Ανάλυση προβλήματος Αλγόριθμος Δομή ακολουθίας

1

1.1 Η έννοια πρόβλημα

Δώστε τον ορισμό του προβλήματος.
Με τον όρο Πρόβλημα εννοείται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής.

Τι ονομάζουμε επίλυση ενός προβλήματος.
Επίλυση (η αντιμετώπιση) ενός προβλήματος ονομάζουμε τη διαδικασία μέσω της οποίας, ο λύτης του προβλήματος βρίσκει το ζητούμενο (επιτυγχάνει τον επιθυμητό στόχο).

1.2 Κατανόηση προβλήματος

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

Τι σημαίνει ο όρος "δεδομένο";
Με τον όρο δεδομένο δηλώνεται οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις του.

Τι σημαίνει ο όρος "πληροφορία";
Με τον όρο πληροφορία αναφέρεται οποιοδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων.

Τι σημαίνει ο όρος "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.
Επί χιλιετίες ο "μηχανισμός" επεξεργασίας των δεδομένων ήταν και εξακολουθεί να είναι ο ανθρώπινος εγκέφαλος. Στις μέρες μας, ένας άλλος "μηχανισμός" επεξεργασίας δεδομένων είναι ο υπολογιστής.

1.3 Δομή προβλήματος

Τι είναι η δομή ενός προβλήματος;
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.

Τι σημαίνει ανάλυση του προβλήματος;
Η καταγραφή της δομής ενός προβλήματος σημαίνει αυτόματα ότι έχει αρχίσει η διαδικασία ανάλυσης του προβλήματος σε άλλα απλούστερα. Με τη σειρά τους τα νέα προβλήματα μπορούν να αναλυθούν σε άλλα, ακόμη πιο απλά. Η διαδικασία αυτή της ανάλυσης μπορεί να συνεχιστεί μέχρις ότου τα επιμέρους προβλήματα που προέκυψαν θεωρηθούν αρκετά απλά και η αντιμετώπισή τους χαρακτηριστεί ως δυνατή.

Με ποιους τρόπους παρουσιάζεται η ανάλυση ενός προβλήματος;
Η ανάλυση αυτή ενός προβλήματος παρουσιάζεται με δύο τρόπους:

- **Φραστικά**, δηλαδή στη φυσική γλώσσα που μιλάμε.
- **Διαγραμματικά**. Για τη γραφική απεικόνιση της δομής ενός προβλήματος χρησιμοποιείται συχνότατα η διαγραμματική αναπαράσταση. Σύμφωνα με αυτή:

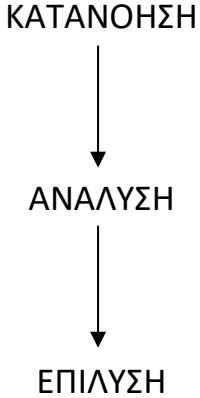
  Το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο.
  Κάθε ένα από τα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο.
  Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα σχηματίζονται ένα επίπεδο χαμηλότερα. Έτσι σε κάθε κατώτερο επίπεδο, δημιουργείται η γραφική αναπαράσταση των προβλημάτων στα οποία αναλύονται τα προβλήματα του αμέσως υψηλότερου επιπέδου.

1.4 Καθορισμός απαιτήσεων

Η σωστή επίλυση ενός προβλήματος προϋποθέτει τον επακριβή προσδιορισμό των δεδομένων τα οποία παρέχει το πρόβλημα. Απαιτεί επίσης τη λεπτομερειακή καταγραφή των ζητουμένων που αναμένονται ως αποτελέσματα της επίλυσης του προβλήματος.

Ποια είναι τα τρία στάδια αντιμετώπισης ενός προβλήματος.

- **Κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος.
- **Ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επιμέρους απλούστερα προβλήματα.
- **Επίλυση**, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.



1.5 Τι είναι αλγόριθμος

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

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια.

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

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

iii. **Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της.

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

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

1.6 Περιγραφή και αναπαράσταση αλγορίθμων

i. **Με ελεύθερο κείμενο (free text),** που αποτελεί τον πιο ανεπεξέργαστο τρόπο παρουσίασης αλγορίθμου. Έτσι ελλοχεύει ο κίνδυνος να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.

ii. **Με διαγραμματικές τεχνικές (diagramming techniques),** που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί, η πιο παλιά και η πιο γνωστή ίσως, είναι το διάγραμμα ροής (flow chart).

iii. **Με φυσική γλώσσα (natural language)** κατά βήματα.
Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιαστεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, το κριτήριο του καθορισμού.

iv. **Με κωδικοποίηση (coding),** δηλαδή με ένα πρόγραμμα που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Ο καλύτερος τρόπος αναπαράστασης ενός αλγορίθμου είναι η κωδικοποίηση, γιατί μπορεί να αναπαραστήσει όλα τα είδη των αλγορίθμων, αλλά δεν είναι εύκολος. Για να μπορέσουμε να καταλάβουμε τι κάνει ο αλγόριθμος πρέπει να διαβάσουμε όλες τις εντολές.

Τα διαγράμματα ροής είναι ιδιαίτερα εποπτικά, με μια ματιά μπορούμε να καταλάβουμε το τι κάνει ο αλγόριθμος, αλλά δεν μπορούν να αναπαραστήσουν όλα τα είδη των αλγορίθμων και γίνονται ιδιαίτερα μεγάλα σε μέγεθος με αποτέλεσμα να είναι δύσχρηστα.

Σε αυτό το βιβλίο θα χρησιμοποιήσουμε την ψευδογλώσσα Γλώσσα (όπως ορίζεται από το σχολικό βιβλίο και τις οδηγίες του Παιδαγωγικού Ινστιτούτου), η οποία μπορεί πάρα πολύ εύκολα να μετατραπεί σε οποιαδήποτε πραγματική γλώσσα προγραμματισμού όπως η Pascal ή η C. Θα δείξουμε επίσης την αντιστοιχία των εντολών της Γλώσσας με τα σχήματα του διαγράμματος ροής.

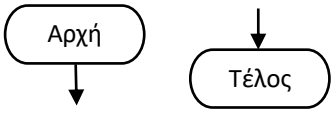
1.7 Βασικές συνιστώσες αλγορίθμου

Είναι οι δομές ακολουθίας, επιλογής και επανάληψης.

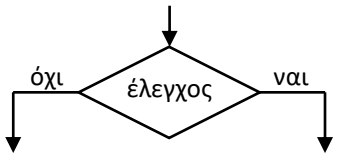
1.8 Διαγράμματα Ροής

Ένα διάγραμμα ροής αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα γεωμετρικά σχήματα ενώνονται μεταξύ τους με βέλη, που δηλώνουν τη σειρά εκτέλεσης των ενεργειών αυτών. Τα κυριότερα γεωμετρικά σχήματα είναι:

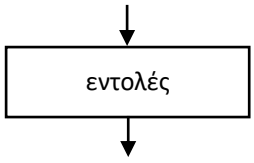
i. **Έλλειψη**, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου.



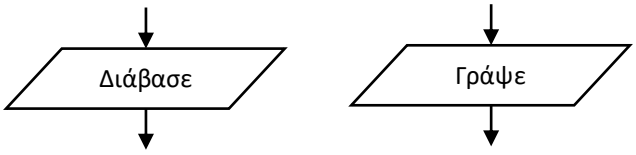
ii. **Ρόμβος**, που δηλώνει έλεγχο με δύο εξόδους (ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ) για απάντηση.



iii. **Ορθογώνιο**, που δηλώνει την εκτέλεση μιας ή περισσότερων πράξεων.



iv. **Πλάγιο παραλληλόγραμμο**, που δηλώνει είσοδο ή έξοδο στοιχείων. Πολλές φορές το σχήμα αυτό μπορεί να διαφοροποιείται προκειμένου να προσδιορίζεται και το είδος της συσκευής από όπου γίνεται η είσοδος ή η έξοδος.



1.9 Βασικά χαρακτηριστικά του ψευδοκώδικα

Ορισμοί

Σταθερές *είναι προκαθορισμένες τιμές που παραμένουν αμετάβλητες κατά τη διάρκεια εκτέλεσης του αλγορίθμου.*

Μεταβλητή *είναι ένα γλωσσικό αντικείμενο που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Στη μεταβλητή εκχωρείται μια τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου.*

Οι τελεστές *είναι τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε αριθμητικούς, λογικούς και συγκριτικούς.*

Εκφράσεις. *Οι εκφράσεις διαμορφώνονται από τις μεταβλητές, τις σταθερές και τους τελεστές. Μία έκφραση μπορεί να αποτελείται από μία μόνο μεταβλητή ή σταθερά μέχρι μία πολύπλοκη μαθηματική παράσταση.*

Για να γράψουμε ένα πρόγραμμα χρησιμοποιώντας ψευδοκώδικα (ή όπως διαφορετικά ονομάζεται Γλώσσα) πρέπει να ακολουθήσουμε κάποιους κανόνες.

Για να δώσουμε όνομα στο πρόγραμμα ή στις μεταβλητές μπορούμε να χρησιμοποιήσουμε χαρακτήρες κεφαλαίους ή πεζούς, ελληνικούς ή λατινικούς και αριθμούς. Δεν μπορούμε να χρησιμοποιήσουμε σύμβολα ή κενά, εκτός από το σύμβολο της κάτω παύλας, που τη χρησιμοποιούμε συνήθως αντί για το κενό. Επίσης δεν επιτρέπεται κανένα όνομα να ξεκινάει από αριθμό πρέπει οπωσδήποτε να ξεκινάει από χαρακτήρα.

Επειδή μερικές λέξεις χρησιμοποιούνται από την ίδια τη Γλώσσα για συγκεκριμένους λόγους, αυτές οι λέξεις δεν μπορούν να χρησιμοποιηθούν ως ονόματα. Οι λέξεις αυτές ονομάζονται δεσμευμένες.

Τύποι μεταβλητών

Η Γλώσσα υποστηρίζει τέσσερις τύπους μεταβλητών.

i. **Ακέραιες.** Μπορούμε να καταχωρίσουμε οποιαδήποτε ακέραια τιμή αρνητική ή θετική (χωρίς δεκαδικό τμήμα).

ii. **Πραγματικές.** Μπορούμε να καταχωρίσουμε οποιονδήποτε αριθμό.

iii. **Χαρακτήρες.** Μπορούμε να καταχωρίσουμε είτε μεμονωμένο χαρακτήρα, είτε λέξη, είτε ολόκληρη πρόταση. Αυτά πρέπει να βρίσκονται υποχρεωτικά μέσα σε εισαγωγικά (απλά ' ' στο πρόγραμμα και διπλά " " στον αλγόριθμο) για να μην μπερδευτούν με ονόματα μεταβλητών ή δεσμευμένων λέξεων.

iv. **Λογικές.** Μπορούμε να καταχωρίσουμε αποκλειστικά και μόνο δύο τιμές, ΑΛΗΘΗΣ και ΨΕΥΔΗΣ.

Πρέπει να γίνεται πολύ προσεκτική επιλογή του τύπου της κάθε μεταβλητής. Λανθασμένη επιλογή μπορεί να οδηγήσει σε ανακρίβεια των αποτελεσμάτων και σε υπερβολική χρήση της μνήμης του υπολογιστή.

Αριθμητικοί τελεστές

Αριθμητικός τελεστής	*Πράξη*
+	Πρόσθεση
–	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
mod	Υπόλοιπο της διαίρεσης
div	Πηλίκο της διαίρεσης

Παραδείγματα χρησιμοποίησης τελεστών.

y ← 3 * x + 2 y ← x / (z – 5)

z ← x mod 2 i ← i + 1

Όταν κάνουμε πράξεις πρέπει να προσέχουμε τη προτεραιότητα των πράξεων. Αυτή είναι:

i. Παρενθέσεις.
ii. Ύψωση σε δύναμη.
iii. Διαιρέσεις - Πολλαπλασιασμοί.
iv. Προσθέσεις - Αφαιρέσεις.

Όταν έχουμε τελεστές με την ίδια προτεραιότητα τότε οι πράξεις εκτελούνται από αριστερά προς τα δεξιά.

Πρέπει επίσης να προσέχουμε και τους τύπους των μεταβλητών μεταξύ των οποίων κάνουμε πράξεις. Πάντα προσπαθούμε να κάνουμε πράξεις μεταξύ μεταβλητών ίδιου τύπου και να καταχωρίσουμε το αποτέλεσμα σε μεταβλητή ίδιου τύπου. Αν δεν μπορούμε να το πετύχουμε αυτό και πρέπει να κάνουμε πράξεις με μεταβλητές διαφορετικού τύπου, τότε καταχωρούμε το αποτέλεσμα σε μεταβλητή του πιο γενικού τύπου (π.χ. αν κάνουμε πράξη μεταξύ μιας ακέραιας και μιας πραγματικής μεταβλητής, το αποτέλεσμα θα το καταχωρίσουμε σε πραγματική μεταβλητή). Ειδική περίπτωση αποτελούν ο τελεστής της διαίρεσης, όπου μας δίνει πάντα πραγματικό αποτέλεσμα και οι τελεστές mod και div που μας δίνουν πάντα ακέραιο αποτέλεσμα.

Συγκριτικοί τελεστές

Συγκριτικός τελεστής	*Έλεγχος*
>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο
=	Ίσο
<>	Διάφορο
<=	Μικρότερο ή ίσο
<	Μικρότερο

Οι συγκριτικοί τελεστές χρησιμοποιούνται οποτεδήποτε θέλουμε να κάνουμε κάποιο έλεγχο, π.χ. η εντολή x = 3 σημαίνει: είναι το x ίσο με 3;

Το αποτέλεσμα ενός ελέγχου θα είναι πάντα *αληθής* ή *ψευδής*, δηλαδή μπορεί να καταχωρηθεί σε μία λογική μεταβλητή.

Λογικοί τελεστές

Οι λογικοί τελεστές χρησιμοποιούνται για να κάνουμε πράξεις μεταξύ λογικών μεταβλητών ή μεταξύ εκφράσεων που έχουν αποτέλεσμα *αληθής* ή *ψευδής*.

Α	*Β*	*Α ΚΑΙ Β*	*Α Η Β*	*ΟΧΙ Α*
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

Προτεραιότητα λογικών πράξεων:

i. Παρενθέσεις
ii. ΟΧΙ
iii. ΚΑΙ
iv. Η

Όταν έχουμε τελεστές με την ίδια προτεραιότητα τότε οι πράξεις εκτελούνται από αριστερά προς τα δεξιά.

Παραδείγματα χρησιμοποίησης λογικών τελεστών.

Αν *Α = 3*, *Β = 5* και *Γ = -1* τότε η έκφραση

Α > 2 ΚΑΙ Β <> 5 Η Γ <= 2 θα είναι ΑΛΗΘΗΣ, ενώ η έκφραση

(Α < 3 ΚΑΙ Β <= 5) Η ΟΧΙ (Γ < 0 ΚΑΙ Α = 3) θα είναι ΨΕΥΔΗΣ.

ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΤΕΛΕΣΤΩΝ: ΑΡΙΘΜΗΤΙΚΟΙ > ΣΥΓΚΡΙΤΙΚΟΙ > ΛΟΓΙΚΟΙ

Τελεστής εκχώρησης

Ο τελεστής εκχώρησης είναι το σύμβολο <- και σημαίνει παίρνει την τιμή, δηλαδή η μεταβλητή που βρίσκεται στο αριστερό τμήμα του τελεστή εκχώρησης παίρνει την τιμή της παράστασης που βρίσκεται στο δεξί τμήμα του τελεστή εκχώρησης.

Στο αριστερό τμήμα του τελεστή εκχώρησης μπορούμε να βάλουμε αποκλειστικά και μόνο μία μεταβλητή.

Στο δεξί τμήμα του τελεστή εκχώρησης μπορούμε να βάλουμε όσο πολύπλοκη παράσταση θέλουμε.

Προσοχή! Δεν πρέπει να μπερδεύουμε τον τελεστή εκχώρησης με τον τελεστή ελέγχου =.

Συναρτήσεις

Οι μαθηματικές συναρτήσεις που έχει αποθηκευμένες η Γλώσσα είναι:

Συνάρτηση	*Λειτουργία*
ΗΜ(x)	Υπολογισμός ημιτόνου
ΣΥΝ(x)	Υπολογισμός συνημιτόνου
ΕΦ(x)	Υπολογισμός εφαπτομένης
Τ_Ρ(x)	Υπολογισμός τετραγωνικής ρίζας
ΛΟΓ(x)	Υπολογισμός φυσικού λογάριθμου (ln)
Ε(x)	Υπολογισμός του e^x
Α_Μ(x)	Ακέραιο μέρος του x
Α_Τ(x)	Απόλυτη τιμή του x

Στις τριγωνομετρικές συναρτήσεις το x πρέπει να είναι είτε ακέραιος είτε πραγματικός και το τόξο εκφράζεται σε μοίρες. Σε όλες τις γλώσσες προγραμματισμού οι τριγωνομετρικές συναρτήσεις δέχονται το τόξο σε ακτίνια. Ο τύπος μετατροπής από μοίρες σε ακτίνια είναι $r = \frac{\pi\mu^o}{180^o}$. Το αποτέλεσμα είναι πάντοτε πραγματικός αριθμός.

Στη συνάρτηση **T_P(x)** το x πρέπει να είναι ακέραιος ή πραγματικός θετικός και το αποτέλεσμα είναι πάντα πραγματικός.

Στη συνάρτηση **ΛΟΓ(x)** το x μπορεί να είναι ακέραιος ή πραγματικός μεγαλύτερος του μηδενός και το αποτέλεσμα είναι πάντα πραγματικός.

Στη συνάρτηση **Ε(x)** το x μπορεί να είναι ακέραιος ή πραγματικός και το αποτέλεσμα είναι πάντα πραγματικός.

Στη συνάρτηση **A_M(x)** το x πρέπει να είναι πραγματικός και το αποτέλεσμα είναι πάντα ακέραιος. Η συνάρτηση A_M(x) είναι η μόνη συνάρτηση που επιστρέφει ακέραιο αποτέλεσμα και είναι ο μόνος ασφαλής τρόπος για να μετατρέψουμε ένα πραγματικό σε ακέραιο.

Προσοχή! Η συνάρτηση ακέραιο μέρος δεν κάνει στρογγυλοποίηση, απλά κόβει το δεκαδικό τμήμα. Π.χ.

A_M(3.1) θα δώσει 3 **A_M**(3.9) θα δώσει επίσης 3

Το πως θα κάνουμε στρογγυλοποίηση θα το δούμε λίγο αργότερα.

Εντολές εισόδου και εξόδου

Για είσοδο χρησιμοποιούμε την εντολή **ΔΙΑΒΑΣΕ**

Δεξιά από την εντολή Διάβασε μπορούμε να βάλουμε μία ή περισσότερες μεταβλητές οι οποίες διαχωρίζονται με κόμμα. Όταν το πρόγραμμα φτάσει σε μία εντολή Διάβασε τότε σταματάει η εκτέλεση του και περιμένει από το χρήστη να δώσει μία ή περισσότερες τιμές από το πληκτρολόγιο (ανάλογα με το πόσες μεταβλητές έχουμε βάλει). Τότε καταχωρεί τις τιμές αυτές στις μεταβλητές με τη σειρά που τις έχουμε βάλει και συνεχίζει την εκτέλεση των επόμενων εντολών. Πριν από κάθε εντολή Διάβασε καλό είναι να βάζουμε πάντα ένα μήνυμα για να πληροφορούμε το χρήστη τι περιμένουμε να μας δώσει ως είσοδο.

Για έξοδο χρησιμοποιούμε τις εντολές **ΕΜΦΑΝΙΣΕ** ή **ΕΚΤΥΠΩΣΕ** ή **ΓΡΑΨΕ**

Δεξιά από την εντολή εξόδου μπορούμε να βάλουμε μία ή περισσότερες μεταβλητές ή μηνύματα μέσα σε εισαγωγικά τα οποία διαχωρίζονται με κόμμα. Όταν βάλουμε μια μεταβλητή τότε εμφανίζεται στην οθόνη του υπολογιστή το περιεχόμενο της μεταβλητής αυτής, ενώ όταν βάζουμε κάτι μέσα σε εισαγωγικά τότε θεωρείται μήνυμα και εμφανίζεται στην οθόνη του υπολογιστή όπως ακριβώς το γράψαμε.

Για παράδειγμα αν το x έχει τιμή 3, η εντολή:

ΓΡΑΨΕ 'Το x είναι ' , x

θα εμφανίσει: Το x είναι 3

ενώ η εντολή:

ΓΡΑΨΕ 'Το x είναι, x'

θα εμφανίσει: Το x είναι, x

1.10 Δομή του προγράμματος

Η βασική δομή που θα χρησιμοποιήσουμε για να γράψουμε ένα πρόγραμμα σε Γλώσσα είναι:

Αλγόριθμος <Όνομα αλγορίθμου>
<εντολές>
Τέλος <Όνομα αλγορίθμου>

ΠΡΟΓΡΑΜΜΑ <Όνομα προγράμματος>
ΣΤΑΘΕΡΕΣ
<Δηλώνουμε όλες τις σταθερές του προγράμματος>
ΜΕΤΑΒΛΗΤΕΣ
<Δηλώνουμε όλες τις μεταβλητές που παίρνουν μέρος στο πρόγραμμα καθώς και τον τύπο τους>
ΑΡΧΗ
<εντολές>
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Το τμήμα των σταθερών είναι προαιρετικό, όπως και το τμήμα με τις συναρτήσεις και τις διαδικασίες. Το τμήμα των μεταβλητών είναι υποχρεωτικό. Το όνομα που δίνουμε στο πρόγραμμα συνήθως είναι κάτι σχετικό με αυτό που κάνει το πρόγραμμα όπως και τα ονόματα των μεταβλητών είναι συνήθως σχετικά με το τι εκφράζουν. Αυτό το κάνουμε για να μπορούμε να έχουμε γρήγορα και εύκολα μια ιδέα για το τι συμβολίζει η κάθε μεταβλητή.

π.χ.

Αλγόριθμος Μέσος_όρος_τετραμήνων
Εμφάνισε "Δώσε το όνομα του μαθητή"
Διάβασε ΟΝ
Εμφάνισε "Δώσε βαθμούς τετραμήνων"
Διάβασε Α, Β
ΜΟ ← (Α + Β) / 2
Εμφάνισε ΟΝ, " έχεις Μ.Ο. ", ΜΟ
Τέλος Μέσος_όρος_τετραμήνων

ΠΡΟΓΡΑΜΜΑ Μέσος_όρος_τετραμήνων
ΜΕΤΑΒΛΗΤΕΣ
ΧΑΡΑΚΤΗΡΕΣ: ΟΝ
ΑΚΕΡΑΙΕΣ: Α, Β
ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ
ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε το όνομα του μαθητή'
ΔΙΑΒΑΣΕ ΟΝ
ΓΡΑΨΕ 'Δώσε βαθμούς τετραμήνων'
ΔΙΑΒΑΣΕ Α, Β
ΜΟ <- (Α + Β) / 2
ΓΡΑΨΕ ΟΝ, ' έχεις Μ.Ο. ', ΜΟ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Παρατηρούμε ότι στις σταθερές χρησιμοποιούμε το σύμβολο = και όχι τον τελεστή εκχώρησης. Είναι η μόνη περίπτωση που το = δεν χρησιμοποιείται ως τελεστής ελέγχου. Στις μεταβλητές πρώτα δηλώνουμε τον τύπο ή τους τύπους των μεταβλητών και μετά την άνω και κάτω τελεία γράφουμε τις μεταβλητές.
Οι εντολές μέσα στο πρόγραμμα εκτελούνται ακολουθιακά. Δηλαδή η μία μετά την άλλη με τη σειρά που γράφονται.
Μπορούμε να προσθέσουμε σχόλια μέσα στο πρόγραμμα χρησιμοποιώντας το σύμβολο ! και γράφοντας τα σχόλια μετά από αυτό. Ο ρόλος των σχολίων είναι για να μπορούμε να θυμόμαστε εύκολα γιατί χρησιμοποιήσαμε τις συγκεκριμένες εντολές και πώς σκεφτήκαμε για να λύσουμε το συγκεκριμένο πρόβλημα, όταν θα ξαναδιαβάσουμε το πρόγραμμα.

Προσοχή! Στο σχολικό βιβλίο τα προγράμματα γράφονται με όλες τις εντολές με κεφαλαία. Αυτό γίνεται για λόγους σύμβασης και για να ξεχωρίζει από το υπόλοιπο κείμενο και δεν είναι απαραίτητο. Ο μεταφραστής της Γλώσσας μπορεί να δεχτεί τις εντολές και με κεφαλαία και με μικρά, δεν κάνει δηλαδή διάκριση των πεζών και των κεφαλαίων.

Παρατηρήσεις

Όταν δηλώνουμε μία μεταβλητή τότε δεν γνωρίζουμε το περιεχόμενο της μεταβλητής αυτής μέχρι να της δώσουμε κάποια αρχική τιμή.

Για να δώσουμε αρχική τιμή ή να μεταβάλουμε την τιμή μιας μεταβλητής πρέπει οπωσδήποτε να βάλουμε τη μεταβλητή αυτή στο αριστερό τμήμα ενός τελεστή εκχώρησης ή σε μία εντολή **Διάβασε**.

Για να χρησιμοποιήσουμε μία μεταβλητή στο δεξί τμήμα ενός τελεστή εκχώρησης ή σε μια εντολή ελέγχου ή σε μια εντολή επανάληψης, ή σε μια εντολή **Γράψε** πρέπει οπωσδήποτε προηγουμένως να της δώσουμε κάποια αρχική τιμή.

1.11 Παραδείγματα - Μέθοδοι

Αντιμετάθεση των περιεχομένων δύο μεταβλητών

```
ΠΡΟΓΡΑΜΜΑ Αντιμετάθεση_γενική
ΜΕΤΑΒΛΗΤΕΣ
   ΧΑΡΑΚΤΗΡΕΣ: α, β, temp
ΑΡΧΗ
   ΔΙΑΒΑΣΕ α, β
   temp ← α
   α ← β
   β ← temp
   ΓΡΑΨΕ α, β
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
ΠΡΟΓΡΑΜΜΑ Αντιμετάθεση_αριθμών
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: α, β
ΑΡΧΗ
   ΔΙΑΒΑΣΕ α, β
   α ← α + β
   β ← α – β
   α ← α – β
   ΓΡΑΨΕ α, β
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Στρογγυλοποίηση

Να γραφεί αλγόριθμος που να διαβάζει έναν αριθμό (θετικό) και να τον στρογγυλοποιεί σε ένα δεκαδικό ψηφίο.
Για παράδειγμα: αν δώσουμε 157,32 μας δίνει 157,3 , αν δώσουμε 157,88 μας δίνει 157,9 , αν δώσουμε 60,85 μας δίνει 60,9 , αν δώσουμε 5,99 μας δίνει 6.

```
Αλγόριθμος στρογγυλοποίηση
   Διάβασε Χ
   Χ ← Χ * 10        ! πάμε την υποδιαστολή 1 θέση δεξιά (τόσες, όσα δεκαδικά θέλουμε)
   Χ ← Χ + 0.5       ! προσθέτουμε μισή μονάδα, ώστε να γίνει η στρογγυλοποίηση
   Α ← Α_Μ(Χ)        ! κόβουμε τα δεκαδικά
   Χ ← Α / 10        ! επαναφέρουμε την υποδιαστολή στη θέση της
   Εμφάνισε Χ
Τέλος στρογγυλοποίηση
```

Χρήση div και mod

Όταν διαιρούμε έναν οποιοδήποτε ακέραιο με κάποια δύναμη του 10 (10, 100, 1000 κ.ο.κ.) χρησιμοποιώντας τους τελεστές **mod** και **div**, τότε μπορούμε να φανταστούμε ότι ο αριθμός αυτός χωρίζεται από μία νοητή γραμμή σε δύο τμήματα. Τη γραμμή θα την τοποθετήσουμε τόσες θέσεις από το δεξί άκρο του αριθμού, όσα και τα μηδενικά στη δύναμη του 10. Δηλαδή:
Αν ο αριθμός 86583 διαιρεθεί με **mod** ή **div** με το **100**, τότε η νοητή γραμμή θα τοποθετηθεί **2** ψηφία από το δεξί άκρο του αριθμού, δηλαδή κάπως έτσι:

Διαίρεση με 100 (**mod** ή **div**) --> $\overbrace{\textbf{865}}^{\text{div}}\ \overbrace{\textbf{83}}^{\text{mod}}$

Ανάλογα τώρα με το ποια διαίρεση χρησιμοποιούμε έχουμε και διαφορετικό αποτέλεσμα, δηλαδή:
Η διαίρεση με το **mod** θα μας δώσει το τμήμα στα **δεξιά της νοητής γραμμής** και…
Η διαίρεση με το **div** θα μας δώσει το τμήμα **αριστερά της νοητής γραμμής**
π.χ.

865 | 83

86583 **mod** 100 = 83 και 86583 **div** 100 = 865

Κάνοντας τώρα χρήση αυτής της αρχής, μπορούμε να πάρουμε οποιοδήποτε τμήμα του αριθμού συνδυάζοντας διαιρέσεις με **mod** και **div** και αποθηκεύοντας εάν χρειαστεί, κάποια αποτελέσματα σε προσωρινές μεταβλητές.

Αντίστοιχες διαιρέσεις με το 3600, 60 μπορούν να μετατρέψουν δευτερόλεπτα σε ώρες και λεπτά αντίστοιχα (δες παράδειγμα παρακάτω).

Όπως επίσης και ακέραιες διαιρέσεις με 100, 50 κ.ο.κ. μπορούν να αναλύσουν ένα ποσό σε χαρτονομίσματα (δες παράδειγμα παρακάτω).

Διαχείριση ψηφίων ακεραίου

Αν **x** ο ακέραιος, τότε με διαδοχική χρήση των εντολών **ψηφίο_i ← x MOD 10** και **x ← x DIV 10**, στις μεταβλητές ψηφίο_i εκχωρούνται ένα προς ένα τα ψηφία του **x**, από το τελευταίο προς το πρώτο.

Αν π.χ. δοθεί ως x = 89021, τότε μετά την εντολή **x5 ← x MOD 10** θα είναι x5 = 1, δηλαδή το τελευταίο ψηφίο, ενώ μετά την **x ← x DIV 10** θα είναι x = 8902, δηλαδή το τμήμα του αριθμού που απομένει και χρειάζεται να διασπαστεί.

Να δοθεί ένας τετραψήφιος ακέραιος και να υπολογιστεί και εμφανιστεί ο κατοπτρικός του.

```
Αλγόριθμος ακέραιος
    Διάβασε x                                  ! έστω ότι δίνεται ο τετραψήφιος 1423
    x1 ← x mod 10                              ! x1 = 3
    x ← x div 10                               ! x = 142
    x2 ← x mod 10                              ! x2 = 2
    x ←x div 10                                ! x = 14
    x3 ← x mod 10                              ! x3 = 4
    x4 ← x div 10                              ! x4 = 3
    x  ← x1 * 1000 + x2 * 100 + x3 * 10 + x4   ! x = 3·1000 + 2·100 + 4·10 + 1 = 3241
    ΓΡΑΨΕ x                                    ! Εμφανίζεται ο 3241
Τέλος ακέραιος
```

Δευτερόλεπτα σε ώρες, λεπτά και δεύτερα

Να αναπτύξετε έναν αλγόριθμο (και το αντίστοιχο πρόγραμμα) ο οποίος να διαβάζει έναν ακέραιο, που παριστάνει πλήθος δευτερολέπτων να εμφανίζει σε πόσες ώρες, λεπτά και δευτερόλεπτα αντιστοιχούν.

```
Αλγόριθμος Χρόνος
    Διάβασε time
    sec ← time mod 60
    time ← time div 60
    min ← time mod 60
    hr ← time div 60
    Εμφάνισε hr, min, sec
Τέλος Χρόνος
```

```
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: hr, min, sec, time
ΑΡΧΗ
    ΔΙΑΒΑΣΕ time
    hr ← time DIV 3600
    time ← time MOD 3600
    min ← time DIV 60
    sec ← time MOD 60
    ΓΡΑΨΕ hr, min, sec
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
ΠΡΟΓΡΑΜΜΑ Χρόνος
```

Ανάλυση ποσού σε χαρτονομίσματα

Η ακέραια διαίρεση του ποσού με 50 θα δώσει ως div το πλήθος χαρτονομισμάτων των 50€ και ως mod το υπόλοιπο ποσό. Η ακέραια διαίρεση του υπόλοιπου ποσού με το 20 θα δώσει ως div το πλήθος χαρτονομισμάτων των 20€ και ως mod το νέο υπόλοιπο, κ.ο.κ.

Να γραφεί αλγόριθμος που να διαβάζει έναν ακέραιο αριθμό (ποσό σε ευρώ) και να εμφανίζει το μικρότερο πλήθος χαρτονομισμάτων των 50€, 20€ και 10€ καθώς και το μικρότερο πλήθος κερμάτων των 2€ κα 1€ στα οποία αναλύεται.
Για παράδειγμα: αν δώσουμε 132 υπολογίζει δύο χαρτονομίσματα των 50€, ένα των 20€, ένα των 10€, κανένα των 5€, ένα κέρμα των 2€ και κανένα κέρμα του 1€.

```
Αλγόριθμος χαρτονομίσματα
    Διάβασε ποσό
    των_50 ← ποσό div 50
    ποσό ← ποσό mod 50
    των_20 ← ποσό div 20
    ποσό  ← ποσό mod 20
    των_10 ← ποσό div 10
    ποσό  ← ποσό mod 10
    των_5 ← ποσό div 5
    ποσό  ← ποσό mod 5
    των_2 ← ποσό div 2
    του_1 ← ποσό
    Εμφάνισε "Χαρτονομίσματα των 50: ", των_50
    Εμφάνισε "Χαρτονομίσματα των 20: ", των_20
    Εμφάνισε "Χαρτονομίσματα των 10: ", των_10
    Εμφάνισε "Χαρτονομίσματα των 5: ", των_5
    Εμφάνισε "Κέρματα των 2: ", των_2
    Εμφάνισε "Κέρματα των 1: ", του_1
Τέλος χαρτονομίσματα
```

Ποσοστά

Ένας παντρεμένος υπάλληλος έχει έναν βασικό μισθό. Παίρνει επιπλέον 35€ επίδομα γάμου και 20€ επίδομα για κάθε παιδί. Επί του βασικού μισθού έχει κρατήσεις 20% προς το ασφαλιστικό του ταμείο. Στο ποσό που απομένει μετά την αφαίρεση των ασφαλιστικών εισφορών γίνεται παρακράτηση 11% για προκαταβολή φόρου.
Να αναπτύξετε αλγόριθμο που να ρωτάει τον βασικό μισθό και τον αριθμό παιδιών και να εμφανίζει τις συνολικές ακαθάριστες αποδοχές, τις κρατήσεις και τέλος το καθαρό ποσό που θα εισπράξει ο υπάλληλος.

```
Αλγόριθμος Αποδοχές
    Διάβασε μισθός, παιδιά
! έσοδα
    επίδομα ← 35 + παιδιά * 20
    μικτά ← μισθός + επίδομα
! έξοδα
    ασφάλεια ← μισθός * 0.2
    προΦόρου ← μικτά – ασφάλεια
    φόρος ← προΦόρου * 0.11
! εκκαθάριση
    καθαρά ← μικτά – ασφάλεια – φόρος
    Εμφάνισε "Ακαθάριστος Μισθός : ", μικτά
    Εμφάνισε "Κρατήσεις : ", ασφάλεια + φόρος
    Εμφάνισε "Καθαρός μισθός : ", καθαρά
Τέλος Αποδοχές


ΠΡΟΓΡΑΜΜΑ Αποδοχές
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: μισθός, επίδομα, μικτά, ασφάλεια, προΦόρου, φόρος, καθαρά
    ΑΚΕΡΑΙΕΣ: παιδιά
ΑΡΧΗ
    ΔΙΑΒΑΣΕ μισθός, παιδιά
!έσοδα
    επίδομα ← 35 + παιδιά *20
     μικτά ← μισθός + επίδομα
! έξοδα
    ασφάλεια ← μισθός * 0.2
    προΦόρου ← μικτά – ασφάλεια
    φόρος ← προΦόρου * 0.11
! εκκαθάριση
    καθαρά ← μικτά – ασφάλεια – φόρος
    ΓΡΑΨΕ 'Ακαθάριστος Μισθός : ', μικτά
    ΓΡΑΨΕ 'Κρατήσεις : ', ασφάλεια + φόρος
    ΓΡΑΨΕ 'Καθαρός μισθός : ', καθαρά
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Να γραφεί πρόγραμμα που θα διαβάζει το βάρος ενός ατόμου πριν ξεκινήσει μία δίαιτα και το βάρος του ίδιου ατόμου αφότου τελείωσε την δίαιτα και θα υπολογίζει την ποσοστιαία απώλεια βάρους.

```
ΠΡΟΓΡΑΜΜΑ Δίαιτα
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: πριν, μετά, διαφορά, ποσοστό
ΑΡΧΗ
   ΔΙΑΒΑΣΕ πριν, μετά
   διαφορά ← πριν – μετά              ! τα κιλά που "χάθηκαν"
   ποσοστό ← διαφορά / πριν * 100     ! "μετατροπή" σε ποσοστό
   ΓΡΑΨΕ ποσοστό, '%'
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Τιμές συνάρτησης

Να γραφεί αλγόριθμος που θα διαβάζει την τιμή της μεταβλητής x και μετά θα υπολογίζει και εκτυπώνει την τιμή της συνάρτησης: $f(x)=\frac{1+\sqrt{x^2+1}}{\ln(e^x+1)}$.

```
Αλγόριθμος συνάρτηση
   Διάβασε x
   Fx ← (1 + Τ_Ρ(x ^ 2 + 1)) / (ΛΟΓ(Ε(x) + 1))
   Εμφάνισε Fx
Τέλος συνάρτηση
```

Αναλογίες

Να γραφεί πρόγραμμα που θα διαβάζει το χαρτζιλίκι που θα μοιράσει ο παππούς στα τρία εγγόνια του, ανάλογα με την ηλικία τους, καθώς και τις ηλικίες και τα ονόματα των εγγονών, θα υπολογίζει και θα εμφανίζει το όνομα καθενός και τα χρήματα που του αναλογούν.

```
ΠΡΟΓΡΑΜΜΑ Εγγόνια
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: χαρτζιλίκι, ποσό1, ποσό2, ποσό3
   ΑΚΕΡΑΙΕΣ: ηλικία1, ηλικία2, ηλικία3, σύνολοΗλικιών
   ΧΑΡΑΚΤΗΡΕΣ: όνομα1, όνομα2, όνομα3
ΑΡΧΗ
   ΔΙΑΒΑΣΕ χαρτζιλίκι , όνομα1, ηλικία1, όνομα2, ηλικία2, όνομα3, ηλικία3
   σύνολοΗλικιών ← ηλικία1 + ηλικία2 + ηλικία3
   ποσό1 ← χαρτζιλίκι / σύνολοΗλικιών * ηλικία1
   ποσό2 ← χαρτζιλίκι / σύνολοΗλικιών * ηλικία2
   ποσό3 ← χαρτζιλίκι / σύνολοΗλικιών * ηλικία3
   ΓΡΑΨΕ όνομα1, ' παίρνει ', ποσό1, ' €'
   ΓΡΑΨΕ όνομα2, ' παίρνει ', ποσό2, ' €'
   ΓΡΑΨΕ όνομα3, ' παίρνει ', ποσό3, ' €'
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```