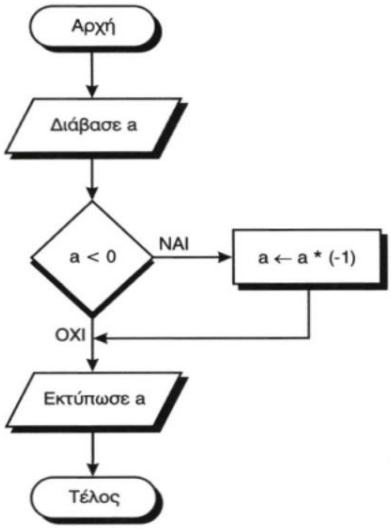
2.3 Περιγραφή και αναπαράσταση αλγορίθμων

a. **Ελεύθερο κείμενο** (Free text): Αποτελεί τον πιο **ανεπεξέργαστο** και **αδόμητο** τρόπο παρουσίασης. Υπάρχει κίνδυνος να οδηγήσει σε μη εκτελέσιμη παρουσίαση, παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων το κριτήριο της **αποτελεσματικότητας**.

b. **Διαγραμματικές τεχνικές** (diagramming techniques): Αποτελούν ένα **γραφικό** τρόπο αναπαράστασης. Μια τεχνική (από τις πιο παλιές και γνωστές) είναι το διάγραμμα **ροής** (Flow Chart). Η χρήση διαγραμματικών τεχνικών δεν είναι η καλύτερη λύση, γι' αυτό εμφανίζονται όλο και σπανιότερα στην βιβλιογραφία και στην πράξη.

Διάγραμμα ροής

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



Τα κυριότερα σχήματα είναι τα εξής:

1. **Έλλειψη**: Δηλώνει την αρχή και το τέλος του αλγορίθμου.
2. **Ρόμβος**: Δηλώνει τον έλεγχο μίας συνθήκης με δύο εξόδους ανάλογα με την τιμή της. {αληθής - ψευδής}
3. **Ορθογώνιο**: Δηλώνει την εκτέλεση μίας ή περισσοτέρων πράξεων.
4. **Πλάγιο παραλληλόγραμμο**: Η είσοδος δεδομένων και η έξοδος αποτελεσμάτων του αλγορίθμου.

c. **Φυσική γλώσσα κατά βήματα** (natural language): Μοιάζει με το **ελεύθερο κείμενο** απλά είναι κατά βήματα. Κίνδυνος να παραβιαστεί το κριτήριο της **καθοριστικότητας**.

d. **Κωδικοποίηση** (coding): Δηλαδή με ένα πρόγραμμα γραμμένο είτε με μία **ψευδογλώσσα** είτε σε κάποιο **προγραμματιστικό περιβάλλον** που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Κεφάλαιο 4. Τεχνικές Σχεδίασης Αλγορίθμων

4.1 Ανάλυση αλγορίθμων

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

- την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
- την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
- την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του

και στη συνέχεια:

- ❖ την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
- ❖ την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

Έτσι, κατά την ανάλυση ενός προβλήματος θα πρέπει να δοθεί απάντηση σε καθεμία από τις επόμενες ερωτήσεις:

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

Κεφάλαιο 6: Εισαγωγή στον προγραμματισμό

6.1 Η έννοια του προγράμματος

Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει, όπως έχει ήδη αναφερθεί, τρία εξίσου σημαντικά στάδια.

- Τον ακριβή προσδιορισμό του προβλήματος.
- Την ανάπτυξη του αντίστοιχου αλγορίθμου.
- Τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή.

Τι μπορεί να κάνει ο υπολογιστής;

Το μόνο πράγμα που κάνει ο υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολουθίες αυτών των δύο ψηφίων, αλλά αυτές τις ενέργειες τις εκτελεί με ασύλληπτη ταχύτητα. Ο υπολογιστής μπορεί απλά να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.

6.4 Τεχνικές σχεδίασης προγραμμάτων

6.4.1 Ιεραρχικής σχεδίασης προγράμματος.

Η τεχνική αυτή (ή αλλιώς τεχνική **από επάνω προς τα κάτω** / top-down), χρησιμοποιεί την στρατηγική της συνεχούς **διαίρεσης** του προβλήματος σε **υπό-προβλήματα** τα οποία είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του **αρχικού** προβλήματος.

Χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές για την υποβοήθηση της σχεδίασης. Υλοποιείται με **τμηματικό** προγραμματισμό.

Τι περιλαμβάνει η τεχνική της ιεραρχικής σχεδίασης;

- Τον **καθορισμό** των **βασικών** λειτουργιών ενός προγράμματος σε **ανώτερο** επίπεδο,
- την **διάσπαση** των λειτουργιών αυτών σε όλο και **μικρότερες** λειτουργίες, μέχρι το **τελευταίο** επίπεδο όπου οι λειτουργίες είναι πολύ **απλές**.

6.4.2 Τμηματικός προγραμματισμός

Κάθε τμήμα του προγράμματος υλοποιείται ξεχωριστά από τα υπόλοιπα τμήματα.

Πως υλοποιείται η ιεραρχική σχεδίαση

Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον τμηματικό προγραμματισμό.

6.4.3 Δομημένος προγραμματισμός

Δεν είναι απλώς ένα είδος προγραμματισμού αλλά μια **μεθοδολογία σύνταξης** προγραμμάτων. **Προήλθε** από μια προσπάθεια περιορισμού της ανεξέλεγκτης **χρήσης** της εντολής **goto**. Όλες οι **σύγχρονες** γλώσσες προγραμματισμού υποστηρίζουν τον δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν την χρήση του goto **περιττή**. Ο δομημένος προγραμματισμός βοηθάει την **ανάλυση** ενός προγράμματος σε **τμήματα** έτσι **περιέχει** τόσο την **ιεραρχική** σχεδίαση όσο και τον **τμηματικό προγραμματισμό**.

Ποιες δομές **χρησιμοποιεί** ο δομημένος προγραμματισμός

Σύμφωνα με τον δομημένο προγραμματισμό, όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας **μόνο** τις τρεις παρακάτω λογικές δομές καθώς και συνδυασμών τους.

1. Δομή ακολουθίας.
2. Δομή επιλογής.
3. Δομή επανάληψης.

Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μία είσοδο και μόνο μία έξοδο
{Εδώ εννοεί ότι έχει μόνο **μία αρχή** και μόνο **ένα τέλος_προγράμματος** !!!}

Γιατί η εντολή GOTO κρίνεται ακατάλληλη

Επειδή η χρήση της εντολής **αλλάζει** την **ροή** ενός προγράμματος, τα προγράμματα γίνονται **δύσκολα** στην παρακολούθηση την κατανόηση και την συντήρηση.

Πλεονεκτήματα δομημένου προγραμματισμού σελ. 119

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

6.3 Φυσικές και τεχνητές γλώσσες

Από τι προσδιορίζεται μια γλώσσα

Οι φυσικές γλώσσες αλλά και οι γλώσσες προγραμματισμού προσδιορίζονται από **4** στοιχεία:

α. Το αλφάβητο.

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

β. Το λεξιλόγιο.

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

γ. Την γραμματική.

Η γραμματική αποτελείται από το τυπικό και το συντακτικό.

Τυπικό: είναι το σύνολο των κανόνων που καθορίζουν αν μία **λέξη** είναι αποδεκτή. Πχ Η λέξη «ΓΡΑΨΕ» είναι αποδεκτή αλλά η «ΓΡΑΨΕΣ» όχι.

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

{ΠΧ:}

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

δ. Την σημασιολογία.

Είναι το σύνολο των κανόνων που καθορίζει το νόημα των **λέξεων** άρα και το νόημα των **προτάσεων** που δημιουργούνται. {Πχ Απογειώθηκε το ντροπαλό κατσαβίδι. (δεν έχει νόημα..)}

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

Διαφορές φυσικών – τεχνητών γλωσσών

1. Οι φυσικές γλώσσες **εξελίσσονται** συνεχώς και δημιουργούνται νέες λέξεις, αλλάζουν οι κανόνες γραμματικής/ συντακτικού με την πάροδο του χρόνου. Αυτό γίνεται γιατί χρησιμοποιούνται για την επικοινωνία μεταξύ των **ανθρώπων**.
2. Οι γλώσσες προγραμματισμού χρησιμοποιούνται για την επικοινωνία **ανθρώπου Η/Υ** και χαρακτηρίζονται από **στασιμότητα**. Ωστόσο και αυτές εξελίσσονται από τους δημιουργούς τους για να διορθώσουν αδυναμίες τους ή να καλύψουν μεγαλύτερο εύρος εφαρμογών αλλά και να ακολουθήσουν τις νέες εξελίξεις.

Οι γλώσσες προγραμματισμού αλλάζουν σε:

επίπεδο **διαλέκτου**. Πχ Basic σε QuickBasic

σε επίπεδο **επέκτασης** Πχ Basic σε Visual Basic

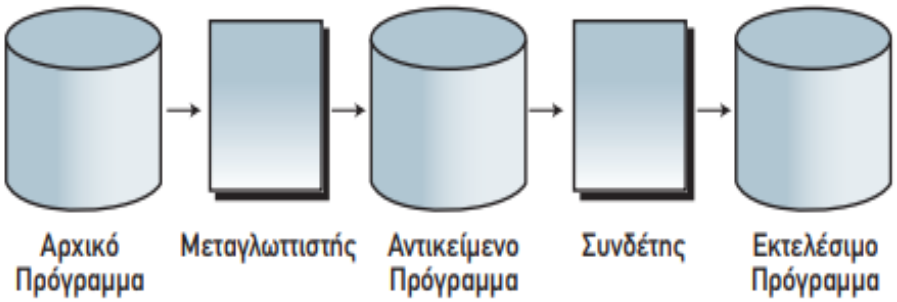
6.7 Προγραμματιστικά περιβάλλοντα

Τι είναι το προγραμματιστικό περιβάλλον:
Προγραμματιστικό περιβάλλον ονομάζουμε το σύνολο των προγραμμάτων και εργαλείων που απαιτούνται και βοηθούν τη συγγραφή την εκτέλεση και κύρια τη διόρθωση ενός προγράμματος. Ένα σύγχρονο προγραμματιστικό περιβάλλον περιλαμβάνει τουλάχιστον:

- **Συντάκτη**
  Για τη συγγραφή και διόρθωση του προγγράματος
- **Μεταγλωττιστή**
  Για τη μετάφραση σε γλώσσα μηχανής
- **Συνδέτη**
  Για τη σύνδεση με τις βιβλιοθήκες και τη δημιουργία του τελικού εκτελέσιμου προγράμματος

Ποια είναι τα βήματα δημιουργίας και μεταγλώττισης ενός προγράμματος:
Η διαδικασία δημιουργίας και μεταγλώττισης ενός προγράμματος είναι η εξής:

1. Ο προγραμματιστής χρησιμοποιεί έναν **συντάκτη** (editor) για την αρχική συγγραφή του προγράμματος, που συνήθως είναι μια ακολουθία εντολών σε κάποια γλώσσα υψηλού επιπέδου. Το πρόγραμμα που δημιουργεί ο προγραμματιστής ονομάζεται **πηγαίο** (source) πρόγραμμα.
2. Εφόσον το πηγαίο πρόγραμμα δεν περιέχει **συντακτικά** λάθη, μετατρέπεται από το μεταγλωτιστή στο ισοδύναμό του πρόγραμμα σε γλώσσα μηχανής που λέγεται **αντικείμενο** (object) πρόγραμμα. Εάν υπάρχουν συντακτικά λάθη, ο μεταγλωττιστής τα ανιχνεύει, ο προγραμματιστής τα διορθώνει, και το διορθωμένο πρόγραμμα υποβάλλεται ξανά για μεταγλώττιση.
3. Αμέσως μετά, ο **συνδέτης - φορτωτής** (linker - loader) ολοκληρώνει τη σύνδεση του αντικείμενου προγράμματος και παράγει το **εκτελέσιμο** (executable) πρόγραμμα. Αυτό γιατί, το αντικείμενο πρόγραμμα αν και σε γλώσσα μηχανής, συνήθως δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις **βιβλιοθήκες** (libraries) της γλώσσας.



Ποιες είναι οι κατηγορίες μεταφραστικών προγραμμάτων:
Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμάτων:

- **Μεταγλωττιστής**
  Ο μεταγλωττιστής είναι ένα πρόγραμμα το οποίο δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου (**πηγαίο πρόγραμμα**) και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής (**αντικείμενο πρόγραμμα**). Το τελευταίο, μπορεί να αποθηκευτεί και να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το πηγαίο πρόγραμμα.
- **Διερμηνευτής**
  Ο διερμηνευτής είναι ένα πρόγραμμα το οποίο σε αντίθεση με το μεταγλωττιστή, δέχεται στην είσοδο μία προς μία τις εντολές του πηγαίου πρόγραμματος, και για κάθε μία εκτελεί αμέσως μια ισοδύναμη ακολυθία εντολών μηχανής.

Ποια τα μειονεκτήματα - πλεονεκτήματα μεταφραστή και διερμηνευτή:

Μεταφραστής:
(+):

- Το εκτελέσιμο πρόγραμμα που παράγει ο μεταγλωττιστής μπορεί να αποθηκευτεί και να εκτελεστεί οποτεδήποτε χωρίς να χρειάζεται πλέον το πηγαίο πρόγραμμα.
- Το εκτελέσιμο πρόγραμμα που παράγει ο μεταγλωττιστής είναι ταχύτερο από τη διαδικασία εκτέλεσης με το διερμηνευτή.

(-):

- Κατά τη φάση της ανάπτυξης του προγράμματος, εάν παρουσιαστούν λάθη, όλες οι εντολές του πηγαίου προγράμματος, πρέπει να ξαναπεράσουν επιτυχώς από τη διαδικασία της μεταγλώττισης και της σύνδεσης. Η διαδικασία επαναλαμβάνεται, μέχρι να διορθωθούν όλα τα λάθη (συντακτικά και λογικά).

Διερμηνευτής:
(+):

- Η χρήση διερμηνευτή έχει το πλεονέκτημα της άμεσης εκτέλεσης κάθε εντολής και συνεπώς και της άμεσης διόρθωσής της.

(-):

- Ο διερμηνευτής δεν παράγει εκτελέσιμο πρόγραμμα, ανεξάρτητο από το πηγαίο πρόγραμμα, όπως ο μεταγλωττιστής. Συνεπώς, για κάθε εκτέλεση του προγράμματος επαναλαμβάνεται η διαδικασία της μετάφρασης και της εκτέλεσης όλων των εντολών του πηγαίου προγράμματος μία προς μία.
- Η διαδικασία εκτέλεσης του προγράμματος με διερμηνευτή, καθίσταται πιο αργή από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής.

Τι συμβαίνει στα σύγχρονα προγραμματιστικά περιβάλλοντα:
Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με μεικτές υλοποιήσεις, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

Τι είναι ο συντάκτης:
Ο συντάκτης (editor) είναι ένα ειδικό πρόγραμμα που χρησιμοποιείται για την αρχική σύνταξη των προγραμμάτων και τη διόρθωσή τους. Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου, με δυνατότητες όμως που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων.

Πηγαίο ή αρχικό πρόγραμμα (source)

- Είναι το **αρχικό** πρόγραμμα που γράφει ο προγραμματιστής με την βοήθεια ενός προγράμματος που ονομάζεται **συντάκτης.**
- Δεν είναι **κατανοητό** από τον υπολογιστή. (εκτός αν είναι γραμμένο απ' ευθείας σε γλώσσα μηχανής)

Λάθη στο πηγαίο πρόγραμμα

- **Λογικά** λάθη.
  - Οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου.
  - Εντοπίζονται παρά μόνο κατά την εκτέλεση του προγράμματος.
  - Είναι τα πλέον σοβαρά και δύσκολα στην διόρθωση.
- **Συντακτικά** λάθη.
  - Οφείλονται σε αναγραμματισμούς γραμμάτων εντολών, παράληψη δήλωσης δεδομένων κλπ.
  - Εντοπίζονται από τον μεταγλωττιστή ή τον διερμηνευτή.
  - Πρέπει να διορθωθούν ώστε να δημιουργηθεί το εκτελέσιμο πρόγραμμα.
    {πχ Γραεψ αντί για Γράψε}

Τι είναι ο συντάκτης

Είναι ένας μικρός επεξεργαστής κειμένου που χρησιμοποιείται για την **συγγραφή** του πηγαίου προγράμματος, την **διόρθωση** των λαθών του και έχει δυνατότητες που διευκολύνουν την **γρήγορη γραφή** των εντολών.

Μετατροπή προγραμμάτων σε γλώσσα μηχανής

Γίνεται με την χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες μεταφραστικών προγραμμάτων: Οι μεταγλωττιστές και οι διερμηνευτές.

Μεταγλωττιστής

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

Συνδέτης φορτωτής

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

Διερμηνευτής

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

Ομοιότητες διερμηνευτή-μεταγλωττιστή

{ η παράγραφος αυτή δεν αναφέρεται ξεκάθαρα στο σχολικό εγχειρίδιο} Και οι δύο **μεταφράζουν** το πηγαίο πρόγραμμα (υψηλού επιπέδου) σε γλώσσα μηχανής. Και οι δύο **ανιχνεύουν** τα **συντακτικά** λάθη.

Διαφορές διερμηνευτή-μεταγλωττιστή

- Ο μεταγλωττιστής μεταγλωττίζει **όλο** το πρόγραμμα και με την βοήθεια του συνδέτη – φορτωτή παράγεται το εκτελέσιμο.
- Ο διερμηνευτής εκτελεί **μία μία** τις εντολές και δεν χρειάζεται συνδέτη-φορτωτή

Διαφορές ως απόρροια των παραπάνω είναι:

- Ο διερμηνευτής αφού εκτελεί τις εντολές μία μία έχει το πλεονέκτημα της **άμεσης** διόρθωσης των λαθών. Για τον λόγο αυτό χρησιμοποιείται συνήθως κατά την **συγγραφή-διόρθωση** ενός προγράμματος.
- Η εκτέλεση ενός προγράμματος με τον διερμηνευτή είναι πιο **αργή**, γιατί για να εκτελεστεί το πρόγραμμα, πρέπει κάθε φορά να **ξαναγίνεται** η διερμηνεία από την αρχή, ενώ ο μεταγλωττιστής παράγει **μια φορά** το αντικείμενο πρόγραμμα και δεν χρειάζεται **ξανά** μεταγλώττιση αφού είναι σχεδόν **εκτελέσιμο** (θυμήσου τον συνδέτη)
- Για να εκτελεστεί ένα πρόγραμμα με τον διερμηνευτή είναι απαραίτητη η **παρουσία** του πηγαίου προγράμματος ενώ με τον μεταγλωττιστή μόνο την **πρώτη** φορά.

(Σύγχρονα) προγραμματιστικά περιβάλλοντα

Για την δημιουργία - εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον 3 προγράμματα:

- Συντάκτης
- Μεταγλωττιστής
- Συνδέτης

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν τα προγράμματα αυτά με **ενιαίο** τρόπο. Το κάθε προγραμματιστικό περιβάλλον έχει διαφορετικά **εργαλεία** και ιδιότητες (ανάλογα με την γλώσσα προγραμματισμού). Για παράδειγμα ένα περιβάλλον **οπτικού** προγραμματισμού παρέχει και **ειδικό** συντάκτη για την δημιουργία **γραφικών** (μενού διαλόγου κλπ)

Κεφάλαιο 7 Βασικές Έννοιες Προγραμματισμού

Οι γλώσσες προγραμματισμού χρησιμοποιούνται για την επίλυση προβλημάτων με υπολογιστή. Η επιλογή της κατάλληλης γλώσσας εξαρτάται από το είδος του προγράμματος, τον εξοπλισμό και τις γνώσεις του προγραμματιστή.

Κάθε γλώσσα

- είναι σχεδιασμένη για συγκεκριμένο σκοπό,
- εξελίσσεται και
- έχει κοινά χαρακτηριστικά με όλες τις άλλες

7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ

Αποτελείται από:

- Γράμματα
  - Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
  - Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
- Ψηφία 0-9
- Ειδικοί χαρακτήρες : + - * / ^ = > < , . ! & [ ] ( ) _
  Και ο κενός χαρακτήρας

7.2 Τύποι δεδομένων

- **Ακέραιος**. Ο τύπος αυτός περιλαμβάνει τους ακέραιους (το σύνολο Z) που είναι γνωστοί από τα μαθηματικά. (δηλαδή **θετικούς** και **αρνητικούς** καθώς και το **0**)
- **Πραγματικός**. Ο τύπος αυτός περιλαμβάνει τους πραγματικούς (το σύνολο R) που είναι γνωστοί από τα μαθηματικά. (δηλαδή αριθμούς με **δεκαδικό** μέρος **θετικούς** και **αρνητικούς** καθώς και το **0**) *{Πρακτικά: Η διαφορά μεταξύ τους είναι μια μεταβλητή πραγματικού τύπου μπορεί να πάρει **και** δεκαδικές τιμές }*
- **Λογικός**. Ο τύπος αυτός δέχεται μόνο δύο τιμές **ΑΛΗΘΗΣ** και **ΨΕΥΔΗΣ**.
- **Χαρακτήρας**. Ο τύπος αυτός αναφέρεται τόσο σε **ένα χαρακτήρα** όσο και σε μία **σειρά** χαρακτήρων. Περιέχει οποιοδήποτε χαρακτήρα μπορεί να γραφεί στο πληκτρολόγιο. Οι

χαρακτήρες πρέπει να βρίσκονται υποχρεωτικά ανάμεσα σε **μονά** εισαγωγικά. Επειδή μπορεί να περιέχει και αριθμούς ονομάζεται συχνά και **αλφαριθμητικός** τύπος.

{Πχ ‘ αριθμός ’ ‘ μαθητής 3ος’ }

7.3 Σταθερές

{Ορισμός} Είναι **προκαθορισμένες** τιμές που δεν **μεταβάλλονται** κατά την διάρκεια **εκτέλεσης** του προγράμματος. Μπορεί να είναι τύπου ακέραιες, πραγματικές, λογικές, χαρακτήρες.

Δήλωση Σταθερών

Η δήλωση των σταθερών γίνεται στο τμήμα δήλωσης σταθερών:

Σύνταξη:

Σταθερές

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

Π.χ.:

Σταθερές
Π=3.14
Όνομα=’Κώστας’

Λειτουργία του τμήματος δήλωσης σταθερών:
Αποδίδει ονόματα σε σταθερές τιμές. Κάθε μία από αυτές τις σταθερές μπορεί να χρησιμοποιηθεί οπουδήποτε στο πρόγραμμα αλλά δεν είναι δυνατή η μεταβολή της τιμής της.

7.4 Μεταβλητές

{Ορισμός} Μια μεταβλητή παριστάνει μία **ποσότητα** που η **τιμή** της μπορεί να **μεταβάλλεται**.

Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα αντιστοιχούνται από τον **μεταγλωττιστή** σε συγκεκριμένες **θέσεις** μνήμης.

Η τιμή της μεταβλητής είναι η τιμή της **αντίστοιχης** θέσης μνήμης.

Ενώ η τιμή της μεταβλητής (θέσης μνήμης) μπορεί να αλλάξει, ο **τύπος** της **δεν** αλλάζει ποτέ.

Η **ΓΛΩΣΣΑ** επιτρέπει την χρήση **τεσσάρων** τύπων δεδομένων (ακέραιες λογικές πραγματικές χαρακτήρες) Το **όνομα** κάθε μεταβλητής ακολουθεί τους **κανόνες** δημιουργίας ονομάτων.

Είναι καλή πρακτική να χρησιμοποιούμε ονόματα μεταβλητών που **υπονοούν** το περιεχόμενό τους.

Δήλωση Μεταβλητών

(Η δήλωση των μεταβλητών γίνεται στο τμήμα δήλωσης μεταβλητών)

- **Σύνταξη** του τμήματος δήλωσης:

  Μεταβλητές

  Ακέραιες: Λίστα-μεταβλητών-1
  Πραγματικές: Λίστα-μεταβλητών-2
  Λογικές: Λίστα-μεταβλητών-3
  Χαρακτήρες: Λίστα-μεταβλητών-4

- **Λειτουργία** του τμήματος δήλωσης:

Δηλώνει τον **τύπο όλων** των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα

Ονόματα προγραμμάτων και δεδομένων

Κάθε πρόγραμμα όπως και τα δεδομένα (μεταβλητές και σταθερές) που χρησιμοποιεί έχουν ένα όνομα με το οποίο αναφερόμαστε σε αυτό.

Τα ονόματα αυτά μπορούν να αποτελούνται από:

- Γράμματα:
  - Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
  - Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
- Ψηφία: 0-9
- Ειδικούς χαρακτήρες: _κάτω παύλα ή underscore)

Περιορισμοί:

- Απαγορεύεται η χρήση δεσμευμένων λέξεων ως ονόματα (πχ γράψε διάβασε)
- Τα ονόματα υποχρεωτικά αρχίζουν από γράμμα (όχι ψηφίο ούτε underscore)

Πχ «εμβαδόν» και όχι σκέτο «Ε»

7.5 **Αριθμητικοί Τελεστές**

Τελεστής	**Αριθμητική πράξη**
^	Ύψωση στην δύναμη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
mod	**Υπόλοιπο** ακέραιας διαίρεσης
div	**Πηλίκο** ακέραιας διαίρεσης

Ιεραρχία αριθμητικών τελεστών

1. ^
2. * / div mod
3. + -

Προσοχή:

- Αν συναντάμε **παρενθέσεις** προηγούνται οι τελεστές εντός των παρενθέσεων.
- Όσοι τελεστές έχουν **ίδια** ιεραρχία εκτελούνται από **αριστερά** προς τα δεξιά (όπως στα μαθηματικά).

7.6 Συναρτήσεις

Ενσωματομένες (εγγενείς) συναρτήσεις της ΓΛΩΣΣΑΣ

ΗΜ(x)	**ΣΥΝ(x)**	**ΕΦ(x)**	**ΛΟΓ(x)**	**Ε(χ)**	**Τ_Ρ(x)**	**Α_Μ(x)**	**Α_Τ(x)**
Ημίτονο	Συνημίτονο	Εφαπτομένη	Φυσικός Λογάριθμος	Εκθετική	Ρίζα	Ακέραιο μέρος	Απόλυτη τιμή

7.7 Αριθμητικές Εκφράσεις

{βλέπε και κεφάλαιο 2ο για εκφράσεις γενικότερα}

Όταν μία τιμή προέρχεται από **αριθμητικό υπολογισμό** τότε αναφερόμαστε σε αριθμητικές εκφράσεις.

Για την **σύνταξη** μιας αριθμητικής έκφρασης χρησιμοποιούνται:

- Αριθμητικές **σταθερές**. *{ πχ 3 }*
- Αριθμητικές **μεταβλητές**. *{ πχ x }*
- Αριθμητικοί **τελεστές** *{ πχ + }*
- **Συναρτήσεις**. *{ πχ A_T( ) }*
- **Παρενθέσεις**.

Κάθε έκφραση παριστάνει μία αριθμητική τιμή η οποία βρίσκεται **μετά** την εκτέλεση των πράξεων. Γι' αυτό είναι **απαραίτητο** όλες οι μεταβλητές που εμφανίζονται σε μία έκφραση να έχουν πάρει **προηγούμενα** κάποια τιμή.

Η εκτέλεση των πράξεων γίνεται με βάση την **ιεραρχία** των αριθμητικών τελεστών.

Οι παρενθέσεις πάντα χρησιμοποιούνται σε **ζεύγη**. **Διαφορετικός** αριθμός αριστερών από δεξιές παρενθέσεις στην ίδια έκφραση είναι ένα από τα πιο συνηθισμένα **συντακτικά** λάθη

2.4.1 Δομή ακολουθίας

- Η **ακολουθιακή** δομή εντολών (**σειριακών** βημάτων) χρησιμοποιείται πρακτικά για την αντιμετώπιση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών.
- Η Δομή ενός προγράμματος με ακολουθιακή δομή περιλαμβάνει εντολές εισόδου και εξόδου καθώς και εκφράσεις που υλοποιούνται με την εντολή εκχώρησης
- **Εκφράσεις** (expressions). Οι εκφράσεις διαμορφώνονται από τους τελεστέους (operands), που είναι σταθερές και μεταβλητές και από τους τελεστές. Η διεργασία αποτίμησης μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Η τελική τιμή μιας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση των παρενθέσεων. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση

7.8 Εντολή εκχώρησης

Χρησιμοποιείται για την **απόδοση** τιμών στις **μεταβλητές** κατά την **εκτέλεση** του προγράμματος.

Σε μία εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του **ιδίου τύπου**.

Το σύμβολο "←" διαφοροποιείται από το = το οποίο χρησιμοποιείται ως **συγκριτικός** τελεστής καθώς και ως τελεστής απόδοσης τιμών στο τμήμα δηλώσεων των σταθερών.

- Σύνταξη: Όνομα_μεταβλητής ← έκφραση
- Λειτουργία: Υπολογίζεται η έκφραση δεξιά του ← και η τιμή της εκχωρείται στην μεταβλητή που βρίσκεται αριστερά.
- Παράδειγμα: Α←'όμορφα'

7.9 Εντολές Εισόδου – Εξόδου

Εντολή εισόδου (διάβασε)

- Σύνταξη: Διάβασε λίστα-μεταβλητών
- Λειτουργία: Η εντολή οδηγεί στην είσοδο τιμών από το πληκτρολόγιο και την εκχώρηση τους στις μεταβλητές που αναφέρονται στην λίστα.
- Παράδειγμα: ΔΙΑΒΑΣΕ χ,ψ

Ποιο αναλυτικά:

Η εντολή διάβασε ακολουθείται πάντα από τουλάχιστον ένα **όνομα μεταβλητής**. Αν υπάρχουν περισσότερες τότε αυτές χωρίζονται με **κόμμα** (,). **Διακόπτεται** η εκτέλεση του προγράμματος καθώς το πρόγραμμα **περιμένει** την εισαγωγή τιμών από το **πληκτρολόγιο**. Μόλις εισαχθούν οι τιμές συνεχίζεται η εκτέλεση του προγράμματος στην **επόμενη** εντολή. Εξυπηρετεί την **επικοινωνία** του **χρήστη** με τον υπολογιστή.

Εντολή εξόδου (γράψε)

- Σύνταξη: Γράψε λίστα-στοιχείων
- Λειτουργία: Εμφανίζει σταθερές τιμές καθώς και τιμές μεταβλητών (ότι αναγράφεται στην λίστα)
- Παράδειγμα: ΓΡΑΨΕ ‘ Ο φόρος είναι ’ , φπα, ’ € ’

Ποιο αναλυτικά:

Έχει ως αποτέλεσμα την εμφάνιση τιμών στην **μονάδα εξόδου**. Η μονάδα εξόδου μπορεί να είναι η **οθόνη**, ο **εκτυπωτής**, ή γενικά οποιαδήποτε μονάδα εξόδου έχει **οριστεί**. Η λίστα στοιχείων μπορεί να περιέχει **σταθερές** τιμές και **ονόματα** μεταβλητών. Όταν κάποιο όνομα μεταβλητής περιέχεται στην λίστα τότε αρχικά **ανακτάται** η τιμή της και μετά η τιμή αυτή **εμφανίζεται** στην μονάδα εξόδου. Η χρήση της αναφέρεται κυρίως στην εμφάνιση **μηνυμάτων** και **αποτελεσμάτων.** Εξυπηρετεί την **επικοινωνία** του **υπολογιστή** με τον χρήστη.

7.10 Δομή προγράμματος

Δομή προγράμματος σε ΓΛΩΣΣΑ

- **Επικεφαλίδα**: Πρόγραμμα Όνομα-προγράμματος Το όνομα πρέπει να ακολουθεί τους κανόνες δημιουργίας ονομάτων
- **Τμήμα δήλωσης σταθερών**: Εδώ δηλώνονται οι τυχόν σταθερές που χρησιμοποιεί το πρόγραμμα.
- **Τμήμα δήλωσης μεταβλητών**:Δηλώνονται υποχρεωτικά τα ονόματα όλων των μεταβλητών μαζί με τον τύπο τους.
- **Κύριο μέρος**: Περιλαμβάνει όλες τις εκτελέσιμες εντολές ανάμεσα στην αρχή και στο τέλος_προγράμματος.

Παρατηρήσεις:

- Κάθε εντολή γράφεται σε **ξεχωριστή γραμμή**. Αν κάποια εντολή πρέπει να συνεχιστεί σε επόμενη γραμμή τότε ο πρώτος χαρακτήρας της επόμενης πρέπει να είναι ο: **&.**
- Αν ο **πρώτος** χαρακτήρας μίας γραμμής είναι ο **!** τότε η γραμμή αυτή αποτελεί **σχόλιο** και **δεν εκτελείται**
- Αν το πρόγραμμα χρησιμοποιεί **υποπρογράμματα** τότε αυτά γράφονται **μετά** το **τέλος_προγράμματος.**

*{οι παρακάτω **ενότητες** αναφέρονται στη Δομή της Επιλογής γενικότερα και αναφέρονται στην ενότητα 11 της προτεινόμενης Διδασκαλίας }*

11	2.4.2, 2.4.3, 2.4.4, 8.1, 8.1.1, 8.1.2	3.1, 3.1.1, 3.1.2	Δομή επιλογής, Διαδικασίες πολλαπλών επιλογών, Εμφωλευμένες διαδικασίες, Εντολές επιλογής, Εντολή ΑΝ, Εντολή ΕΠΙΛΕΞΕ

Δομή ενός αλγόριθμου σε ψευδογλώσσα

Αλγόριθμος όνομα
Δεδομένα // ... //
Εντολές
Αποτελέσματα // ... //
Τέλος όνομα

ΣΧΟΛΙΑ:

Στα δεδομένα αναγράφουμε ανάμεσα στις κάθετες γραμμές όλες τις μεταβλητές ή/και πίνακες που εμφανίζονται σε εντολές εισόδου, ενώ στα αποτελέσματα όλες τις μεταβλητές ή/και πίνακες που εμφανίζονται σε εντολές εξόδου.

Προσοχή

λίγο στο γεγονός ότι όταν δηλώνουμε πίνακες στα δεδομένα ή στα αποτελέσματα ΔΕΝ τοποθετούμε αγκύλες στους πίνακες άρα δεν δηλώνουμε το μέγεθός τους !!
Εδώ ίσως αναρωτηθείς ... μα που το ξέρει ο υπολογιστής το μέγεθος του πίνακα;

Απάντηση:

ο αλγόριθμος δεν είναι πρόγραμμα αλλά μια έκφραση ενός αλγορίθμου σε ψευδογλώσσα άρα απευθύνεται σε έναν άνθρωπο και όχι στην μηχανή!
Πχ Να γραφεί αλγόριθμος σε ψευδογλώσσα που υπολογίζει και εμφανίζει το εμβαδά ενός τριγώνου.

***Αλγόριθμος** Τρίγωνο*
***Δεδομένα** // βάση,ύψος //*
*Εμβαδό<--βάση*ύψος/2*
***Αποτελέσματα** // Εμβαδό //*
***Τέλος** όνομα*

Κεφάλαιο 8 Επιλογή Επανάληψη

8.1 Εντολές Επιλογής

- Όπως πολύ συχνά συμβαίνει και στην καθημερινή μας ζωή, έτσι και στον προγραμματισμό, είναι ανάγκη να λαμβάνονται κάποιες αποφάσεις ανάλογα με το αν ισχύουν ή όχι κάποια ή κάποιες συνθήκες.
- Για να μπορέσουμε να επιτύχουμε τη λήψη αποφάσεων και τη σωστή δρομολόγηση του προγράμματος ώστε να εκτελεί τις κατάλληλες εντολές για κάθε περίπτωση, χρησιμοποιούμε τη δομή της επιλογής .
- **Η** διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συν θήκης (λογική έκφραση) που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής) και ακολουθεί η απόφαση εκτέλεσης κάποιας ενέργειας με βάση την τιμή της λογικής αυτής συνθήκης

Τι είναι λογική έκφραση:

Μία λογική έκφραση είναι μία έκφραση που έχει λογική τιμή δηλαδή αληθής ή ψευδής. Για την σύνταξη μιας λογικής έκφρασης χρησιμοποιούνται: Σταθερές, μεταβλητές, αριθμητικές εκφράσεις, συγκριτικοί τελεστές, λογικοί τελεστές, και παρενθέσεις

Ποιοι είναι οι συγκριτικοί τελεστές

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

Συγκρίσεις

Οι συγκρίσεις έχουν ως αποτέλεσμα μία λογική τιμή (Αληθής-ψευδής)

Γίνονται σε δεδομένα όλων των τύπων της ΓΛΩΣΣΑΣ :

- Σύγκριση **αριθμών**: Η σύγκριση μεταξύ δύο αριθμών γίνεται με **προφανή** τρόπο.
- Σύγκριση **χαρακτήρων**: Η σύγκριση μεταξύ **ατομικών** χαρακτήρων βασίζεται στην **αλφαβητική** σειρά ΠΧ το "α" < "β" : αληθής Η σύγκριση **σειράς** χαρακτήρων βασίζεται στην σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρι να βρεθεί κάποια **διαφορά**.
  { Πχ "κακός" < "καλός" : αληθής (γιατί το κ είναι μικρότερο του λ) }
- Σύγκριση **Λογικών**: Η σύγκριση μεταξύ λογικών έχει **νόημα** μόνο στην περίπτωση του **=** και του **<>**

Σύνθετες λογικές εκφράσεις

Σχηματίζονται από **απλές** λογικές εκφράσεις με την χρήση των λογικών τελεστών: Όχι, Και, Η

Π.χ.

0 < X < 5 X > 0 ΚΑΙ X < 5

X = 1 ή 2 ή 3 X = 1 Ή X = 2 Ή X = 3

Η ιεραρχία των λογικών τελεστών είναι μικρότερη των αριθμητικών.

Σχετική ιεραρχία των τελεστών

1. αριθμητικοί τελεστές,
2. συγκριτικοί,
3. λογικοί.

Οι μορφές της Δομής Επιλογής

Υπάρχουν τέσσερις μορφές επιλογής :

1. Η Απλή Επιλογή
2. Η Σύνθετη Επιλογή
3. Η Πολλαπλή Επιλογή , και
4. Η Εμφωλευμένη Επιλογή

Η δομή επιλογής υλοποιείται με την εντολή "Αν". Η εντολή "Αν" έχει τρεις διαφορετικές μορφές:

Απλή επιλογή	**Σύνθετη επιλογή**	**Πολλαπλή επιλογή**	**Εμφωλευμενη επιλογή**
Αν .. τότε	Αν..τότε αλλιώς	Αν..τότε αλλιώς_αν	Αν..τότε Αν..τότε Τέλος_αν Τέλος_αν

Προσοχή: Κάθε εντολή Αν πρέπει να κλείνει με τέλος_αν

*{προσοχή όμως εδώ: Σύμφωνα με το σχολικό σου (κεφάλαιο 2) σε μια ψευδογλώσσα όταν οι εντολές εντός μιας απλής επιλογής είναι μια και **μόνο** μία, δικαιούμαστε να μην βάλουμε τελος_αν αρκεί να γράψουμε την εντολή δίπλα από την λέξη τότε.*

8.1.1 Εντολή ΑΝ

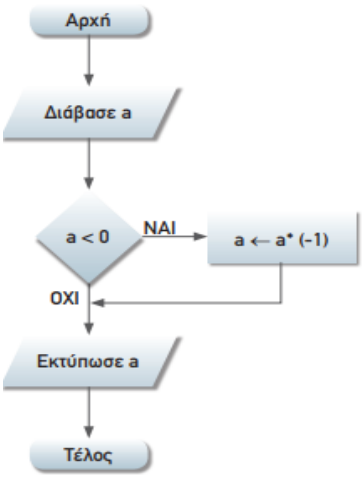
2.4.2 Δομή επιλογής

[ΒΙΒΛΙΟ 4]: «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον», Τετράδιο Μαθητή, Γ΄ Γενικού Λυκείου

1. **απλή** επιλογή

Αν *συνθήκη* **τότε**
εντολή ή εντολές
Τέλος_αν

Να διαβαστεί ένας αριθμός και να εκτυπωθεί η απόλυτη τιμή του.



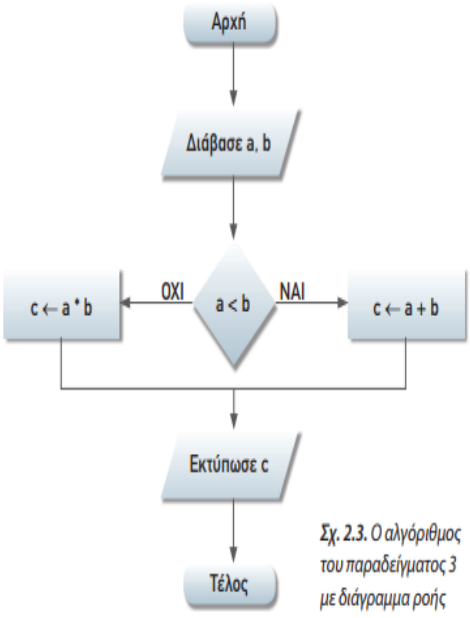

```
Αλγόριθμος Παράδειγμα_2
        Διάβασε a
        Αν a < 0 τότε
              a ←a*(-1)
        Τέλος_αν
        Εκτύπωσε a
Τέλος Παράδειγμα_2
```

2. **σύνθετη επιλογή**

Σε πολλές όμως περιπτώσεις θα χρειάζεται να εκτελεστεί μια ομάδα εντολών ακόμη και στη περίπτωση όπου δεν ισχύει η λογική συνθήκη της επιλογής.

Αν *συνθήκη* **τότε**
εντολή ή εντολές
αλλιώς
εντολή ή εντολές
Τέλος_αν

Να διαβασθούν δύο αριθμοί και σε περίπτωση που ο πρώτος αριθμός είναι μικρότερος του δεύτερου, να υπολογισθεί και να εκτυπωθεί το άθροισμά τους, διαφορετικά να υπολογισθεί και να εκτυπωθεί το γινόμενό τους.



Σχ. 2.3. Ο αλγόριθμος του παραδείγματος 3 με διάγραμμα ροής

```
Αλγόριθμος Παράδειγμα_3
   Διάβασε a, b
   Αν a < b τότε
        c ←a + b
   αλλιώς
        c ←a * b
   Τέλος_αν
   Εκτύπωσε c
Τέλος Παράδειγμα_3
```

2.4.3 Διαδικασίες πολλαπλών επιλογών

[ΒΙΒΛΙΟ 4]: «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον», Τετράδιο Μαθητή, Γ΄ Γενικού Λυκείου

- Χρησιμοποιούμε την δομή της πολλαπλής επιλογής όταν υπάρχουν περισσότεροι από 2 εναλλακτικοί «δρόμοι» τους οποίους μπορεί να επιλέξει το πρόγραμμα για την εκτέλεσή του, ανάλογα με το ποια λογική συνθήκη ικανοποιείται κάθε φορά.
- Στη δομή της πολλαπλής επιλογής μπορούμε να έχουμε στην ίδια επιλογή περισσότερες από μια λογικές συνθήκες.
- Πρέπει να επισημάνουμε ότι στην δομή αυτή κάθε φορά ικανοποιείται μόνο μια λογική συνθήκη (ανεξάρτητα από το πόσες έχουμε). Έτσι μόλις βρεθεί η λογική συνθήκη η οποία ικανοποιείται αμέσως εκτελείται η ομάδα εντολών που ανήκει στη συνθήκη αυτή και τερματίζεται η επιλογή, αυτό άμεσα σημαίνει ότι όλες οι άλλες συνθήκες είναι ψευδής (έχουμε δηλαδή μόνο μια συνθήκη αληθής).

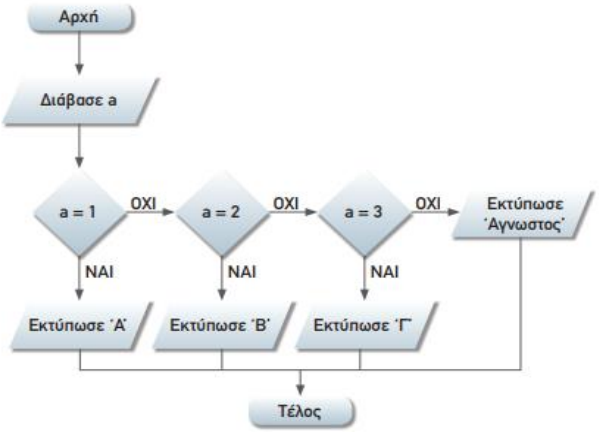
Σύνταξη:

```
Αν συνθήκη-1 τότε
      Ομάδα εντολών 1
Αλλιώς_αν συνθήκη-2 τότε
      Ομάδα εντολών 2
        . . .
Αλλιώς
      Ομάδα εντολών ν
Τελος_αν
```

Λειτουργία ο Εκτελούνται οι εντολές που βρίσκονται στο **αντίστοιχο** τμήμα όταν η συνθήκη είναι αληθής.
ο Η εκτέλεση **συνεχίζεται** την εντολή μετά το τέλος_αν.

Να διαβασθεί ένας ακέραιος και να εκτυπωθεί το αντίστοιχο γράμμα της αλφαβήτου αν ο ακέραιος έχει τιμή 1 ή 2 ή 3, διαφορετικά να εκτυπωθεί η λέξη "άγνωστος".

```
Αλγόριθμος Παράδειγμα
Διάβασε a
Αν a = 1 τότε
      εκτύπωσε "Α"
αλλιώς_αν a = 2 τότε
      εκτύπωσε "Β"
αλλιώς_αν a = 3 τότε
      εκτύπωσε "Γ"
αλλιώς
      εκτύπωσε "άγνωστος"
Τέλος_αν
Τέλος Παράδειγμα
```



[ΒΙΒΛΙΟ 4]: «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον», Τετράδιο Μαθητή, Γ΄ Γενικού Λυκείου

2.4.4 Εμφωλευμένες Διαδικασίες

[ΒΙΒΛΙΟ 4]: «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον», Τετράδιο Μαθητή, Γ΄ Γενικού Λυκείου

Πολλαπλές επιλογές μπορούν να γίνουν και με μία εμφωλευμένη δομή

{Ορισμός:} *Εμφωλευμένα ΑΝ ονομάζονται δύο ή περισσότερες εντολές της μορφής* ***ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ*** *που περιέχονται η μία μέσα στην άλλη.*

{εμφωλευμένες μπορεί να είναι και εντολές επανάληψης αλλά και συνδυασμός δομών επιλογής και επανάληψης}

Π.χ.

```
,,,,,,,
ΔΙΑΒΑΣΕ Βάρος,Ύψος
ΑΝ Βάρος < 80 ΤΟΤΕ
        ΑΝ Ύψος < 1.70 ΤΟΤΕ
                ΓΡΑΨΕ 'Ελαφρύς, κοντός'
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
,,,,,
```

Η χρήση εμφωλευμένων εντολών **ΑΝ** οδηγεί συνήθως σε πολύπλοκες δομές που αυξάνουν την πιθανότητα του λάθους καθώς και τη δυσκολία κατανόησης του προγράμματος. Πολύ συχνά οι εντολές που έχουν γραφεί με εμφωλευμένα **ΑΝ**, μπορούν να γραφούν πιο απλά χρησιμοποιώντας σύνθετες εκφράσεις ή την εντολή επιλογής **ΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ_ΑΝ**, που θα παρουσιαστεί στη συνέχεια. Το προηγούμενο τμήμα προγράμματος μπορεί να γραφεί ως εξής:

```
,,,,,,,
ΔΙΑΒΑΣΕ Βάρος,Ύψος
ΑΝ Βάρος < 80 ΚΑΙ Ύψος <1.70 ΤΟΤΕ
        ΓΡΑΨΕ 'Ελαφρύς, κοντός'
ΤΕΛΟΣ_ΑΝ
,,,,,
```

{Δες επιπλέον παράδειγμα σελ 142 και 143}

ΠΡΟΣΟΧΗ στην περίπτωση κλιμακωτής χρέωσης

8.1.2 Εντολή ΕΠΙΛΕΞΕ

Αν οι εναλλακτικές περιπτώσεις επιλογής είναι πολλές, μπορεί να χρησιμοποιηθεί η εντολή ΕΠΙΛΕΞΕ, η γενική μορφή της οποίας είναι:
Να αποφεύγεται, αν είναι δυνατόν, η χρήση των εμφωλευμένων ΑΝ, και στη θέση τους να χρησιμοποιούνται απλούστερες δομές που διευκολύνουν την ανάγνωση και την κατανόηση του προγράμματος.

Σύνταξη

```
ΕΠΙΛΕΞΕ έκφραση
ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_1
        εντολές_1
ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_2
        εντολές_2 ......
ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
        εντολές_αλλιώς
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

Παράδειγμα

```
ΔΙΑΒΑΣΕ αριθμός
ΕΠΙΛΕΞΕ αριθμός
ΠΕΡΙΠΤΩΣΗ 0
        ΓΡΑΨΕ 'Μηδέν'
```

```
ΠΕΡΙΠΤΩΣΗ 1,3,5,7,9
      ΓΡΑΨΕ 'Μονός αριθμός'
ΠΕΡΙΠΤΩΣΗ 2,4,6,8
      ΓΡΑΨΕ 'Ζυγός '
ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
      ΓΡΑΨΕ 'Αριθμός < 0 ή >9 ή όχι ακέραιος'
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

Λειτουργία

Υπολογίζεται η τιμή της έκφρασης και εκτελούνται οι εντολές που ανή- κουν στην αντίστοιχη περίπτωση τιμών. Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμία περίπτωση, τότε εκτελούνται οι εντολές αλλιώς.

Στην εντολή αυτή οι λίστες τιμών που συνοδεύουν κάθε περίπτωση μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές ή περιοχή τιμών από-έως.
Η χρήση της εντολής **ΕΠΙΛΕΞΕ** λόγω της συμπαγούς δομής της προσφέ- ρει σημαντικά πλεονεκτήματα στον προγραμματισμό.

{Δες επιπλέον Ενότητα 3.1, ***ΠΑΡΑΡΤΗΜΑ Α ΟΔΗΓΙΕΣ ΜΕΛΕΤΗΣ ΜΑΘΗΤΗ(2η Έκδοση)*** *σελ. 21-34 και* Ενότητες 3.1.1, 3.1.2 ***ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟΕΝΔΕΙΚΤΙΚΕΣ ΛΥΣΕΙΣ ΑΣΚΗΣΕΩΝ*** *σελ. 48}*

2.4.5 Δομή επανάληψης

Σελ 44-51 αναλύονται στις επόμενες ενότητες

8.2 Εντολές Επανάληψης

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

Η δομή επανάληψης έχει τρεις μορφές-εντολές:

- Την δομή «Για»
- την δομή «όσο» και
- την δομή «μέχρις_ότου».

Όλες ελέγχονται από μία συνθήκη ή οποία καθορίζει την έξοδο από τον **βρόχο**.

{ Προσοχή: βρόχος γενικά σημαίνει θηλειά αλλά στην πληροφορική σημαίνει δομή επανάληψης }

8.2.1 Εντολή ΟΣΟ … ΕΠΑΝΑΛΑΒΕ

- **Σύνταξη**:
  Όσο συνθήκη επανάλαβε
  Εντολή-1
  Εντολή-2
  ..
  Εντολή-ν
  Τελος_επανάληψης
- **Λειτουργία** ○ **Ελέγχεται** η συνθήκη και αν είναι **αληθής** εκτελούνται οι εντολές που βρίσκονται **ανάμεσα** στις όσο και τέλος_επανάληψης.
  - Στην συνέχεια **ελέγχεται πάλι** η συνθήκη και αν ισχύει εκτελούνται πάλι οι ίδιες εντολές.
  - Όταν η συνθήκη γίνει **ψευδής** τότε **σταματά** η επανάληψη και εκτελείται η **εντολή** μετά το τέλος_επανάληψης. *{αν βέβαια υπάρχει}*
  - Με την δομή ΟΣΟ μπορούν να εκφραστούν **όλες** οι άλλες δομές επανάληψης ○ Χαρακτηριστικό της είναι ότι ο αριθμός των επαναλήψεων **δεν** είναι γνωστός.
  - Για να σταματήσει η επανάληψη πρέπει **υποχρεωτικά** μέσα στον βρόχο να υπάρχει μία **εντολή** η οποία κάνει την συνθήκη **ψευδή**.

8.2.2 Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ

- **Σύνταξη**:

  Αρχή_επανάληψης
  Εντολή-1
  Εντολή-2
  ..
  Εντολή-ν
  Μέχρις_οτου <*συνθήκη* >

- **Λειτουργία**
  - Εκτελούνται οι εντολές **μεταξύ** των αρχή_επανάληψης και μέχρις_οτου.
  - Στη συνέχεια **ελέγχεται** η συνθήκη και αν είναι **ψευδής** τότε εκτελούνται **πάλι** οι εντολές.
  - Όταν η συνθήκη βρεθεί **αληθής** τότε **σταματά** η επανάληψη και εκτελείται η εντολή **μετά** το μέχρις_ότου..

Η δομή αυτή εκτελείται τουλάχιστον μια φορά.

8.2.3 Εντολή ΓΙΑ ... ΑΠΟ ... ΜΕΧΡΙ

- **Σύνταξη**:

  Για μεταβλητή **από** τιμή1 **μέχρι** τιμή2 **με βήμα** τιμή3
  Εντολή-1
  Εντολή-2
  ..
  Εντολή-ν
  Τελος_επανάληψης

- **Λειτουργία**
  - Οι εντολές του βρόχου εκτελούνται για **όλες** τις τιμές της μεταβλητής από την τιμή1 μέχρι την τιμή2 αυξανόμενες κατά την τιμή3.
  - Αν το βήμα είναι **1** τότε **παραλείπεται**
  - Χρησιμοποιείται όταν γνωρίζουμε εκ των **προτέρων** τον αριθμό των επαναλήψεων.

Τι ονομάζουμε τιμή φρουρό;

Η χρήση τιμών για τον τερματισμό μιας επαναληπτικής διαδικασίας είναι συνήθης στον προγραμματισμό. Η τιμή αυτή ορίζεται από τον προγραμματιστή και αποτελεί μια σύμβαση για το τέλος του προγράμματος. Η τιμή αυτή είναι τέτοια ώστε να μην είναι λογικά σωστή για το πρόβλημα.

Εμφωλευμένοι βρόχοι και κανόνες χρήσης

{Ορισμός} Ονομάζονται

- δύο ή περισσότεροι βρόχοι που
- βρίσκονται ο ένας μέσα στον άλλο.

Κανόνες χρήσης εμφωλευμένων βρόχων

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

Ολίσθηση

Όταν ο προγραμματιστής ορίζει ένα δεδομένο τότε αυτό αποθηκεύεται στα κυκλώματα του υπολογιστή σε δυαδική μορφή δηλαδή σε ακολουθίες από 0 και 1.
Αν για παράδειγμα ο προγραμματιστής εκτελέσει την εντολή : χ←17 τότε μέσα στην μνήμη του υπολογιστή (η οποία ουσιαστικά είναι ένα κύκλωμα) αποθηκεύεται ο ισοδύναμος αριθμός του δυαδικού συστήματος, ο οποίος είναι το 00010001. Ολίσθηση είναι η μετακίνηση όλων των ψηφίων ενός αριθμού κατά μια θέση.

Διακρίνουμε **δύο** ολισθήσεις:

- Ολίσθηση προς τα **αριστερά**.

  Ισοδυναμεί με πολλαπλασιασμό επί δύο. Μετακινούμε όλα τα ψηφία του προς τα αριστερά προσθέτοντας ένα μηδέν στο τέλος και αγνοώντας το αρχικό μηδέν.

- Ολίσθηση προς τα **δεξιά**.

  Ισοδυναμεί με την ακέραια διαίρεση δια δύο. Μετακινούμε όλα τα ψηφία του προς τα δεξιά, αποκόπτοντας το τελευταίο και προσθέτοντας ένα μηδενικό στην αρχή.

Πολλαπλασιασμός αλά ρωσικά

Τι είναι;

Η πράξη του πολλαπλασιασμού δύο αριθμών δεν εκτελείται από τον υπολογιστή με τον τρόπο που την εκτελούμε εμείς. Εκτελείται με έναν άλλο τρόπο που λέγεται πολλαπλασιασμός αλά ρωσικά.

Περιγραφή της μεθόδου πολλαπλασιασμού (για θετικούς ακέραιους αριθμούς):

Έστω ότι θέλω να πολλαπλασιάσω τους αριθμούς 45 και 19. Οι αριθμοί που πρέπει να πολλαπλασιαστούν γράφονται δίπλα δίπλα και ο πρώτος διπλασιάζεται ενώ ο δεύτερος υπερδιπλασιάζεται (για την ακρίβεια τον υποδιπλασιάζουμε αγνοώντας το δεκαδικό μέρος). Η διαδικασία αυτή επαναλαμβάνεται ώσπου ο δεύτερος να γίνει μηδέν :

Παράδειγμα:

45	19	45
90	9	90
180	4	
360	2	
720	1	720
0		

Τελικά το ζητούμενο γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης όπου αντίστοιχα στην δεύτερη στήλη υπάρχει περιττός αριθμός. Στο παραπάνω παράδειγμα τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη. Δηλαδή είναι 45+90+720=855 !

Αντίστοιχος αλγόριθμος:

```
Αλγόριθμος πολλαπλασιασμός_αλα_ρωσικά
    δεδομένα // Μ1, Μ2 ακέραιοι //
    Ρ←0
    Όσο Μ2 >0 επανάλαβε
        Αν Μ2 mod 2=1 τότε
            Ρ←Ρ+Μ1
        Τελος_αν
    Μ1←Μ1 * 2
    Μ2←Μ2 div 2
    Τελος_επαναληψης
    Αποτελέσματα // Ρ, το γινόμενο των ακεραίων Μ1,Μ2 //
Τέλος πολλαπλασιασμός_αλα_ρωσικά
```

{Παρατήρησε την χρήση των εντολών δεδομένα και αποτελέσματα. Η επεξήγησή τους δίνεται παρακάτω. Παρατήρησε επίσης την εντολή Ρ←Ρ+1. Δεν είναι τυπογραφικό λάθος !!! }

Μετατροπή:

Ο παραπάνω αλγόριθμος μπορεί εύκολα να μετατραπεί ώστε να περιλάβει και την περίπτωση πολλαπλασιασμού αρνητικών ακεραίων. *{ωραία άσκηση για τεστάκι.}*

Γιατί ο υπολογιστής εκτελεί τον πολλαπλασιασμό αλά ρωσικά;

- Γιατί υλοποιείται πιο απλά και λιγότερο χρονοβόρα σε σχέση με τον χειρωνακτικό τρόπο πολλαπλασιασμού.
- Απαιτεί μόνο πολλαπλασιασμό επί δύο, διαίρεση δια δύο και πρόσθεση σε αντίθεση με την γνωστή μας διαδικασία πολλαπλασιασμού που απαιτεί πολλαπλασιασμό με οποιοδήποτε ακέραιο και πρόσθεση.
- Σε επίπεδο κυκλωμάτων ο διπλασιασμός υλοποιείται ταχύτατα με εντολή ολίσθησης προς τα αριστερά ενώ ο υποδιπλασιασμός με ολίσθηση προς τα δεξιά , σε αντίθεση με τον πολλαπλασιασμό με οποιονδήποτε ακέραιο που θεωρείται χρονοβόρα διαδικασία.
- Το τελευταίο γεγονός είναι ο λόγος που ο πολλαπλασιασμός αλά ρωσικά είναι προτιμότερος απ' ότι ο χειρωνακτικός τρόπος πολλαπλασιασμού δύο ακεραίων.

Κεφάλαιο 13 Εκσφαλμάτωση προγράμματος

13.1 Κατηγορίες λαθών

Ενότητα 5 Εκσφαλμάτωση Προγράμματος(από ΒΙΒΛΙΟ ΜΑΘΗΤΗ 2 ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

5.1 Κατηγορίες λαθών

5.2.1 Εκσφαλμάτωση λογικών λαθών στις Δομές Επιλογής

5.1.2 Εκσφαλμάτωση λογικών λαθών στις Δομές Επανάληψης

Μετατροπές από μια Δομή Επανάληψης σε άλλη

Γενικές ασκήσεις εμπέδωσης μέχρι και τη δομή Επανάληψης

Ενότητα 2 Τεχνικές Σχεδίασης Αλγορίθμων (από ΒΙΒΛΙΟ ΜΑΘΗΤΗ 2 ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

2.1 Μέθοδος διαίρει και βασίλευε (μόνο επαναληπτική και όχι αναδρομική προσέγγιση)

Κεφάλαιο 3 Δομές Δεδομένων και Αλγόριθμοι

3.1 Δεδομένα

Τα δεδομένα (data) είναι η αφαιρετική αναπαράσταση της πραγματικότητας και συνεπώς μία απλοποιημένη όψη της.

η Πληροφορική ορίζεται και σε σχέση με την έννοια των δεδομένων. Έτσι, Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

- **Υλικού**. Το υλικό (hardware), δηλαδή η μηχανή, επιτρέπει στα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες αναπαραστάσεις (representations). Τέτοιες μορφές είναι η δυαδική, ο κώδικας ASCII (βλ. παράρτημα), ο κώδικας EBCDIC, το συμπλήρωμα του 1 ήτου 2 κ.λπ.
- **Γλωσσών προγραμματισμού**. Οι γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων τύπων (types) μεταβλητών (variables) για να περιγράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον υπολογιστή.
- **Δομών Δεδομένων**. Δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών. Για παράδειγμα, μία τέτοια δομή είναι η εγγραφή (record), που μπορεί να περιγράφει ένα είδος, ένα πρόσωπο κ.λπ. Η εγγραφή αποτελείται από τα πεδία (fields) που αποθηκεύουν χαρακτηριστικά (attributes) διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κ.λπ. Άλλη μορφή δομής δεδομένων είναι το αρχείο που αποτελείται από ένα σύνολο εγγραφών. Μία επιτρεπτή λειτουργία σε ένα αρχείο είναι η σειριακή προσπέλαση όλων των εγγραφών του.
- Ί **Ανάλυσης Δεδομένων**. Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των Βάσεων Δεδομένων (Databases), της Μοντελοποίησης Δεδομένων (Data Modelling) και της Αναπαράστασης Γνώσης (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

3.2 Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

{Ορισμός}. Είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στην δευτερεύουσα. Η αποθήκευση δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή. Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).

{δηλαδή: ένας πίνακας 100 θέσεων έχει ακριβώς 100 κόμβους}

Βασικές λειτουργίες (πράξεις) των δομών δεδομένων

1. **Προσπέλαση**: πρόσβαση σε ένα κόμβο με σκοπό να εξεταστεί ή να αλλάξει το περιεχόμενό του.
2. **Εισαγωγή**: προσθήκη νέων κόμβων σε μία δομή.
3. **Διαγραφή**: (αντίθετο της εισαγωγής), δηλαδή αφαίρεση ενός κόμβου από μία δομή.
4. **Αναζήτηση**: γίνεται προσπέλαση των κόμβων μίας δομής προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μία δεδομένη ιδιότητα.
5. **Ταξινόμηση**: όπου οι κόμβοι διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
6. **Αντιγραφή**: όλοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη.
7. **Συγχώνευση**: δύο ή περισσότερες δομές ενώνονται σε μία δομή.
8. **Διαχωρισμός**: (αντίστροφα από την συγχώνευση)

Κατηγορίες δομών δεδομένων

Οι δομές διακρίνονται σε δύο μεγάλες κατηγορίες :

- Δυναμικές και
- Στατικές

Στατικές δομές

- Το ακριβές **μέγεθός** τους είναι σταθερό και καθορίζεται κατά την στιγμή του προγραμματισμού τους και όχι κατά την στιγμή της εκτέλεσης του προγράμματος.
- Οι κόμβοι τους αποθηκεύονται σε **συνεχόμενες** θέσεις μνήμης.
- Στην πράξη οι στατικές δομές υλοποιούνται με **πίνακες** .

Δυναμικές δομές

- Οι δομές αυτές δεν έχουν σταθερό **μέγεθος** αλλά ο αριθμός των κόμβων τους μεγαλώνει καθώς εισάγονται νέα δεδομένα, και μικραίνει καθώς διαγράφονται δεδομένα από αυτές.
- Δεν αποθηκεύονται σε **συνεχόμενες** θέσεις μνήμης αλλά στηρίζονται στην τεχνική της δυναμικής παραχώρησης μνήμης (DMA : Dynamic Memory Allocation).
- Όλες οι **σύγχρονες** γλώσσες προγραμματισμού τις υποστηρίζουν.
- Δυναμικές δομές είναι το **αρχείο**, η **εγγραφή** κ.α.

*{**Σημείωση** περιέχονται στην σελίδα 57 του νέου σχολικού.}*

3.3 Πίνακες

{Ορισμός} Είναι ένα **σύνολο** αντικειμένων **ιδίου τύπου** τα οποία αναφέρονται με ένα **κοινό όνομα**.

Παρατηρήσεις

Το **όνομα** του πίνακα μπορεί να είναι **οποιοδήποτε δεκτό** όνομα της ΓΛΩΣΣΑΣ και ο δείκτης είναι μία **ακέραια έκφραση**, **σταθερή** ή **μεταβλητή** που περικλείεται μέσα στα σύμβολα **[ ]**

Δήλωση πίνακα:

Κάθε πίνακας πρέπει υποχρεωτικά να περιέχει δεδομένα του **ιδίου τύπου**, δηλαδή ακέραια, πραγματικά, λογικά, ή αλφαριθμητικά. Ο **τύπος** του πίνακα δηλώνεται **μαζί** με τις άλλες μεταβλητές του προγράμματος στο **τμήμα δήλωσης** μεταβλητών. Εκτός από τον **τύπο** του πίνακα πρέπει να δηλώνεται και ο **αριθμός των στοιχείων** που περιέχει ή καλύτερα ο **μεγαλύτερος** αριθμός στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας και αυτό για να **δεσμευτούν** οι αντίστοιχες συνεχόμενες θέσεις μνήμης.

Στοιχείο:

- Η αναφορά σε ένα στοιχείο το πίνακα γίνεται με το **όνομά** του πίνακα ακολουθούμενο από ένα **δείκτη**.
- Κάθε ένα από τα **αντικείμενα** που τον αποτελούν ονομάζεται **στοιχείο**.
- Κάθε **συγκεκριμένη θέση μνήμης** καλείται **στοιχείο** του πίνακα και προσδιορίζεται από την τιμή **ενός** δείκτη αν ο πίνακας είναι **μονοδιάστατος**. Στην περίπτωση που είναι **δισδιάστατος** προσδιορίζεται από **δύο** δείκτες. Αν είναι **ν-διαστατος** από **ν δείκτες**.
- Ο **δείκτης** είναι μία μεταβλητή που μπορεί να έχει οποιοδήποτε δεκτό **όνομα**. Είναι **σύνηθες** όμως στον προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i, j, k.

Πολυδιάστατοι πίνακες

Αν ο καθορισμός των στοιχείων ενός πίνακα απαιτεί παραπάνω από ένα δείκτη τότε μιλάμε για πολυδιάστατους πίνακες: πχ 3 δείκτες τρισδιάστατος 2 δείκτες δισδιάστατος κλπ

Χρήση πινάκων:

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

Μειονεκτήματα από την χρήση πινάκων

1. **Οι πίνακες απαιτούν μνήμη:** Κάθε πίνακας δεσμεύει από την **αρχή** του προγράμματος **πολλές** θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση πινάκων μπορεί να οδηγήσει ακόμη και σε **αδυναμία εκτέλεσης** του προγράμματος.
2. **Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος:** Οι πίνακες είναι **στατικές** δομές δεδομένων αφού το **μέγεθός** τους, δηλώνεται στην αρχή του προγράμματος παραμένει υποχρεωτικά **σταθερό** κατά την **εκτέλεση**.

9.1 Μονοδιάστατοι πίνακες

3.6 Αναζήτηση

4.7 Ταξινόμηση

9.2 Πότε πρέπει να χρησιμοποιούνται οι πίνακες

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

5.2.3 Εκσφαλματωση λογικών λαθών στους πίνακες

(από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

9.3 Πολυδιάστατοι πίνακες

9.4 Τυπικές επεξεργασίες πινάκων

*{ Καλό είναι να τις μελετήσεις σε συνδυασμό με τις τυπικές επεξεργασίες των **δομών δεδομένων** βλ. Κεφάλαιο 3 σελ}*

1. Υπολογισμός **αθροισμάτων** στοιχείων του πίνακα. Πολύ συχνά απαιτείται η εύρεση του αθροίσματος των στοιχείων ενός πίνακα ή του αθροίσματος των στοιχείων που έχουν μία ιδιότητα.
2. Εύρεση **μεγίστου**-**ελαχίστου**. Αν ο πίνακας δεν είναι ταξινομημένος τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, αλλιώς αν είναι ταξινομημένος το μέγιστο και ελάχιστο βρίσκονται προφανώς στα δύο ακριανά στοιχεία του.
3. **Ταξινόμηση** των στοιχείων του πίνακα. Ένας αλγόριθμος ταξινόμησης είναι ο αλγόριθμος φυσαλίδας αν και δεν είναι ο αποδοτικότερος.
4. **Αναζήτηση** ενός στοιχείου του πίνακα. Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι: Η σειριακή αναζήτηση. Είναι η πιο απλή αλλά και η λιγότερη αποτελεσματική μέθοδος. Δυαδική αναζήτηση. Εφαρμόζεται μόνο σε ταξινομημένους πίνακες και σαφώς αποδοτικότερη από την σειριακή μέθοδο

5. **Συγχώνευση** δύο πινάκων. Σκοπός της είναι η ένωση δύο ή περισσότερων ταξινομημένων πινάκων σε έναν που είναι και αυτός ταξινομημένος

3.4 Στοίβα

Ενότητα 1 Δομές Δεδομένων και Αλγόριθμοι (από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

1.1 Στοίβα

1.1.1 Παραδείγματα υλοποίησης στοίβας με χρήση μονοδιάστατου πίνακα

1.1.2 Ερωτήσεις + Ασκήσεις

3.5 Ουρά

(από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

1.2.1 Παραδείγματα υλοποίησης στοίβας με χρήση μονοδιάστατου πίνακα

1.2.2 Ερωτήσεις + Ασκήσεις

Γενικές Ασκήσεις εμπέδωσης με πίνακες

Κεφάλαιο 10 Υποπρογράμματα

10.1 Τμηματικός προγραμματισμός

10.2 Χαρακτηριστικά των υποπρογραμμάτων

10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού

10.4 Παράμετροι

10.5 Διαδικασίες και Συναρτήσεις

10.5.1 Ορισμός και κλήση Συναρτήσεων

10.5.2 Ορισμός και κλήση Διαδικασιών

10.5.3 Πραγματικές και τυπικές παράμετροι

10.6 Εμβέλεια μεταβλητών – σταθερών

5.2.4 Εκσφαλμάτωση λογικών λαθών στα υποπρογράμματα

(από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

Γενικές Ασκήσεις εμπέδωσης με διαδικασίες και συναρτήσεις

13.2 Εκσφαλμάτωση

5.2.5 Μέθοδος «Μαύρο Κουτί»

5.3 Ερωτήσεις - Ασκήσεις

Ενότητα 1 (από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

1.3 Άλλες δομές δεδομένων

1.3.1 Λίστες

1.3.2 Δένδρα

1.3.3 Γράφοι

1.3.4 Ερωτήσεις εμπέδωσης δυναμικών δομών δεδομένων

6.5 Αντικειμενοστραφής προγραμματισμός

Ενότητα 4 (από ΒΙΒΛΙΟ 2 ΜΑΘΗΤΗ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ)

4.1 Αντικειμενοστραφής Προγραμματισμός: ένας φυσικός τρόπος επίλυσης προβλημάτων

4.2 Χτίζοντας Αντικειμενοστραφή Προγράμματα

4.3 Ομαδοποίηση Αντικειμένων σε Κλάσεις: Αφαιρετικότητα και Ενθυλάκωση

4.4 Η Αντικειμενοστραφής «Οικογένεια»: Κλάσεις - Πρόγονοι, Κλάσεις - Απόγονοι

4.5 Ορίζοντας την Κατάλληλη Συμπεριφορά: Πολυμορφισμός

4.6 Ερωτήσεις εμπέδωσης στην αντικειμενοστραφή προσέγγιση