ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ
ΣΕ
ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

Συνοπτική Θεωρία του σχολικού βιβλίου
Αναλυτική Μεθοδολογία
Ασκήσεις
Προβλήματα πανελληνίων

Κουκούδης Βασίλης

ΚΕΦΑΛΑΙΟ 1

1.1 Η έννοια πρόβλημα

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

1.2 Κατανόηση προβλήματος

Η οποιαδήποτε προσπάθεια αντιμετώπισης ενός προβλήματος είναι καταδικασμένη σε αποτυχία αν προηγουμένως δεν έχει γίνει απόλυτα κατανοητό το πρόβλημα που τίθεται.
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 μορφή με την οποία παρουσιάζεται ένα πρόβλημα μπορεί να είναι οποιαδήποτε αρκεί να μπορεί να γίνει αντιληπτή από μία από τις πέντε ανθρώπινες αισθήσεις.

Σαφήνεια διατύπωσης

Η **κατανόηση** ενός προβλήματος **εξαρτάται** σε μεγάλο βαθμό από την **διατύπωσή** του. Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός, είτε ο γραπτός.
Ο λόγος σαν μέσο επικοινωνίας και συνεννόησης πρέπει να χαρακτηρίζεται από **σαφήνεια**. **Άστοχη χρήση ορολογίας**, **λανθασμένη σύνταξη**, είναι δύο στοιχεία που μπορούν να προκαλέσουν παρερμηνείες και παραπλανήσεις. Η παρερμηνεία είναι δυνατή ακόμα και σε περιπτώσεις όπου όλοι οι λεξικολογικοί και συντακτικοί κανόνες κρατούνται με ευλάβεια.

Σημαντικός ακόμα παράγοντας στη σωστή αντιμετώπιση ενός προβλήματος είναι η **αποσαφήνιση του χώρου στον οποίο αναφέρεται**. Η πληροφορία αυτή παρέχεται επίσης από την εκφώνηση του προβλήματος. Τα δεδομένα του προβλήματος είναι αυτά που θα μας παρέχουν αυτήν την πληροφορία.
7
Ορισμοί :
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.

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

Ορισμός :
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.

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.

Η ανάλυση αυτή του προβλήματος σε άλλα απλούστερα αναδύει παράλληλα και τη δομή του προβλήματος. 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 ψηλότερου επιπέδου.

1.4 Καθορισμός απαιτήσεων

Η σωστή επίλυση ενός προβλήματος προϋποθέτει τον επακριβή προσδιορισμό των **δεδομένων** που παρέχει το πρόβλημα.
Απαιτεί επίσης την λεπτομερειακή καταγραφή των **ζητούμενων** που αναμένονται σαν αποτελέσματα της επίλυσης του προβλήματος.

Θα πρέπει να δοθεί μεγάλη προσοχή στην ανίχνευση των δεδομένων ενός προβλήματος. Επισημαίνεται πως δεν είναι πάντοτε εύκολο να διακρίνει κάποιος τα δεδομένα. Υπάρχουν πολλές περιπτώσεις προβλημάτων όπου τα δεδομένα θα πρέπει να “ανακαλυφθούν” μέσα στα λεγόμενα του προβλήματος.

Το ίδιο προσεκτικά θα πρέπει να αποσαφηνιστούν και τα ζητούμενα του προβλήματος. Δεν είναι πάντοτε ιδιαίτερα κατανοητό τι ακριβώς ζητάει ένα πρόβλημα. Σε μια τέτοια περίπτωση θα πρέπει να θέτονται μια σειρά από ερωτήσεις με στόχο την διευκρίνηση πιθανών απορριών σχετικά με τα ζητούμενα, τον τρόπο παρουσίασής τους, το εύρος τους κ.λπ. Οι ερωτήσεις αυτές μπορούν να απευθύνονται είτε στο δημιουργό του προβλήματος, είτε στον ίδιο μας τον εαυτό αν εμείς καλούμαστε να αντιμετωπίσουμε το πρόβλημα.

Στάδια αντιμετώπισης ενός προβλήματος είναι τρία:

_ **κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος ***(καθορισμός απαιτήσεων)***
_ **ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επί μέρους απλούστερα προβλήματα ***(εύρεση καταγραφή δομής)***
_ **επίλυση**, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.

1.5 Κατηγορίες προβλημάτων

1. ***Με κριτήριο τη δυνατότητα επίλυσης*** ενός προβλήματος, διακρίνουμε τρεις κατηγορίες προβλημάτων :

_ **Επιλύσιμα**, είναι εκείνα τα προβλήματα για τα οποία η λύση τους είναι ήδη γνωστή και έχει διατυπωθεί. Επιλύσιμα μπορεί επίσης να χαρακτηριστούν και προβλήματα, των οποίων η λύση δεν έχει ακόμα διατυπωθεί, αλλά ή συνάφειά τους με άλλα ήδη επιλυμένα μας επιτρέπει να θεωρούμε σαν βέβαιη τη δυνατότητα επίλυσής τους.
_ **Ανοικτά**, ονομάζονται εκείνα τα προβλήματα για τα οποία η λύση τους δεν έχει μεν ακόμα βρεθεί, αλλά παράλληλα δεν έχει αποδειχθεί, ότι δεν επιδέχονται λύση.
_ **Άλυτα**, χαρακτηρίζονται εκείνα τα προβλήματα για τα οποία έχουμε φτάσει στην παραδοχή, ότι δεν επιδέχονται λύση.

2. ***Με κριτήριο το βαθμό δόμησης*** των λύσεών τους***, τα επιλύσιμα προβλήματα*** μπορούν να διακριθούν σε τρεις επίσης κατηγορίες :

_ **Δομημένα**, χαρακτηρίζονται εκείνα τα προβλήματα των οποίων η επίλυση προέρχεται από μια **αυτοματοποιημένη διαδικασία**.
_ **Ημιδομημένα**, ονομάζονται τα προβλήματα εκείνα των οποίων η λύση επιδιώκεται στα **πλαίσια ενός εύρους πιθανών λύσεων**, αφήνοντας στον ανθρώπινο παράγοντα περιθώρια επιλογής της.
_ **Αδόμητα**, χαρακτηρίζονται τα προβλήματα εκείνα των οποίων οι λύσεις **δεν μπορούν να δομηθούν** ή δεν έχει διερευνηθεί σε βάθος η δυνατότητα δόμησής τους. Πρωτεύοντα ρόλο στην επίλυση αυτού του τύπου προβλημάτων κατέχει **η ανθρώπινη διαίσθηση**.

3. ***Με κριτήριο το <u>είδος της επίλυσης που επιζητούν</u>***, τα προβλήματα διακρίνονται σε τρεις κατηγορίες :

_ **Απόφασης**, όπου η απόφαση που πρόκειται να ληφθεί σαν λύση του προβλήματος που τίθεται, απαντά σε ένα ερώτημα και πιθανόν αυτή η απάντηση να είναι ένα “Ναι” ή ένα “Όχι”. Αυτό που θέλουμε να διαπιστώσουμε σε ένα πρόβλημα απόφασης είναι αν υπάρχει απάντηση που ικανοποιεί τα δεδομένα που θέτονται από το πρόβλημα.
_ **Υπολογιστικά**, όπου το πρόβλημα που τίθεται απαιτεί τη διενέργεια υπολογισμών, για να μπορεί να δοθεί μία απάντηση στο πρόβλημα. Σε ένα υπολογιστικό πρόβλημα ζητάμε να βρούμε τη τιμή της απάντησης που ικανοποιεί τα δεδομένα που παρέχει το πρόβλημα.
_ **Βελτιστοποίησης**, όπου το πρόβλημα που τίθεται επιζητά το βέλτιστο αποτέλεσμα για τα συγκεκριμένα δεδομένα που διαθέτει. Σε ένα πρόβλημα βελτιστοποίησης αναζητούμε την απάντηση που ικανοποιεί κατά τον καλύτερο τρόπο τα δεδομένα που παρέχει το πρόβλημα.

1.6 Πρόβλημα και υπολογιστής

Οι λόγοι που αναθέτουμε την επίλυση ενός προβλήματος σε υπολογιστή σχετίζονται με
_ την πολυπλοκότητα των υπολογισμών,
_ την επαναληπτικότητα των διαδικασιών,
_ την ταχύτητα εκτέλεσης των πράξεων,
_ το μεγάλο πλήθος των δεδομένων.

Όσο και αν τυχόν ξαφνιάζει, **ο υπολογιστής** δεν μπορεί να **εκτελεί** παρά **μόνο τρεις λειτουργίες** :

_ **πρόσθεση**, η οποία αποτελεί τη **βασική αριθμητική πράξη**, δεδομένου ότι και οι άλλες αριθμητικές πράξεις μπορούν να αντιμετωπιστούν, σαν διαδικασίες πρόσθεσης
_ **σύγκριση**, η οποία συνιστά τη **βασική λειτουργία** για την επιτέλεση όλων των **λογικών πράξεων**,
_ **μεταφορά δεδομένων**, λειτουργία που προηγείται και έπεται της επεξεργασίας δεδομένων.

Με βάση αυτές τις τρεις λειτουργίες διεκπεραιώνει όλες τις εργασίες που του αναθέτονται και επιλύει όλα τα προβλήματα που αναλαμβάνει.

ΤΕΣΤ

Δίνονται οι παρακάτω ομάδες λέξεων. Σε κάθε μια από αυτές, να βάλεις τις λέξεις στη σωστή σειρά.

1. Επίλυση, ανάλυση, κατανόηση (αναφορά σε πρόβλημα)
2. Επεξεργασία, έλεγχος, έξοδος, είσοδος (αναφορά σε δεδομένα)

Συμπλήρωσε τα κενά με το σωστή λέξη ή λέξεις που λείπει(ουν)

3. Η επίλυση ενός προβλήματος ξεκινά από την _____του.
4. _____είναι το αποτέλεσμα επεξεργασίας δεδομένων.
5. Σημαντικός παράγοντας στην κατανόηση ενός προβλήματος είναι η _____.
6. Με τον όρο _____ προβλήματος αναφερόμαστε στα συστατικά μέρη που το αποτελούν.
7. Για να μπορέσουμε να επιλύσουμε ένα πρόβλημα θα πρέπει να γίνει ο καθορισμός _____.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

8. Πρόβλημα είναι μια οποιαδήποτε κατάσταση που πρέπει να αντιμετωπίσουμε.
9. Ο ανθρώπινος εγκέφαλος είναι ένας μηχανισμός επεξεργασίας δεδομένων.
10. Για την παραγωγή πληροφοριών απαιτούνται δεδομένα.
11. Ο υπολογιστής και το πρόβλημα είναι έννοιες αλληλένδετες.
12. Ένα πρόβλημα μπορεί να αναπαρασταθεί είτε διαγραμματικά, είτε φραστικά.

1. Να δοθεί ο ορισμός των όρων δεδομένο, επεξεργασία δεδομένων, πληροφορία.
2. _ Να αναφερθούν οι κατηγορίες των προβλημάτων.
3. _ Για ποιους λόγους αναθέτεται η επίλυση ενός προβλήματος σε υπολογιστή;
4. _ Περιγράψτε τους τρόπους περιγραφής και αναπαράστασης των προβλημάτων.

2008 ΘΕΜΑ 1 Δ
2000 ΘΕΜΑ 1 Β1
2001 ΕΠΑΝ ΘΕΜΑ 1 Γ
2006 ΕΣΠΕ ΘΕΜΑ 1 Α1 Α2
2002 ΕΣΠΕ ΘΕΜΑ 1 Α Β
2005 ΕΣΠΕ ΘΕΜΑ 1 Α

ΚΕΦΑΛΑΙΟ 2

2.1 Τι είναι αλγόριθμος

Ορισμός:
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***.

_ ***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.

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

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

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

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

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 κοκ.).

_ ***Γλωσσών Προγραμματισμού*** (programming languages). Το **είδος** της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλ, χαμηλότερου ή υψηλότερου επιπέδου) **αλλάζει τη δομή και** ηλαδή **τον αριθμό των εντολών ενός αλγορίθμου**.
Γενικά μία γλώσσα που είναι χαμηλοτέρου 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 επίλυση ενός προβλήματος.

_ ***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

Τρόποι αναπαράστασης ενός αλγορίθμου:

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

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

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 χρησιμοποιούμενα γεωμετρικά σχήματα είναι τα εξής:

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

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

_ *ορθογώνιο*, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων, και

_ *πλάγιο παραλληλόγραμμο,* που δηλώνει είσοδο ή έξοδο στοιχείων.

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

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

2.4 Βασικές συνιστώσες/εντολές ενός αλγορίθμου

Συνιστώσες αλγορίθμου ή αλγοριθμικές δομές.	Δομή ακολουθίας	Δομή επιλογής	Δομή επανάληψης
Εντολές	**<u>Δηλωτικές εντολές</u>** Αλγόριθμος Τέλος **<u>Εκτελεστέες εντολές</u>** Διάβασε Εκτύπωσε Εμφάνισε <- *(Εντολή εκχώρησης)*	**<u>Εκτελεστέες εντολές</u>** Αν…τότε Επίλεξε... Τέλος_επιλογών	**<u>Εκτελεστέες εντολές</u>** Όσο ...επανάλαβε Για...από...μέχρι αρχή_επανάληψης... μέχρις_ότου

Στοιχεία ψευδογλώσσας

Σταθερές (constands). Με τον όρο αυτό αναφερόμαστε σε προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου. Οι σταθερές διακρίνονται σε
Αριθμητικές: χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, το +, το – και το κόμμα ως δεκαδικό σημείο, π.χ. 123, +5, -1,25
Αλφαριθμητικές: σχηματίζονται από οποιουσδήποτε χαρακτήρες εντός διπλών εισαγωγικών, π.χ. “Τιμή”, “Κατάσταση αποτελεσμάτων”
Λογικές: υπάρχουν δύο, οι Αληθής και Ψευδής.

Μεταβλητές (variables). Μια μεταβλητή είναι ένα γλωσσικό αντικείμενο, που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Στη μεταβλητή εκχωρείται μια τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε **αριθμητικές**, **αλφαριθμητικές** και **λογικές**.
Για τη σύνθεση του ονόματος μιας μεταβλητής χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, οι αλφαβητικοί χαρακτήρες πεζοί και κεφαλαίοι, καθώς και ο χαρακτήρας _ (underscore) δεν μπορεί να αρχίζει με αριθμητικό χαρακτήρα και δεν μπορεί να είναι μια **δεσμευμένη λέξη**.
Π.χ. : χ , κόστος , χ2, τιμή_μονάδος, i, αλλά όχι 2x όχι Αν όχι Όσο κ.λ.π.

Τελεστές (operators). Πρόκειται για τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε αριθμητικούς, λογικούς και συγκριτικούς.
Αριθμητικοί +, -, *, /, ^, MOD, DIV
Συγκριτικοί: <=, <, =, >= , >, <>
Λογικοί: **και** (σύζευξη), **ή** (διάζευξη), **όχι** (άρνηση).

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.
Π.χ.

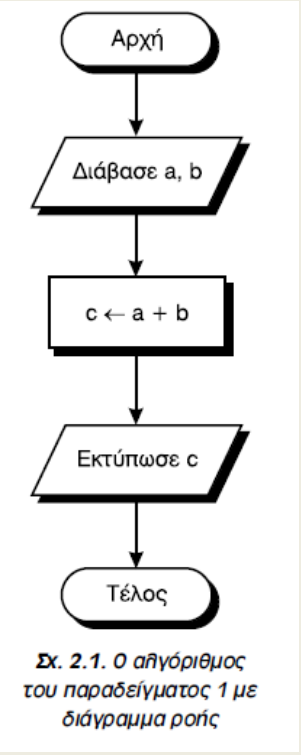
Έκφραση	***Αποτίμηση***	***Έκφραση***	***Αποτίμηση***
2	2	Χ+3	10
"Δώσε"	"Δώσε"	"Νι"+"κος"	"Νίκος"
2+3*2	8		
5>2	Αληθής		

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.

Παράδειγμα 1. Ανάγνωση και εκτύπωση αριθμών

Να διαβασθούν δύο αριθμοί, να υπολογισθεί και να εκτυπωθεί το άθροισμά τους.



Σχ. 2.1. Ο αλγόριθμος του παραδείγματος 1 με διάγραμμα ροής

```
Αλγόριθμος Παράδειγμα_1
! Προκειμένου να διαχωρίζονται οι επεξηγηματικές φράσεις από τις
! λέξεις-κλειδιά του αλγορίθμου, στις πρώτες προτάσσεται το σύμβολο !
! Εισαγωγή Δεδομένων
Εμφάνισε "Δώσε έναν αριθμό:"
Διάβασε a
Εμφάνισε "Δώσε έναν αριθμό:"
Διάβασε b
! Υπολογισμοί
c ← a + b
! Εξαγωγή αποτελεσμάτων (πληροφορία)
Εκτύπωσε "Το άθροισμα είναι :", c
Τέλος Παράδειγμα_1
```

Ένας αλγόριθμος διατυπωμένος σε ψευδογλώσσα αρχίζει πάντα με τη λέξη ***Αλγόριθμος*** συνοδευόμενη με το όνομα του αλγορίθμου και
τελειώνει με τη λέξη ***Τέλος*** συνοδευόμενη επίσης με το όνομα του αλγορίθμου. (Δηλωτικές εντολές).

Εμφάνισε: Εμφανίζει στην οθόνη την σταθερά ή μεταβλητή (τιμή), ή έκφραση (τιμή) που ακολουθεί. Μπορεί να ακολουθούν περισσότερες από μία οι οποίες χωρίζονται με κόμμα.
Π.χ.

Εντολή	Αποτέλεσμα στην οθόνη
Εμφάνισε "Δώσε έναν αριθμό:"	Δώσε έναν αριθμό:
Εμφάνισε 2	2
Εμφάνισε χ	7
Εμφάνισε "Το άθροισμα είναι :", c	Το άθροισμα είναι : 10
Εμφάνισε "Το γινόμενο είναι :", 2*3	Το γινόμενο είναι : 6

Διάβασε: Ακολουθεί μια ή περισσότερες μεταβλητές οι οποίες χωρίζονται με κόμμα. Ο αλγόριθμος περιμένει από το χρήστη να πληκτρολογήσει τιμή ή τιμές οι οποίες και δίνονται στις αντίστοιχες μεταβλητές ανεξάρτητα αν είχαν από πριν τιμή ή όχι. Ακόμη και αν είχαν τιμή, η τιμή αυτή αλλάζει και έχουν πλέον τη νέα τιμή που πληκτρολόγησε ο χρήστης.
Π.χ.

Εντολή	**Πληκτρολόγιο**	**Τιμή μεταβλητών**
Διάβασε α	2	α:2
Διάβασε α,β	2,3	α:2 β:3
Διάβασε όνομα	Κώστας	όνομα: "Κώστας"
Διάβασε όνομα,βαθμός	Κώστας,20	όνομα: "Κώστας" βαθμός: 20

Στο παράδειγμα 1 θα μπορούσαμε να γράψουμε:
Αλγόριθμος Παράδειγμα_1
Εμφάνισε "Δώσε δύο αριθμούς:"
Διάβασε a, b
…

Εναλλακτικά:
Τα δεδομένα εισόδου (αν υπάρχουν) περιγράφονται στη δεύτερη γραμμή του αλγορίθμου εντός των συμβόλων // ... //. Αντίστοιχα τα αποτελέσματα εξόδου δίνονται στην προτελευταία γραμμή του αλγορίθμου εντός των συμβόλων // ... //.

Αλγόριθμος Παράδειγμα_1
! Εισαγωγή Δεδομένων
Δεδομένα // α, β//
! Υπολογισμοί
c ← a + b
! Εξαγωγή αποτελεσμάτων (πληροφορία)
Αποτελέσματα // c //
Τέλος Παράδειγμα_1

Μετά την ανάγνωση των τιμών των μεταβλητών a και b γίνεται ο υπολογισμός του αθροίσματος με την εντολή: c←a + b. Η εντολή αυτή αποκαλείται εντολή **εκχώρησης τιμής**. Η γενική μορφή της είναι:

Μεταβλητή ←'Εκφραση

και η λειτουργία της είναι "γίνονται οι πράξεις στην έκφραση και το αποτέλεσμα αποδίδεται, μεταβιβάζεται, εκχωρείται στη μεταβλητή".

Στην εντολή αυτή χρησιμοποιείται το αριστερό βέλος, προκειμένου να δείχνει τη φορά της εκχώρησης. Ας σημειωθεί ότι δεν πρόκειται για εξίσωση, παρ'όλο που σε άλλα βιβλία μπορεί να χρησιμοποιείται το σύμβολο ίσον "=" για τον ίδιο σκοπό.

Τέλος ο αλγόριθμος ολοκληρώνεται με την εντολή **Εκτύπωσε**, που εκτυπώνει το τελικό αποτέλεσμα στον εκτυπωτή. Η σύνταξη της εντολής αυτής είναι ανάλογη με αυτή της Εμφάνισε.

Στον προηγούμενο αλγόριθμο οι μεταβλητές a και b είναι τα δεδομένα που αποτελούν την *είσοδο*, ενώ η μεταβλητή c αντιπροσωπεύει το αποτέλεσμα, δηλαδή την *έξοδο* του αλγορίθμου. Επιπλέον, ο αλγόριθμος έχει απολύτως καθορισμένη την κάθε εντολή (*καθοριστικότητα*), τελειώνει μετά από συγκεκριμένο αριθμό βημάτων (*περατότητα*), ενώ κάθε εντολή του είναι ιδιαίτερα σαφής και απλή (*αποτελεσματικότητα*). Επομένως ο αλγόριθμος αυτός πληροί τα κριτήρια που χαρακτηρίζουν τον ορισμό της έννοιας του αλγορίθμου, όπως αυτά περιγράφηκαν στην παράγραφο 2.1.

Άσκηση 1:
Να αντιστοιχήσετε τα παρακάτω:

1	2	Α	Σταθερά
2	"ΝΙΚΟΣ"	Β	Μεταβλητή
3	2*(4+6)^(1/2)	Γ	Έκφραση
4	4	Δ	Έκφραση
5	(5>4) ΚΑΙ (Χ<3)	Ε	Λογική έκφραση
6	Χ=3	Ζ	Έκφραση
7	7	Η	Σταθερά
8	ΝΙΚΟΣ	Θ	Έκφραση

Άσκηση 2:
Να αποτιμήσετε τις εκφράσεις αν x=1:
((x+3)*5+5)^(1/2)
x>2
(x+8) DIV 4
x+8 MOD 4

Άσκηση 3:
Ποιες από τις παρακάτω εντολές είναι λάθος και γιατί;

1. Διάβασε 2
2. Διάβασε x+ψ
3. Διάβασε Πέτρο
4. Διάβασε "Πέτρο"
5. 2← x
6. Εκτύπωσε "Πέτρο", Πέτρο
7. Διάβασε x,ψ
8. x+ψ←4+5
9. Πέτρο←7
10. "Πέτρο"←7
11. Εκτύπωσε x+ψ
12. x← 4>5

Άσκηση 4:
Να γράψετε τι θα εκτυπώσει ο παρακάτω αλγόριθμος:

```
Αλγόριθμος άσκηση
x←3
ψ←4
Εκτύπωσε x,ψ
x←x+1
Εκτύπωσε x,ψ
ψ←2*ψ+x
x←6
Εκτύπωσε x,ψ
Τέλος άσκηση
```

Παράδειγμα 1β:
Να γράψετε έναν αλγόριθμο ο οποίος υπολογίζει το μήκος της περιφέρειας και το εμβαδόν ενός κύκλου.

Λύση:

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: ακτίνα
Ζητούμενα: Εμβαδόν, Περιφέρεια

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Απλή δομή ακολουθίας.

- Εισαγωγή δεδομένων
- Υπολογισμοί με βάση τους τύπους : $E=\pi\rho^2$, $\Pi=2\pi\rho$
- Εκτύπωση ζητουμένων

Λύση «στο χέρι»
*Έστω ρ=1μ τότε Ε=3,14*1=3,14τ.μ. και Π=2*3,14*1=6,28μ*

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος κύκλος
! Εισαγωγή δεδομένων
π=3,14
Εμφάνισε ¨Δώσε την ακτίνα σε μέτρα¨
Διάβασε ρ
! Υπολογισμοί
Ε ← π*ρ^2
Πε← 2*π*ρ
! Εμφάνιση αποτελεσμάτων
Εμφάνισε ¨Το εμβαδόν είναι ¨, Ε , ¨τ.μ. και η περιφέρεια ¨, Πε , ¨ μ.¨
Τέλος κύκλος
```

Βήμα 4ο. Επαλήθευση

Στην επαλήθευση κάνουμε ένα πίνακα με όλες τις μεταβλητές και ακολουθώντας τα βήματα του αλγορίθμου βάζουμε τις τιμές. Όπου υπάρχει εντολή διάβασε βάζουμε δικές μας βολικές τιμές.

π	ρ	Ε	Πε
3,14	1	3,14	6,28

Άσκηση 5:

Γράψτε τον αλγόριθμο για το παρακάτω πρόβλημα και στη συνέχεια πραγματοποιήστε εικονική εκτέλεσή του έτσι ώστε να βεβαιωθείτε ότι λειτουργεί σωστά. Επίσης να κάνετε το λογικό διάγραμμα.
Δίδονται οι πλευρές ενός τριγώνου και υπολογίζεται το εμβαδόν του τριγώνου με τον τύπο του Ήρωνα $E = \text{ρίζα}(\ \tau(\tau-\alpha)(\tau-\beta)(\tau-\gamma)\ )$ όπου τ είναι η ημιπερίμετρος του τριγώνου.
Προβληματιστείτε πάνω στο ερώτημα “Μπορεί ο υπολογισμός αυτός να γίνεται για κάθε τριάδα αριθμών”. Προσπαθήστε να δικαιολογήσετε τη απάντησή σας όσο καλύτερα μπορείτε.

Άσκηση 6:

Στο παρακάτω αλγόριθμο να συμπληρώσετε τις εντολές ώστε η εντολή εκτύπωσε να εκτυπώνει 2,1 και όχι 1,2. Να τις επαληθεύσετε με πίνακα τιμών.

Αλγόριθμος ανταλλαγή
x←1
ψ←2
…….
Εκτύπωσε x,ψ
Τέλος ανταλλαγή

(Σπαζοκεφαλιά: μπορείτε να το κάνετε και με άλλο τρόπο χωρίς να χρησιμοποιήσετε $3^{η}$ μεταβλητή;)

Παράδειγμα 2:

Να γραφεί ένας αλγόριθμος που θα δέχεται έναν ακέραιο αριθμό και θα βρίσκει το τελευταίο ψηφίο του (μονάδες).

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: αριθμός
Ζητούμενα: τελευταίο ψηφίο του (μονάδες).

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*

Απλή δομή ακολουθίας.

- Εισαγωγή δεδομένων
- Υπολογισμοί με βάση τον τύπο : τψ= α MOD 10
- Εκτύπωση ζητουμένων

Λύση «στο χέρι»

Έστω α=234 τότε α MOD 10 = 4 (α DIV 10 = 23)

Βήμα 3ο. Αλγόριθμος

Αλγόριθμος τελευταίο_ψηφίο
! Εισαγωγή δεδομένων
Εμφάνισε ¨Δώσε αριθμό¨
Διάβασε α
! Υπολογισμοί
τψ← α MOD 10
! Εμφάνιση αποτελεσμάτων
Εμφάνισε ¨Το εμβαδόν τελευταίο ψηφίο είναι: " τψ
Τέλος τελευταίο_ψηφίο

Βήμα 4ο. Επαλήθευση

α	τψ
234	4

Άσκηση 7:

Να γραφεί αλγόριθμος που να δέχεται έναν τριψήφιο αριθμό και να εμφανίζει το άθροισμα των ψηφίων του.

Άσκηση 8:

Για να υπολογίσουμε τη ροή του αίματος στον ανθρώπινο οργανισμό χρησιμοποιούμε τον τύπο ροής υγρών σε σωλήνες. Για παράδειγμα, η ροή του αίματος στην αορτή (την βασική αρτηρία που μεταφέρει αίμα σε όλα τα όργανα εκτός από τους πνεύμονες) υπολογίζεται από τον τύπο $ΡΟΗ= 5500πρ^4$, όπου ρ η ακτίνα της αορτής.

Μία υγιής αορτή έχει διάμετρο περίπου 0,02m. Η μείωση της διαμέτρου (στένωση) της αορτής προκαλεί σοβαρά καρδιαγγειακά νοσήματα αφού οποιαδήποτε στένωση προκαλεί πολύ μεγάλη μείωση της ροής αίματος. Για παράδειγμα, στένωση κατά 33% της αορτής προκαλεί μείωση κατά 80% της ροής του αίματος, με πολύ σοβαρές επιπλοκές στην υγεία του ανθρώπου.

Να γράψετε αλγόριθμο που να υπολογίζει τη ροή του αίματος σε μία φυσιολογική αορτή (με ακτίνα 0.01m) και την ποσοστιαία μεταβολή της ροής που επέρχεται με μείωση της ακτίνας της αορτής κατά 10%, 33% και 50%.

2.4.2 Δομή Επιλογής

Στην πραγματικότητα πολύ λίγα προβλήματα μπορούν να επιλυθούν με τον προηγούμενο τρόπο της σειριακής/ακολουθιακής δομής ενεργειών.
Η πλέον συνηθισμένη περίπτωση είναι να λαμβάνονται κάποιες αποφάσεις με βάση κάποια δεδομένα κριτήρια, που μπορεί να είναι διαφορετικά κάθε φορά.
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 συνθήκης.

Στην παράσταση αλγορίθμων με ψευδογλώσσα η επιλογή υλοποιείται με την εντολή *Αν...τότε*. Η σύνταξη της εντολής είναι:

Αν *συνθήκη* **τότε** *εντολή*

και η λειτουργία της είναι:
Αν ισχύει η συνθήκη (δηλαδή αν είναι **αληθής**), τότε μόνο εκτελείται η εντολή. Σε κάθε περίπτωση εκτελείται στη συνέχεια η εντολή, που ακολουθεί.

Στην εντολή *Αν...τότε* είναι πιθανό, όταν ισχύει η συνθήκη, να απαιτείται η εκτέλεση **περισσότερων από μία εντολές**. Στην περίπτωση αυτή οι διαδοχικές εντολές γράφονται από κάτω και σε **εσοχή**, ενώ το σχήμα επιλογής κλείνει με τη λέξη **Τέλος_αν**. Π.χ

Αν *συνθήκη* **τότε**
    εντολή_1
    εντολή_2
    .....
    εντολή_ν
Τέλος_αν

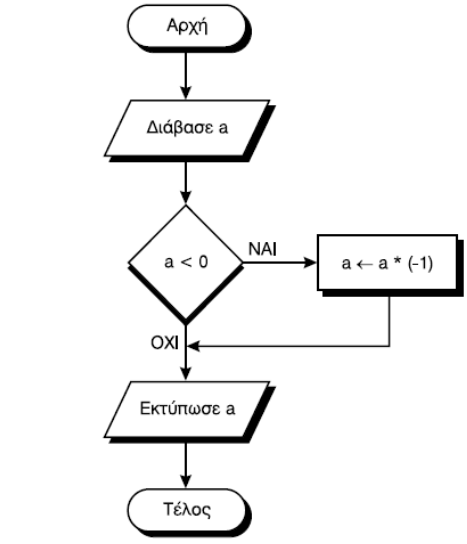
Παράδειγμα 3:
Να διαβαστεί ένας αριθμός να μετατραπεί στην απόλυτη τιμή του και να εκτυπωθεί.

Βήμα 1^{ο}. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: αριθμός
Ζητούμενα: απόλυτη τιμή του

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Δομή επιλογής

- Εισαγωγή δεδομένων
- Υπολογισμοί
  Περιπτώσεις 2 με βάση την τιμή του α.
  1. α>=0 δεν χρειάζεται να κάνω τίποτα. Αφήνω τον α όπως είναι.
  2. α<0 πρέπει να αλλάξω το πρόσημο του α. Μπορώ να το κάνω πολλαπλασιάζοντας με -1
- Εκτύπωση ζητουμένων

Λύση «στο χέρι»
Έστω α=-2 τότε α(-1) είναι η απόλυτη τιμή του α.*
Έστω α=2 τότε α είναι η απόλυτη τιμή του α.



Σχ. 2.2. Ο αλγόριθμος του παραδείγματος 2 με διάγραμμα ροής

Βήμα 3ο. Αλγόριθμος

Αλγόριθμος Παράδειγμα_3
Διάβασε a
Αν a < 0 **τότε** a ← a*(-1)
Εκτύπωσε a
Τέλος Παράδειγμα_3

Βήμα 4ο. Επαλήθευση

α		α
2		-2
		2

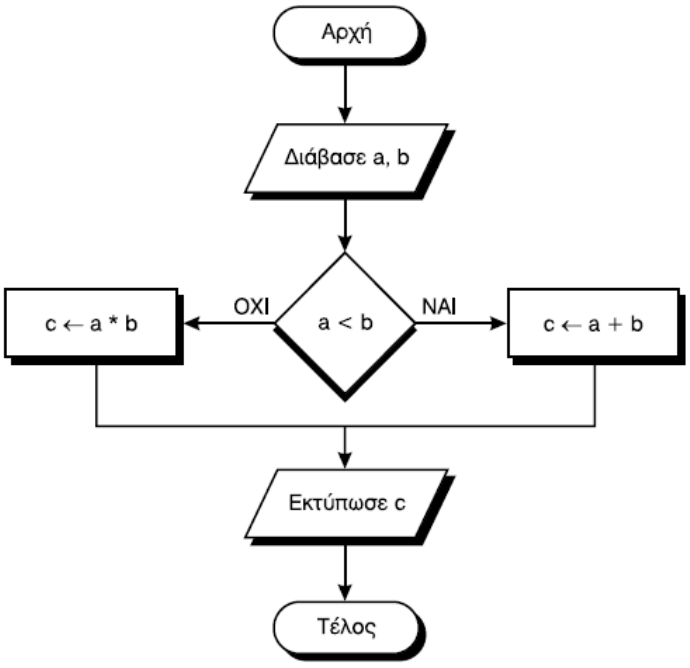
Η γενική μορφή της εντολής επιλογής, είναι:

Αν *συνθήκη* **τότε**
    εντολή ή εντολές (α)
αλλιώς
    εντολή ή εντολές (β)
Τέλος_αν

και η λειτουργία της είναι:
Αν ισχύει η συνθήκη (δηλαδή αν είναι **αληθής**), τότε εκτελείται η εντολή ή εντολές (α) αν είναι ψευδής εκτελείται η εντολή ή εντολές (β) . Σε κάθε περίπτωση εκτελείται στη συνέχεια η εντολή, που ακολουθεί.

Παράδειγμα 4
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.

```
Αλγόριθμος Παράδειγμα_4
Διάβασε a, b
Αν a < b τότε
        c ← a + b
  αλλιώς
        c ←a * b
Τέλος_αν
Εκτύπωσε c
Τέλος Παράδειγμα_4
```



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

Άσκηση 9:
Να γράψετε αλγόριθμο που θα διαβάζει 2 αριθμούς, θα βρίσκει τον μεγαλύτερο και θα τον τυπώνει.

Άσκηση 10:
Να γράψετε αλγόριθμο που θα διαβάζει έναν θετικό ακέραιο και θα τυπώνει αν είναι μονός ή ζυγός.

Άσκηση 11:
Να γράψετε αλγόριθμο που θα διαβάζει 2 αριθμούς και θα βρίσκει και θα τυπώνει το άθροισμά τους, τη διαφορά τους, το γινόμενό τους και το πηλίκο τους. Να κάνετε και το λογικό διάγραμμα.

Παράδειγμα 4β:
Να γράψετε έναν αλγόριθμο που θα διαβάζει το όνομα ενός μαθητή και τον γενικό βαθμό του στην Α Λυκείου. Στη συνέχεια θα τυπώνει αν ο μαθητής πέρασε την τάξη ή όχι.

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: όνομα, βαθμός
Ζητούμενα: πέρασε ή όχι (πρόβλημα απόφασης)

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*

Δομή επιλογής

- Εισαγωγή δεδομένων
- Υπολογισμοί
  Περιπτώσεις 2 με βάση την τιμή του βαθμού.
  1. β>=9,5 πέρασε.
  2. β<9,5 δεν πέρασε.
- Εκτύπωση ζητουμένων

Λύση «στο χέρι»

Έστω α=15 τότε πέρασε

Έστω α=8 τότε δεν πέρασε.

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος αποτελέσματα
Εμφάνισε "Δώσε το όνομα και το βαθμό"
Διάβασε ον, β
Αν β>=9,5 τότε
      αποτέλ← "πέρασε"          ( ή Εκτύπωσε ον, "πέρασε")
  αλλιώς
      αποτέλ← "δεν πέρασε"      (ή Εκτύπωσε ον, "δεν πέρασε" )
Τέλος_αν
Εκτύπωσε ον,αποτέλ              (ή εδώ τίποτα)
Τέλος αποτελέσματα
```

Βήμα 4ο. Επαλήθευση

ον	**β**	**αποτέλ**
Βασίλης	15	πέρασε

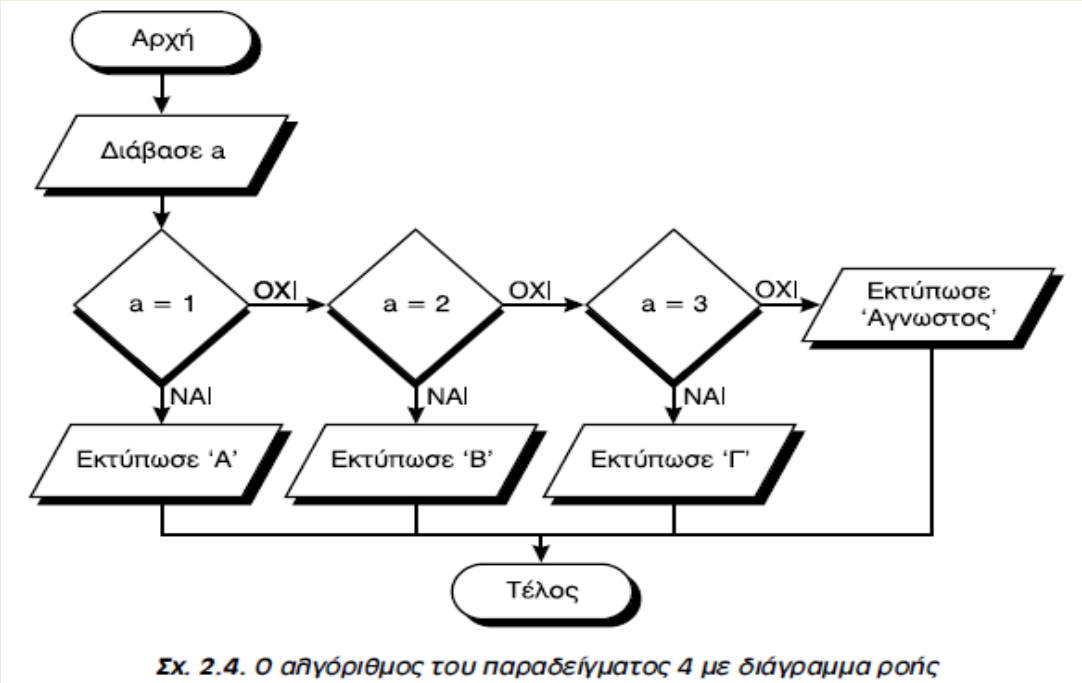
ον	**β**	**αποτέλ**
Βασίλης	8	Δεν πέρασε

Διαδικασίες πολλαπλών επιλογών

Οι διαδικασίες των πολλαπλών επιλογών εφαρμόζονται στα προβλήματα όπου μπορεί να ληφθούν περισσότερες από 2 διαφορετικές αποφάσεις ανάλογα με την τιμή που παίρνει μία έκφραση. Για παράδειγμα, κάθε γράμμα της αλφαβήτου μπορεί να αντιστοιχηθεί σε κάποιον ακέραιο αριθμό από το 1 μέχρι και 24, για τις ανάγκες κάποιας κωδικοποίησης. Στο παράδειγμα που ακολουθεί παρουσιάζεται μία περίπτωση πολλαπλών επιλογών με διαφορετική ακολουθία εντολών σε κάθε περίπτωση.

Παράδειγμα 5. Ανάθεση γραμμάτων σε αριθμούς
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_5
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_5
```



Σχ. 2.4. *Ο αλγόριθμος του παραδείγματος 4 με διάγραμμα ροής*

Άσκηση 12:
Να γράψετε αλγόριθμο ο οποίος θα διαβάζει έναν ακέραιο αριθμό και θα τυπώνει τη λέξη "θετικός", "μηδέν" ή "αρνητικός" ανάλογα με το τι είναι ο αριθμός αυτός. Να κάνετε και το λογικό διάγραμμα.

Παράδειγμα 5β:
Στο παράδειγμα 4β (πέρασε – δεν πέρασε ο μαθητής) υποθέτουμε πως ο καθηγητής δίνει μια αποδεκτή βαθμολογία δηλαδή από 0 μέχρι και 20. Τι γίνετε όμως αν ο καθηγητής κάνει λάθος και δώσει π.χ. -5 ή 25; Τι θα βγάλει το πρόγραμμα; (…) Κανονικά θα πρέπει να το ελέγχει αυτό ο αλγόριθμος και να έχει μια τρίτη επιλογή όπου θα λέει π.χ. "λάθος δεδομένα".
Άρα οι περιπτώσεις δεν είναι 2 αλλά 3. Πως όμως ορίζονται αυτές με βάση τον βαθμό β;
Περιπτώσεις:

1. β<0	:	"λάθος δεδομένα"
2. 0<=β<9,5	:	"δεν πέρασε"
3. 9,5<=β<=20	:	"πέρασε"
4. 20<β	:	"λάθος δεδομένα"

α. Ενώ φαίνονται 4 οι περιπτώσεις είναι 3 γιατί στην $1^η$ και στη $4^η$ κάνουμε την ίδια ενέργεια. (Ο αριθμός των περιπτώσεων είναι με βάση τις διαφορετικές ενέργειες).

β. Για να γράψουμε μετά από μια αν … την συνθήκη 0<=β<9,5 χρησιμοποιούμε τον λογικό τελεστή ***και***.

Σε πολλές περιπτώσεις η συνθήκη είναι αρκετά πιο “δύσκολη”, δηλαδή εμπεριέχει αποφάσεις που πιθανόν να βασίζονται σε περισσότερα από ένα κριτήρια. Ο συνδυασμός των κριτηρίων αυτών καθορίζει και τις “λογικές” πράξεις που μπορούν να γίνουν μεταξύ διαφορετικών συνθηκών. Πολύ συχνά στην καθημερινή ζωή κάποιες αποφάσεις βασίζονται σε συνδυασμούς κριτηρίων και λογικών πράξεων. Για παράδειγμα, το πρόβλημα της προετοιμασίας μας για έξοδο μπορεί να επεκταθεί ως εξής “αν βρέχει **ή** αν χιονίζει θα πάρω ομπρέλα”, είτε στην πρόταση “αν έχει ήλιο **και** αν έχει ζέστη θα πάρω καπέλο”, είτε στην πρόταση “αν **δεν** έχει ήλιο θα πάρω ομπρέλα”. Οι τρεις αυτές προτάσεις περιγράφουν και τις τρεις λογικές πράξεις που μπορεί να ισχύουν μεταξύ διαφορετικών συνθηκών. Η λογική πράξη **ή** είναι αληθής όταν οποιαδήποτε από τις δύο προτάσεις είναι αληθής. Η λογική πράξη **και** είναι αληθής όταν και οι δύο προτάσεις είναι αληθείς ενώ η λογική πράξη **όχι** (η λέξη “δεν” στο παράδειγμά μας) είναι αληθής όταν η πρόταση που την ακολουθεί είναι ψευδής. Ο επόμενος πίνακας δίνει τις τιμές των τριών αυτών λογικών πράξεων για όλους τους συνδυασμούς τιμών.

Πρόταση Α	Πρόταση Β	Α ή Β	Α και Β	όχι Α

Πρόταση Α Πρόταση Β Α ή Β Α και Β όχι Α

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

Ο αλγόριθμος όταν ελέγχει όλες τις περιπτώσεις γίνεται:

```
Αλγόριθμος αποτελέσματα
Εμφάνισε "Δώσε το όνομα και το βαθμό"
Διάβασε ον, β
Αν 0<=β και β<9,5 τότε
        αποτέλ← " δεν πέρασε"
   αλλιώς_αν 9,5<=β και β<=20 τότε
        αποτέλ← " πέρασε"
   αλλιώς_αν β<0 ή β>20 τότε
        αποτέλ← "λάθος δεδομένα"
Τέλος_αν
Εκτύπωσε ον,αποτέλ
Τέλος αποτελέσματα
```

ή επειδή δεν υπάρχει άλλη εναλλακτική:

```
Αλγόριθμος αποτελέσματα
Εμφάνισε "Δώσε το όνομα και το βαθμό"
Διάβασε ον, β
Αν 0<=β και β<9,5 τότε
        αποτέλ← " δεν πέρασε"
   αλλιώς_αν 9,5<=β και β<=20 τότε
        αποτέλ← " πέρασε"
   αλλιώς
        αποτέλ← "λάθος δεδομένα"
Τέλος_αν
Εκτύπωσε ον,αποτέλ
Τέλος αποτελέσματα
```

Άσκηση 13:
Να γράψετε έναν αλγόριθμο ο οποίος θα διαβάζει το όνομα, το επώνυμο και το βαθμό ενός μαθητή και θα τυπώνει το ονοματεπώνυμο του και τον χαρακτηρισμό "άριστα" (18,20] , "πολύ καλά" (15,18], "καλά" [13,15], "μέτρια" [9,5 , 13) και "απέτυχε" [0, 9,5).

Άσκηση 14:
Σε ένα φυτώριο υπάρχουν 3 είδη δένδρων που θα δοθούν για δενδροφύτευση. Το 1ο είδος δένδρου θα δοθεί στην περιοχή της Μακεδονίας, το 2ο στην περιοχή της Θράκης, και το 3ο είδος στην περιοχή της Πελοποννήσου. Να σχεδιασθεί το διάγραμμα ροής και να γραφεί ένας αλγόριθμος που θα διαβάζει τον αριθμό του είδους του δένδρου και θα εκτυπώνει την περιοχή στην οποία θα γίνει η δενδροφύτευση.

Άσκηση 15:
Οι υπάλληλοι μίας εταιρείας συμφώνησαν για το μήνα Δεκέμβριο να κρατηθούν από το μισθό τους δύο ποσά, ένα για την ενίσχυση του παιδικού χωριού SOS και ένα για την ενίσχυση των σκοπών της UNICEF. Ο υπολογισμός του ποσού των εισφορών εξαρτάται από τον αρχικό μισθό του κάθε υπαλλήλου και υπολογίζεται με βάση τα παρακάτω όρια μισθών :

Μισθός	Εισφορά 1	Εισφορά 2
Έως 1000Ε	5%	4%
Από 1000Ε έως και 1500Ε	6,5%	5%
Από 1500 έως και 2000Ε	7%	6%
Πάνω από 2000Ε	8%	7,5%

Να γραφεί αλγόριθμος που να δέχεται ως είσοδο το μισθό του και στη συνέχεια να υπολογίζει το ποσό των δύο εισφορών και το καθαρό ποσό που θα πάρει ο υπάλληλος.

Άσκηση 16:
Να γράψετε έναν αλγόριθμο ο οποίος θα διαβάζει έναν αριθμό από το 1 έως και το 12 ο οποίος θα αντιστοιχεί στον μήνα (από Γενάρη 1 μέχρι Δεκέμβρη 12). Στη συνέχεια θα βρίσκει και θα τυπώνει την εποχή στην οποία ανήκει ο μήνας που διάβασε.

Άσκηση 17:
Αν η μεταβλητή Α έχει την τιμή 10, η μεταβλητή Β έχει την τιμή 5 και η μεταβλητή Γ έχει την τιμή 3 ποιες από τις παρακάτω εκφράσεις είναι αληθείς και ποιες ψευδείς.
Α. ΟΧΙ (Α >Β)
Β. Α > Β ΚΑΙ Α<Γ Η Γ=<Β
Γ. Α>Β ΚΑΙ (Α<Γ Η Γ=<Β)
Δ. Α = Β Η (Γ-Β) < 0
Ε. (Α > Β ΚΑΙ Γ< Β) Η (Β <> Γ ΚΑΙ Α< Γ)

Άσκηση 18:
Να γραφεί πρόγραμμα που να υπολογίζει τις ρίζες της δευτεροβάθμιας εξίσωσης αx2+βx+γ=0. Αν δεν υπάρχουν πραγματικές ρίζες, να εκτυπώνει αντίστοιχο μήνυμα.

Άσκηση 19: Θέμα 3ο 2002
Άσκηση 20: Θέμα 1ο Γ 2004
Άσκηση 21: Θέμα 1ο Ε 2006
Άσκηση 22: Θέμα 2ο 1 2006
Άσκηση 23: Θέμα 3ο 2000
Άσκηση 24: Θέμα 1ο Δ ΕΣΠ 2006
Άσκηση 25: Θέμα 1ο Β ΕΣΠ 2000
Άσκηση 26: Θέμα 1ο Β ΕΣΠ 2005

Σε μια εντολή ***αν…αλλιώς_αν*** μόλις ο αλγόριθμος συναντήσει μια συνθήκη η οποία ικανοποιείται εκτελεί τις εντολές που ακολουθούν και στη συνέχεια φεύγει κατευθείαν μετά την τέλος_αν χωρίς καν να ελέγξει αν ικανοποιείται και μια επόμενη συνθήκη. Υπό την έννοια αυτή η παρακάτω εντολή:

```
Αν θερμοκρασία<=-5 τότε
        εκτύπωσε "παγωνιά"
   αλλιώς_αν θερμοκρασία<=5 τότε
        εκτύπωσε "πολύ κρύο"
   αλλιώς_αν θερμοκρασία<=17 τότε
        εκτύπωσε "κρύο"
   αλλιως_αν θερμοκρασία<=28 τότε
        εκτύπωσε "ζέστη"
   αλλιώς
        εκτύπωσε "καύσωνας"
Τέλος_αν
```

λειτουργεί σωστά γιατί αν π.χ. η θερμοκρασία είναι 4 θα εκτυπώσει "πολύ κρύο" το οποίο είναι και το σωστό, αλλά παρόλο που και οι υπόλοιπες συνθήκες (θερμοκρασία<=17, θερμοκρασία<=18 κλπ) ικανοποιούνται δεν θα τυπώσει τα αντίστοιχα. Βασική όμως προϋπόθεση για να λειτουργήσει σωστά είναι τα όρια των τιμών να γράφονται στη σειρά. Π.χ. αν γράφαμε:

```
Αν θερμοκρασία<=-5 τότε
        εκτύπωσε "παγωνιά"
     αλλιώς_αν θερμοκρασία<=17 τότε
        εκτύπωσε "κρύο"
   αλλιώς_αν θερμοκρασία<=5 τότε
        εκτύπωσε "πολύ κρύο"
   αλλιως_αν θερμοκρασία<=28 τότε
        εκτύπωσε "ζέστη"
   αλλιώς
        εκτύπωσε "καύσωνας"
Τέλος_αν
```

για ποιες θερμοκρασίες θα εκτύπωνε λάθος και πιο θα ήταν αυτό;

Άσκηση 27: Να ξαναγράψετε τον παραπάνω αλγόριθμο στη δεύτερη εκδοχή του με πλήρη όρια τιμών και να ελέγξετε αν κάνει το ίδιο λάθος.

Πάντως το πιο ασφαλές είναι τα όρια τιμών να γράφονται πλήρη και με την σειρά.
Το επόμενο παράδειγμα είναι πολύ σημαντικό και αναφέρετε σε μια σειρά αλγορίθμων που ονομάζονται "***κλιμακωτοί***".

Παράδειγμα 6:
Ο φόρος εισοδήματος υπολογίζεται ως εξής: Το ποσό του εισοδήματος μέχρι και 10.000Ε δεν φορολογείται καθόλου. Το ποσό από 10.000Ε μέχρι και 22.000Ε φορολογείται με 10%. Το ποσό από 22.000Ε μέχρι και 35.000Ε φορολογείται με 15%. Το ποσό πέραν των 35.000Ε φορολογείται με 25%. Να γράψετε αλγόριθμο ο οποίος θα διαβάζει το εισόδημα και θα υπολογίζει και τυπώνει τον φόρο.

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: εισόδημα (ε)
Ζητούμενα: φόρος **(πρόβλημα υπολογιστικό)**

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Δομή επιλογής, κλιμακωτός υπολογισμός

- Εισαγωγή δεδομένων
- Υπολογισμοί
  Περιπτώσεις 5 με βάση την τιμή του φόρου.
  1. ε<=0 "λάθος δεδομένα"
  2. 0<=ε<=10000 φ=0
  3. 10000<ε<=22000 φ=0+(ε-10000)*10%
  4. 22000<ε<=35000 φ=0+12000*10%+(ε-22000)*15%
  5. 35000<ε φ=0+12000*10%+13000*15%+(ε-35000)*25%
- Εκτύπωση ζητουμένων

Λύση «στο χέρι»
Έστω ε=2.000Ε τότε είναι φανερό πως δεν φορολογείται καθόλου φ=0.

Έστω ε=13.000Ε τότε δεν θα φορολογηθεί με 10% (φ=1.300Ε) για όλο το ποσό. Οι πρώτες 10.000 θα μείνουν αφορολόγητες και με 10% θα φορολογηθεί μόνο το επιπλέον δηλαδή οι 3.000Ε άρα φ=300Ε.

*Έστω ε=23.000Ε τότε οι πρώτες 10.000 θα μείνουν αφορολόγητες, οι επόμενες 12.000Ε (δηλαδή από τις 10.000 μέχρι τις 22.000) θα φορολογηθούν με 10% και το ποσό πέρα των 22.000Ε (δηλαδή οι 23.000-22000=1000Ε) θα φορολογηθούν με 15% άρα ο φόρος θα είναι φ=0+12000*10%+(23000-22000)*15%=1200+150=1350Ε.*
*Έστω ε=40.000Ε. Με παρόμοιο συλλογισμό προκύπτει πως ο φόρος : φ=0+12000*10%+13000*15%+(40000-35000)*25%=1200+1950+1250=4400*

Σ αυτού του είδους τις κλιμακωτές ασκήσεις πρέπει να ξεκαθαρίζουμε τα όρια και τις διαφορές τους κι αυτό γιατί ανάλογα με την διατύπωση μπορεί να μας δίνει μόνο τα όρια (όπως έγινε εδώ) ή μόνο τις διαφορές όπως για παράδειγμα στην ακόλουθη διατύπωση του ίδιου προβλήματος:

Ο φόρος εισοδήματος υπολογίζεται ως εξής: Το ποσό του εισοδήματος μέχρι και 10.000Ε δεν φορολογείται καθόλου. Οι επόμενες 12.000Ε φορολογούνται με 10%. Οι επόμενες 13.000Ε φορολογούνται με 15%. Και το επιπλέον ποσό φορολογείται με 25%. Να γράψετε αλγόριθμο ο οποίος θα διαβάζει το εισόδημα και θα υπολογίζει και τυπώνει τον φόρο.

Ανεξάρτητα από τη διατύπωση εμείς χρειαζόμαστε και τα όρια (τα χρησιμοποιούμε στις συνθήκες) και τις διαφορές (τις χρησιμοποιούμε στους υπολογισμούς). Οπότε ανεξάρτητα τι μας δίνει θα πρέπει να κάνουμε ένα πίνακα σαν τον παρακάτω:

Όρια	*Διαφορές*	*Ποσοστό*
0		
10000	*10000*	*0%*
22000	*12000*	*10%*
35000	*13000*	*15%*
>35000		*25%*

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος φόρος
Εμφάνισε "Δώσε εισόδημα:"
Διάβασε ε
Αν ε<0 τότε
        Εκτύπωσε "Λάθος δεδομένα"
   αλλιώς_αν 0<=ε και ε<=10000 τότε
        φ←0
        Εκτύπωσε "Ο φόρος είναι ", φ , "Ε"
    αλλιώς_αν 10000<ε και ε<=22000 τότε
        φ←0+(ε-10000)*10/100
        Εκτύπωσε "Ο φόρος είναι ", φ , "Ε"
    αλλιώς_αν 22000<ε και ε<=35000 τότε
        φ←0+12000*10/100+(ε-22000)*15/100
        Εκτύπωσε "Ο φόρος είναι ", φ , "Ε"
     αλλιώς
        φ←0+12000*10/100+13000*15/100+(ε-35000)*25/100
        Εκτύπωσε "Ο φόρος είναι ", φ , "Ε"
Τέλος_αν
 Τέλος φόρος
```

Ερώτηση: Η εντολή ***Εκτύπωσε "Ο φόρος είναι " , φ , "Ε"*** θα μπορούσε να γραφεί μια φορά μόνο μετά την Τέλος_αν όπως έγινε με την αντίστοιχη του Παραδείγματος 4β; Αν όχι γιατί; Πότε και τι πρόβλημα θα δημιουργούνταν; Πια άλλη αλλαγή στον αλγόριθμο θα μας επέτρεπε να το κάνουμε αυτό; Γενικεύοντας, πότε νομίζετε πως μια εντολή αντί να γράφεται πολλές φορές μέσα στις διάφορες αν… ή αλλιώς_αν θα μπορεί να γράφετε μια φορά στο τέλος μετά την Τέλος_αν;

Βήμα 4ο. Επαλήθευση (να γίνει σαν άσκηση. Για πόσες τιμές του ε πρέπει να γίνει η επαλήθευση;)

Άσκηση 29: Θέμα 4ο 2000

Άσκηση 30:

Ο λογαριασμός του νερού είναι τριμηνιαίος και υπολογίζεται με βάση την κατανάλωση νερού. Η αξία του νερού υπολογίζεται από τον παρακάτω πίνακα:

Κατανάλωση/μήνα σε κυβ. μέτρα	**Τιμή σε δραχμές/κυβικό**
0-5	117
5-20	178
20-27	514
27-35	720
>35	900

Στην αξία του νερού προστίθεται το πάγιο (έστω 500 δρχ), η αποχέτευση 40% της αξίας του νερού, άλλες επιβαρύνσεις 1% καθώς και το ΦΠΑ που είναι 18% στο σύνολο του λογαριασμού.

Να γραφεί αλγόριθμος που διαβάζει το ονοματεπώνυμο του καταναλωτή, τον αριθμό του μετρητή νερού την κατανάλωση (ανά τρίμηνο) και να υπολογίζει και να τυπώνει τα ποσά του λογαριασμού.

Άσκηση 31: Θέμα 3ο 2004

Αν οι διαφορετικές επιλογές είναι πολλές, τότε μπορεί να χρησιμοποιηθεί ή εντολή ***Επίλεξε...Τέλος επιλογών*** (select case).

Επίλεξε *έκφραση*
περίπτωση *λίστα_τιμών_1*
εντολές_1
περίπτωση *λίστα_τιμών_2*
εντολές_2
.....
περίπτωση_αλλιώς
εντολές_αλλιώς
Τέλος_επιλογών

Υπολογίζεται η τιμή της έκφρασης και εκτελούνται οι εντολές που ανήκουν 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. Π.χ. *1* ή *2,3,4* ή *10..100* .

Παράδειγμα 4_με_επίλεξε. Ανάθεση γραμμάτων σε αριθμούς
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_4
Διάβασε a
Επίλεξε a
  περίπτωση 1 τότε
       εκτύπωσε 'Α'
  περίπτωση 2 τότε
       εκτύπωσε 'Β'
  περίπτωση 3 τότε
       εκτύπωσε 'Γ'
  περίπτωση _αλλιώς
       εκτύπωσε 'άγνωστος'
Τέλος_επιλογών
Τέλος Παράδειγμα_4
```

Στο παράδειγμα αυτό η έκφραση είναι η απλή μεταβλητή a. Στη θέση της θα μπορούσε να είναι μια πιο πολύπλοκη έκφραση π.χ. 2*a+x. Επίσης στη λίστα τιμών έχουμε μόνο μια τιμή σε κάθε περίπτωση π.χ. στην τρίτη περίπτωση έχουμε την τιμή 3. Θα μπορούσαμε να είχαμε μια λίστα τιμών οι οποίες χωρίζονται με κόμμα π.χ. 3,4,5 ←→ (a=3 ή a=4 ή a=5). Ή θα μπορούσαμε να είχαμε μια περιοχή τιμών π.χ. 8..30 ←→ (8<=a<=30) ή συνδυασμό τους π.χ. 3,4,5,8..30 ←→ (a=3 ή a=4 ή a=5 ή 8<=a<=30).

Άσκηση 32: Θέμα 1ο Γ2 2008

Πολλές φορές σε κάποια προβλήματα είναι 2 και όχι μια, οι μεταβλητές που πρέπει να εξετάσουμε για να πάρουμε μια απόφαση. Σ αυτή τη περίπτωση βρίσκουμε όλες τις δυνατές περιπτώσεις από τους συνδυασμούς των ξεχωριστών περιπτώσεων. Το πλήθος των συνδυασμών είναι το γινόμενο του πλήθους των ξεχωριστών περιπτώσεων Π.χ.:

Παράδειγμα 7:
Να γράψετε αλγόριθμο ο οποίος σαν είσοδο θα δέχεται το φύλλο ("Α" άντρας και "Γ" γυναίκα) καθώς και το ύψος ενός ατόμου. Στη συνέχεια θα χαρακτηρίζει το άτομο με βάση το ύψος ως εξής: Ένας άνδρας χαρακτηρίζεται "ψηλός" αν είναι πάνω από 1,80μ αλλιώς χαρακτηρίζεται "κοντός". Μια γυναίκα αντίστοιχα χαρακτηρίζεται "ψηλή" αν είναι πάνω από 1,70μ αλλιώς χαρακτηρίζεται "κοντή".

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: φύλλο (φ), ύψος (υ)
Ζητούμενα: χαρακτηρισμός ύψους

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Δομή επιλογής, συνδυασμός περιπτώσεων 2x2=4

- Εισαγωγή δεδομένων
- Υπολογισμοί
  Περιπτώσεις 4 με βάση τον συνδυασμό 2(φ)x2(υ)=4.
  1. φ="Α" και υ<1.80 "κοντός"
  2. φ="Α" και υ>=1.80 "ψηλός"
  3. φ="Γ" και υ<1.70 "κοντή"
  4. φ="Γ" και υ>=1.70 "ψηλή"
- Εκτύπωση ζητουμένων

Λύση «στο χέρι» απλή.

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος ύψος
Εμφάνισε "Δώσε φύλλο και ύψος:"
Διάβασε φ,υ
Αν φ="Α" και υ<1.80 τότε
        Εκτύπωσε "κοντός"
  αλλιώς_αν φ="Α" και υ>=1.80  τότε
        Εκτύπωσε "ψηλός "
   αλλιώς_αν φ="Γ" και υ<1.70 τότε
        Εκτύπωσε "κοντή "
   αλλιώς_αν φ="Γ" και υ>=1.70 τότε
        Εκτύπωσε "ψηλή"
Τέλος_αν
 Τέλος ύψος
```

Βήμα 4ο. Επαλήθευση (να γίνει σαν άσκηση.)

Εμφωλευμένες Διαδικασίες

Πολλαπλές επιλογές, ειδικά δύο ή περισσότερων μεταβλητών μπορούν να γίνουν και με μία **εμφωλευμένη δομή**. Το επόμενο παράδειγμα είναι το 7β με εμφωλευμένες επιλογές και περιγράφει τον τρόπο με τον οποίο μία εντολή *Αν...τότε* είναι η εντολή που εκτελείται, όταν ισχύει (ή δεν) ισχύει η συνθήκη μίας άλλης εντολής *Αν...τότε*. Βέβαια η λογική αυτή μπορεί να επεκταθεί, δηλαδή να έχουμε νέα εμφωλευμένη δομή μέσα σε μία εμφωλευμένη δομή κοκ.

Παράδειγμα 7β_με_εμφωλευμένες_δομές:
Να γράψετε αλγόριθμο ο οποίος σαν είσοδο θα δέχεται το φύλλο ("Α" άντρας και "Γ" γυναίκα) καθώς και το ύψος ενός ατόμου. Στη συνέχεια θα χαρακτηρίζει το άτομο με βάση το ύψος ως εξής: Ένας άνδρας χαρακτηρίζεται "ψηλός" αν είναι πάνω από 1,80μ αλλιώς χαρακτηρίζεται "κοντός". Μια γυναίκα αντίστοιχα χαρακτηρίζεται "ψηλή" αν είναι πάνω από 1,70μ αλλιώς χαρακτηρίζεται "κοντή".

Βήμα 1^{ο}. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: φύλλο (φ), ύψος (υ)
Ζητούμενα: χαρακτηρισμός ύψους

Βήμα 2^{ο}. Ανάλυση, *Λύση «στο χέρι»*
Δομή επιλογής, εμφωλευμένες δομές επιλογής.

- Εισαγωγή δεδομένων
- Υπολογισμοί
  Περιπτώσεις 2 με βάση το φύλλο:
  1. φ="Α"
     Υποπεριπτώσεις 2 με βάση το ύψος:
     1. υ<1.80 "κοντός"
     2. υ>=1.80 "ψηλός"
  2. φ="Γ"
     Υποπεριπτώσεις 2 με βάση το ύψος:
     1. υ<1.70 "κοντή"
     2. υ>=1.70 "ψηλή
- Εκτύπωση ζητουμένων

Λύση «στο χέρι» απλή.

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος ύψος
Εμφάνισε "Δώσε φύλλο και ύψος:"
Διάβασε φ,υ
Αν φ="Α"  τότε
        Αν υ<1.80 τότε
                Εκτύπωσε "κοντός"
           αλλιώς
                Εκτύπωσε "ψηλός "
        Τέλος_αν
   αλλιώς
        Αν υ<1.70 τότε
                Εκτύπωσε "κοντή "
            αλλιώς
                Εκτύπωσε "ψηλή"
        Τέλος_αν
Τέλος_αν
 Τέλος ύψος
```

Βήμα 4ο. Επαλήθευση (να γίνει σαν άσκηση.)

Άσκηση 33:
Να γράψετε έναν αλγόριθμο ο οποίος θα λύνει εξισώσεις πρώτου βαθμού. Ο αλγόριθμος να γραφεί σε 2 εκδοχές μια χωρίς χρήση εμφωλευμένων δομών και μια με χρήση εμφωλευμένων δομών. Και για τις 2 περιπτώσεις να κάνετε και τα λογικά διαγράμματα.

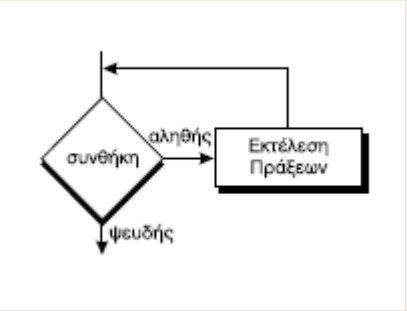
Άσκηση 34:
Να γράψετε έναν αλγόριθμο ο οποίος θα λύνει την γενική εξίσωση: $\alpha X^2+\beta X+\gamma=0$. (Προσοχή δεν είναι αναγκαστικά η δευτεροβάθμια γιατί αν ο χρήστης δώσει τιμή 0 για το α τότε είναι πρωτοβάθμια.)

Άσκηση 35: Θέμα 3ο 2003
Άσκηση 36: Θέμα 3ο ΕΠΑΝΑΛ 2001

2.4.5 Δομή Επανάληψης

Η διαδικασία της επανάληψης είναι ιδιαίτερα συχνή, αφού πλήθος προβλημάτων μπορούν να επιλυθούν με κατάλληλες επαναληπτικές διαδικασίες. Η λογική των επαναληπτικών διαδικασιών εφαρμόζεται στις περιπτώσεις, όπου μία ακολουθία εντολών πρέπει να εφαρμοσθεί σε ένα σύνολο περιπτώσεων, που έχουν κάτι κοινό.
Αυτό επιτυγχάνεται με τη χρήση της εντολής *Όσο ...επανάλαβε*. Η σύνταξη της εντολής αυτής είναι:

Εντολή/λές που δίνουν αρχικές τιμές στις μεταβλητές της συνθήκης
Όσο *συνθήκη* **επανάλαβε**
…
εντολές
…
εντολή/λες που αλλάζει την τιμή τουλάχιστον μιας μεταβλητής της συνθήκης
Τέλος_επανάληψης



Η λειτουργία της εντολής είναι η εξής: **Επαναλαμβάνεται** η εκτέλεση των εντολών, **όσο** η συνθήκη είναι **αληθής**. Όταν η συνθήκη γίνει ψευδής, τότε ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το '*Τέλος_επανάληψης*'.

Υπάρχουν 2 βασικές κατηγορίες επαναλήψεων:
- "**Γνωστού πλήθους επαναλήψεων**" και
- "**Άγνωστου πλήθους επαναλήψεων**".

Επαναλήψεις γνωστού πλήθους

Στις επαναλήψεις αυτές χρησιμοποιείται μια μεταβλητή σαν μετρητής. Πριν την *'Οσο…* ο μετρητής παίρνει μια αρχική τιμή (συνήθως το 1). Η συνθήκη ελέγχει τον μετρητή (συνήθως να είναι <= μιας τελικής τιμής). Μέσα στον ***βρόγχο*** ο μετρητής αυξάνει (συνήθως κατά 1). Η επανάληψη τερματίζεται όταν ο μετρητής φτάσει την τελική τιμή.

Γενικό σχήμα:

i ← αρχική_τιμή
Όσο i <= τελική_τιμή **επανάλαβε**
…
i ← i + βήμα
Τέλος_επανάληψης

Παράδειγμα 7.
Να γραφεί αλγόριθμος που να εμφανίζει τους αριθμούς από 1 έως 100 και τα τετράγωνά τους.

Αλγόριθμος Παράδειγμα_7
i ←1
Όσο i<=100 **επανάλαβε**
    Εμφάνισε i , i^2
    i ← i + 1
Τέλος_επανάληψης
Τέλος Παράδειγμα_7

Γνωστού πλήθους θεωρούνται και οι εξής περιπτώσεις:

Παράδειγμα 7_1.
Να γραφεί αλγόριθμος που να εμφανίζει τους άρτιους αριθμούς από 11 έως 100 και τα τετράγωνά τους.

Αλγόριθμος Παράδειγμα_7_1
i ←12
Όσο i<=100 **επανάλαβε**
    Εμφάνισε i , i^2
    i ← i + 2
Τέλος_επανάληψης
Τέλος Παράδειγμα_7_1

Πόσες φορές επαναλαμβάνετε ο βρόγχος σ αυτό το παράδειγμα;

Παράδειγμα 7_2.
Να γραφεί αλγόριθμος που να εμφανίζει τους περιττούς αριθμούς από 100 έως 20 και τα τετράγωνά τους.

Αλγόριθμος Παράδειγμα_7_2
i ←99
Όσο i>=20 **επανάλαβε**
    Εμφάνισε i , i^2
    i ← i - 2
Τέλος_επανάληψης
Τέλος Παράδειγμα_7_2

Στη περίπτωση αυτή έχουμε αντίστροφη μέτρηση, η αρχική τιμή είναι μεγαλύτερη από την τελική, το βήμα αρνητικό και στη συνθήκη ελέγχουμε αν ο μετρητής είναι >= από την τελική τιμή.
Πόσες φορές επαναλαμβάνετε ο βρόγχος σ αυτό το παράδειγμα;

Παράδειγμα 7_3.
Να γραφεί αλγόριθμος που να διαβάζει 10 αριθμούς και να εμφανίζει αυτούς και τα τετράγωνά τους.

Αλγόριθμος Παράδειγμα_7_3
i ←1
Όσο i<=10 **επανάλαβε**
    Εμφάνισε "Δώσε αριθμό:"
    Διάβασε x
    Εμφάνισε x , x^2
    i ← i + 1
Τέλος_επανάληψης
Τέλος Παράδειγμα_7_3

Όταν λέμε "**γνωστού πλήθους**" δεν εννοούμε ότι το πλήθος αυτό πρέπει σώνει και καλά να είναι γνωστό στον προγραμματιστή την ώρα που γράφει τον αλγόριθμο (πρόγραμμα), αλλά **στον χρήστη του προγράμματος την ώρα που αποφασίζει να το "τρέξει"** (λίγο πριν την έναρξη του προγράμματος). Υπό την έννοια αυτή το παρακάτω τροποποιημένο παράδειγμα θεωρείται γνωστού πλήθους.

Παράδειγμα 7_4.
Να γραφεί αλγόριθμος που να διαβάζει Ν αριθμούς και να εμφανίζει αυτούς και τα τετράγωνά τους.

Αλγόριθμος Παράδειγμα_7_3
! Στις περιπτώσεις αυτές το πρώτο πράγμα που πάντα θα δίνει ο χρήστης είναι το πλήθος
Εμφάνισε "Δώσε πλήθος:"
Διάβασε Ν
i ←1
Όσο i<=N **επανάλαβε**
    Εμφάνισε "Δώσε αριθμό:"
    Διάβασε x
    Εμφάνισε x , x^2
    i ← i + 1
Τέλος_επανάληψης
Τέλος Παράδειγμα_7_3

Άσκηση 34:
Τι θα γίνει στον Αλγόριθμο Παράδειγμα_7_3 αν ο χρήστης δώσει στο Ν τιμή Ν=-10, Ν=0, Ν=1, Ν=20 ? Τι θα γίνει αν Ν=20 αλλά ο προγραμματιστής έχει "ξεχάσει" να γράψει την εντολή i ← i + 1; Τι θα γίνει αν Ν=20 αλλά ο προγραμματιστής έχει "ξεχάσει" να γράψει την εντολή i ← 0; Τι συμβαίνει με τα αλγοριθμικά κριτήρια σε κάθε περίπτωση; Σκέψου τουλάχιστον άλλα 3 παρόμοια λάθη και εξήγησε.

Επαναλήψεις άγνωστου πλήθους

Πότε όμως το πλήθος θεωρείται άγνωστο; Όταν ο χρήστης έχει ξεκινήσει ήδη την εκτέλεση του προγράμματος χωρίς να έχει ιδέα πόσες επαναλήψεις πρέπει να γίνουν, δέχεται ροή δεδομένων, τα εισάγει στον υπολογιστή και ξαφνικά διαπιστώνει το τέλος. Για παράδειγμα ένας μαγαζάτορας θέλει στο τέλος της ημέρας να έχει μια λίστα με το ποσό που πλήρωσε ο κάθε πελάτης που ψώνισε (και το σύνολο της ημέρας αλλά αθροίσματα θα μάθουμε αργότερα). Με το που ανοίγει το μαγαζί ξεκινάει το πρόγραμμα και κάθε φορά που ψωνίζει ένας πελάτης εισάγει το ποσό στο πρόγραμμα. Είναι φυσικό να μην ξέρει πόσοι πελάτες θα ψωνίσουν μέχρι το βράδυ. Άρα ο αλγόριθμος γνωστού πλήθους επαναλήψεων του παραδείγματος_7_3 δεν μπορεί να λειτουργήσει αφού το πρώτο πράγμα που θα του ζητούσε πρωί πρωί είναι το πλήθος των πελατών που θα ψωνίσουν το υπόλοιπο της ημέρας!!! Παρόλα αυτά όταν φτάσει το βράδυ και είναι έτοιμος να κλείσει πρέπει με κάποιο τρόπο να τερματίσει τις επαναλήψεις.

Γενικό σχήμα:

Διάβασε x
Όσο x <> τιμή_φρουρός **επανάλαβε**
    …
    Διάβασε x
Τέλος_επανάληψης

Στο σχήμα αυτό η επαναλήψεις θα συνεχίζονται όσο δίνουμε στο x τιμές διαφορετικές από μια προεπιλεγμένη και προσυμφωνημένη τιμή (θα την ξέρει ο χρήστης) την οποία ονομάζουμε τιμή φρουρό. Η τιμή αυτή θα πρέπει να επιλέγετε έτσι ώστε να είναι τελείως παράλογη για την φυσική σημασία του x. Π.χ. Αν το x είναι λεφτά ή βάρος ή μήκος κ.λ.π. οποιαδήποτε αρνητική τιμή π.χ. -1 είναι κατάλληλη για φρουρός. Αν όμως είναι θερμοκρασία το -1 δεν είναι κατάλληλη τιμή φρουρός θα μπορούσαμε να βάλουμε το -500. Αν το x είναι χαρακτήρας π.χ. όνομα ή φύλο κ.λ.π. η λέξη "Τέλος" ή έστω ένα σκέτο "Τ" θα μπορούσε να επιλεχθεί σαν τιμή φρουρός. Η μόνη δυσκολία σ αυτή τη περίπτωση είναι όταν το x δεν έχει καμία φυσική σημασία, (το πρόβλημα είναι καθαρά μαθηματικό και το x καθαρός αριθμός χωρίς περιορισμούς π.χ. θετικός αριθμός). Τότε αναγκαστικά σαν τιμή φρουρό βάζουμε π.χ τον αριθμό -99999999 ελπίζοντας να μην τον χρειαστούμε ποτέ σαν πραγματικό δεδομένο. (Στη περίπτωση αυτή ο αλγόριθμός μας είναι αναγκαστικά λάθος!!!!)
Η τιμή φρουρός μπορεί να είναι και μια ολόκληρη περιοχή τιμών. Π.χ. αν το x είναι λεφτά (μη αρνητικός αριθμός) αντί να έχουμε την συνθήκη x<>-1 μπορούμε να γράψουμε x>=0.

Παράδειγμα 8: **Να γραφεί αλγόριθμος που να διαβάζει ένα άγνωστο πλήθος θετικών αριθμών και να εμφανίζει τον κάθε αριθμό και την ρίζα του.**

Αλγόριθμος Παράδειγμα_8
Διάβασε x
Όσο x > 0 **επανάλαβε**
    Εμφάνισε x, ρίζα(x)
    Διάβασε x
Τέλος_επανάληψης
Τέλος Παράδειγμα_8

Ερώτηση: γιατί εδώ η συνθήκη x>0 είναι προτιμότερη από την x<>-1 ;

Άσκηση 35:
Τι θα γίνει στον Αλγόριθμο Παράδειγμα_8 αν ο προγραμματιστής έχει "ξεχάσει" να γράψει την πρώτη Διάβασε x εντολή; Αν ο προγραμματιστής έχει "ξεχάσει" να γράψει την δεύτερη Διάβασε x εντολή; Αν ο προγραμματιστής έγραψε την δεύτερη Διάβασε x εντολή πριν την εντολή Εμφάνισε; Τι συμβαίνει με τα αλγοριθμικά κριτήρια σε κάθε περίπτωση; Σκέψου τουλάχιστον άλλα 2 παρόμοια λάθη και εξήγησε.

Ακόμα και στις περιπτώσεις άγνωστου πλήθους επαναλήψεων μπορούμε να χρησιμοποιήσουμε μετρητή, όχι για την συνθήκη τερματισμού, αλλά για να γίνει, στο τέλος, γνωστό το αρχικά άγνωστο πλήθος. Π.χ.:

Παράδειγμα 8_1: **Να γραφεί αλγόριθμος που να διαβάζει ένα άγνωστο πλήθος θετικών αριθμών και να εμφανίζει τον κάθε αριθμό και την ρίζα του. Επίσης να εμφανίζει το τελικό πλήθος των αριθμών αυτών.**

Αλγόριθμος Παράδειγμα_8_1
i ←1
Διάβασε x
Όσο x > 0 **επανάλαβε**
    Εμφάνισε x, ρίζα(x)
    i ← i + 1
    Διάβασε x
Τέλος_επανάληψης
Εμφάνισε i-1 !γιατί -1 ; Με τι τροποποίηση δεν θα χρειαζόταν το -1;
Τέλος Παράδειγμα_8_1

Πολλές φορές ο έλεγχος **της τιμής φρουρού είναι έμμεσος**. Στη συνθήκη τερματισμού ελέγχουμε μια ολόκληρη παράσταση η οποία πάντως περιέχει οπωσδήποτε την τιμή φρουρό. Π.χ.:

Παράδειγμα 8_2: **Να γραφεί αλγόριθμος που να διαβάζει ένα άγνωστο πλήθος αριθμών και να εμφανίζει τον κάθε αριθμό και το τετράγωνό του. Ο αλγόριθμος τερματίζει όταν το διπλάσιο του τετραγώνου του αριθμού γίνει μεγαλύτερο του 1000.**

Αλγόριθμος Παράδειγμα_8_2
Διάβασε x
Όσο 2* x^2<=1000 **επανάλαβε**
    Εμφάνισε x, x^2
    Διάβασε x
Τέλος_επανάληψης
Τέλος Παράδειγμα_8_2

Πολλές φορές μπορεί να υπάρχουν πολλαπλές συνθήκες τερματισμού με συνδυασμό γνωστού πλήθους με μια ή και περισσότερες άγνωστου πλήθους (οπότε και γενικά θεωρείται άγνωστο το πλήθος). Π.χ.:

Παράδειγμα 8_3: **Να γραφεί αλγόριθμος που να διαβάζει μέχρι 10 θετικούς αριθμούς και να εμφανίζει τον κάθε αριθμό και την ρίζα του. Θα τελειώνει αν ο αριθμός είναι αρνητικός ή η ρίζα του μεγαλύτερη του 1000 ή επειδή ήδη διαβάστηκαν 10 αριθμοί (όποιο από τα 3 συμβεί πρώτο). Επίσης να εμφανίζει το τελικό πλήθος των αριθμών αυτών.**

Αλγόριθμος Παράδειγμα_8_3
i ←1
Διάβασε x
Όσο x > 0 και ριζα(χ)<=1000 και i <=10 **επανάλαβε**
    Εμφάνισε x, ρίζα(x)
    i ← i + 1
    Διάβασε x
Τέλος_επανάληψης
Εμφάνισε i-1 **!** γιατί -1 ; Με τι τροποποίηση δεν θα χρειαζόταν το -1;
Τέλος Παράδειγμα_8_3

Παράδειγμα 8β:
Να γράψετε έναν αλγόριθμο που θα διαβάζει τα ονόματα των μαθητών και τον γενικό βαθμό τους στην Α Λυκείου. Στη συνέχεια θα τυπώνει το κάθε όνομα και αν ο μαθητής πέρασε την τάξη ή όχι. Τέλος θα τυπώνει το πόσοι μαθητές πέρασαν και πόσοι δεν πέρασαν.

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: όνομα, βαθμός +(εκ των υστέρων) πλήθος μαθητών
Ζητούμενα: πέρασε ή όχι , πλήθος_πέρασε, πλήθος_δεν_πέρασε

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Δομή επανάληψης με εμφωλευμένη δομή επιλογής
Δομή επανάληψης: Πρέπει να κάνω την ίδια "δουλειά" για όλους τους μαθητές. Επειδή είναι λογικό ένας καθηγητής να ξέρει το πλήθος των μαθητών του θεωρώ το πρόβλημα **γνωστού πλήθους** επαναλήψεων. (Άρα στα δεδομένα πρέπει να προσθέσω το πλήθος Ν των μαθητών.)
Για να βρω τα πλήθη πρέπει να χρησιμοποιήσω 2 άλλους μετρητές έναν που να μετρά αυτούς που πέρασαν κι έναν αυτούς που δεν πέρασαν.
Δομή επιλογής:

Περιπτώσεις 2 με βάση την τιμή του βαθμού.

1. β>=9,5 πέρασε και το πλήθος αυτών που πέρασαν αυξάνετε κατά 1.
2. β<9,5 δεν πέρασε και το πλήθος αυτών που δεν πέρασαν αυξάνετε κατά 1.

Λύση «στο χέρι»
Έστω 5 μαθητές με βαθμολογίες 12,8,15,7,19. Τρεις πέρασαν 2 έμειναν.

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος αποτελέσματα
Εμφάνισε "Δώσε το πλήθος μαθητών"
Διάβασε Ν
ππ←0
πδ←0
i ←1
Όσο i <=Ν επανάλαβε
        Εμφάνισε "Δώσε το όνομα και το βαθμό"
        Διάβασε ον, β
        Αν β>=9,5 τότε
                αποτέλ← "πέρασε"          ( ή Εκτύπωσε ον, "πέρασε")
                ππ←ππ+1
         αλλιώς
               αποτέλ← "δεν πέρασε"       (ή Εκτύπωσε ον, "δεν πέρασε" )
                πδ←πδ+1
        Τέλος_αν
        Εκτύπωσε ον,αποτέλ                 (ή εδώ τίποτα)
        i ← i + 1
Τέλος_επανάληψης
Εκτύπωσε "Πέρασαν ",ππ, "μαθητές. Δεν πέρασαν ", πδ, "μαθητές"
Τέλος αποτελέσματα
```

Βήμα 4ο. Επαλήθευση

Αριθμός επαναλήψεων	Ν	Β	ον	ππ	πδ	i	αποτέλ
Πρίν	5			0	0	1	
1η	5	12	ΚΟΣ	1	0	2	πέρασε
2η	5	8	ΝΙΚ	1	1	3	Δεν περ
3η	5	15	ΠΕΤ	2	1	4	πέρασε
4η	5	7	ΓΙΩ	2	2	5	Δεν περ
5η	5	19	ΑΝΤ	3	2	6	πέρασε
Μετά	5	19	ΑΝΤ	3	2	6	πέρασε

Παράδειγμα 8γ:
Η στατιστική υπηρεσία θέλει να μετρήσει πόσα τροχοφόρα πέρασαν κατά τη διάρκεια μιας ημέρας από τα διόδια Σχηματαρίου. Επιπλέον θέλει να γνωρίζει πόσα από αυτά ήταν δίκυκλα και πόσα τετράτροχα. Εγκατέστησε εκεί έναν υπάλληλο με έναν υπολογιστή ο οποίος κάθε φορά που έβλεπε δίκυκλο καταχωρούσε το "Δ" ενώ κάθε φορά που έβλεπε τετράτροχο το "Τ". Στο τέλος της ημέρας ο υπολογιστής τυπώνει τα ζητούμενα. Να γράψετε τον αντίστοιχο αλγόριθμο.

Βήμα 1ο. Κατανόηση, Καθορισμός απαιτήσεων (Δεδομένα, Ζητούμενα).
Δεδομένα: τύπος_τροχοφόρου
Ζητούμενα: πλήθος_τροχοφόρων, πλήθος_2κύκλων, πλήθος_4κύκλων

Βήμα 2ο. Ανάλυση, *Λύση «στο χέρι»*
Δομή επανάληψης με εμφωλευμένη δομή επιλογής
Δομή επανάληψης: Πρέπει να κάνω την ίδια "δουλειά" για όλα τα τροχοφόρα. Επειδή είναι δεν είναι λογικό κάποιος να ξέρει το πλήθος των τροχοφόρων που θα περάσουν θεωρώ το πρόβλημα **άγνωστου πλήθους** επαναλήψεων. Επειδή θα διαβάζει επαναληπτικά τον τύπο των τροχοφόρων με αποδεκτές τιμές "Δ" ή "Τ" θα χρησιμοποιήσω την τιμή "ΤΕΛΟΣ" σαν τιμή φρουρό.
Για να βρω τα πλήθη μπορώ να χρησιμοποιήσω 2 (αντί για 3) μετρητές έναν που να μετρά το σύνολο που πέρασαν κι έναν τα δίκυκλα. Το τρίτο πλήθος θα το βρω με αφαίρεση.
Δομή επιλογής:
Περιπτώσεις 1 με βάση την τιμή του τύπου.
1. τ="Δ" το πλήθος αυτών που πέρασαν αυξάνετε κατά 1.

μετά την επανάληψη
πλήθος_4κύκλων ←πλήθος_τροχοφόρων - πλήθος_2κύκλων
Λύση «στο χέρι»
Έστω "Δ" "Τ","Δ","Τ","Δ", "ΤΕΛΟΣ". ➔ *5 οχήματα, 3 2_κυκλα, 2 4_κυκλα.*

Βήμα 3ο. Αλγόριθμος

```
Αλγόριθμος τροχοφόρα
πο←0
πδ←0
Εμφάνισε "Δώσε το τύπο τροχοφόρου:"
Διάβασε ττ
Όσο ττ <>"ΤΕΛΟΣ" επανάλαβε
        πο←πο+1
        Αν ττ="Δ" τότε πδ←πδ+1
        Εμφάνισε "Δώσε το τύπο τροχοφόρου:"
        Διάβασε ττ
Τέλος_επανάληψης
Εκτύπωσε "Σύνολο:",πο, " 2_κυκλα:",πδ, " 4_κυκλα:",πο-πδ
Τέλος τροχοφόρα
```

Βήμα 4ο. Επαλήθευση

Αριθμός επαναλήψεων	πο	πδ	ττ
Πρίν	0	0	Δ
1η	1	1	Τ
2η	2	1	Δ
3η	3	2	Τ
4η	4	2	Δ
5η	5	3	ΤΕΛΟΣ
Μετά	5	3	ΤΕΛΟΣ

Άσκηση 36: Να γράψετε έναν αλγόριθμο ο οποίος για τον μήνα Ιανουάριο θα διαβάζει το όνομα της κάθε ημέρας και τη μέση θερμοκρασία της. Η ημέρα θα χαρακτηρίζεται "κρύα", "κανονική" ή "ζεστή" για θερμοκρασίες κάτω από το μηδέν, μηδέν μέχρι και 10 και πάνω από 10 αντίστοιχα. Για κάθε μέρα θα τυπώνει ημερομηνία, όνομα μέρας και χαρακτηρισμό. Επίσης θα τυπώνει το πλήθος των κρύων, κανονικών και ζεστών ημερών του μήνα.

Άσκηση 37: Να γράψετε τον αλγόριθμο της άσκησης 36 για άγνωστο πλήθος ημερών.

Άσκηση 38: Θέμα 2ο 2001
Άσκηση 39: Θέμα 2ο 2002
Άσκηση 40: Θέμα 2ο 2003
Άσκηση 41: Θέμα 3ο επαναλ 2007
Άσκηση 42: Θέμα 2ο εσπεριν 2006
Άσκηση 43: Θέμα 2ο εσπεριν 2005

Αθροίσματα

Μια σημαντική κατηγορία προβλημάτων αφορούν την εύρεση του αθροίσματος ενός (γνωστού ή άγνωστου) πλήθους αριθμών. Η φιλοσοφία του αλγορίθμου είναι ίδια μ αυτήν που χρησιμοποιούμε όταν έχουμε να βρούμε με το "μυαλό" το άθροισμα πολλών αριθμών. Ξεκινούμε έχοντας το 0 στο μυαλό μας σαν μέχρι_στιγμής_άθροισμα και κάθε φορά που διαβάζουμε ή ακούμε έναν αριθμό τον προσθέτουμε στο μέχρι_στιγμής_άθροισμα που έχουμε στο μυαλό μας. Συνεχίζουμε έτσι μέχρι να τελειώσουν όλοι οι αριθμοί και τότε το μέχρι_στιγμής_άθροισμα γίνεται το τελικό μας άθροισμα.

Με βάση τα παραπάνω το γενικό αλγοριθμικό σχήμα ενός αθροίσματος ανεξάρτητα από το γνωστό ή άγνωστο πλήθος, τον τρόπο και την εντολή επανάληψης είναι:

…

sum←0

…

Εντολή αρχής επαναληπτικής διαδικασίας

…

sum←sum+x

…

Εντολή τέλους επαναληπτικής διαδικασίας

…

όπου x είναι η μεταβλητή που κάθε φορά παίρνει την τιμή του αριθμού που πρέπει να προσθέσουμε. Όπως και με όλες τις επαναλήψεις τα αθροίσματα χωρίζονται σε δυο βασικούς τύπους γνωστού και άγνωστου πλήθους. Τα αθροίσματα άγνωστου πλήθους χωρίζονται σε δύο βασικές υποκατηγορίες αλλά και δυο υπουποκατηγορίες της μιας υποκατηγορίας!!!!

Κατηγορίες αθροισμάτων:

- **Αθροίσματα γνωστού πλήθους**
- **Αθροίσματα άγνωστου πλήθους**
  - **Με έλεγχο στον προσθετέο**
  - **Με έλεγχο σε άλλη μεταβλητή**
  - **Με έλεγχο στο άθροισμα**
    - **Να σταματά μόλις το άθροισμα φτάσει ή ξεπεράσει μια τιμή**
    - **Να σταματά πριν το άθροισμα ξεπεράσει μια τιμή**

- **Αθροίσματα γνωστού πλήθους**

```
Αλγόριθμος άθροισμα_γνωστού_πλήθους
Εμφάνισε "Δώσε το πλήθος"
Διάβασε N
i ←1                                        ! ή i←0
sum←0
Όσο i <=N επανάλαβε                         !  ή Όσο i<N επανάλαβε
      Εμφάνισε "Δώσε …"
      Διάβασε x
      sum← sum+x
      i ← i + 1
Τέλος_επανάληψης
Εκτύπωσε "Άθροισμα=",sum
Τέλος  άθροισμα_γνωστού_πλήθους
```

Άσκηση 44: Να σχεδιάσετε το λογικό διάγραμμα του παραπάνω αλγορίθμου.

Άσκηση 45: Να γράψετε αλγόριθμο που θα διαβάζει κάθε μάθημα (τίτλο μαθήματος) και τον αντίστοιχο βαθμό του ελέγχου και θα εκτυπώνει για κάθε μάθημα αύξοντα αριθμό, τίτλο μαθήματος, και βαθμό του ελέγχου ενός μαθητή και θα υπολογίζει και θα εκτυπώνει τον Μ.Ο. της βαθμολογίας του. Στο Μ.Ο. δεν προσμετράτε ο βαθμός της Γυμναστικής.
Υπόδειξη: Αν αυτό το τελευταίο με την Γυμναστική σε δυσκολεύει αρχικά αγνόησέ το! Γράψε τον αλγόριθμο σαν να μην υπήρχε, έλεγξέ τον, και στη συνέχει σκέψου, τροποποίησε τον και έλεγξέ τον ξανά.

Άσκηση 46: Να γράψετε αλγόριθμο που θα διαβάζει (με τυχαία ανάμικτη σειρά) το όνομα, το φύλο (Α,Κ) και τον τελικό βαθμό στην Α Λυκείου των μαθητών ενός τμήματος. Θα τυπώνει λίστα με αύξοντα αριθμό, όνομα και βαθμό και θα υπολογίσει και θα τυπώνει τον Μ.Ο. βαθμολογίας όλων των μαθητών καθώς και το Μ.Ο. βαθμολογίας των αγοριών και των κοριτσιών και μήνυμα ποιοι έχουν καλύτερες επιδόσεις τα αγόρια ή τα κορίτσια;

- **Αθροίσματα άγνωστου πλήθους**
  - **Με έλεγχο στον προσθετέο**

```
Αλγόριθμος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στον_προσθετέο
Εμφάνισε "Δώσε …"
Διάβασε x
sum←0
Όσο x <>τιμή_φρουρός επανάλαβε  !π.χ. x<>-999999 ή x>=0 ή x<999999
        sum←sum+x
        Εμφάνισε "Δώσε …"
        Διάβασε x
Τέλος_επανάληψης
Εκτύπωσε " Άθροισμα=",sum
Τέλος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στον_προσθετέο
```

Άσκηση 47: Να σχεδιάσετε το λογικό διάγραμμα του παραπάνω αλγορίθμου.

Άσκηση 48: Στην είσοδο ενός τελωνείου υπάρχει μια πλαστιγγογέφυρα η οποία μετρά το βάρος των φορτηγών που μπαίνουν μέσα μεταφέροντας εμπορεύματα. Η τιμή αυτή μεταβιβάζεται αυτόματα σ έναν υπολογιστή. να γράψετε αλγόριθμο που θα υπολογίζει το συνολικό βάρος όλων των φορτηγών, καθώς και τον μέσο όρο βάρους των φορτηγών που μπαίνουν στο τελωνείο κατά τη διάρκεια ενός 24ωρου. Ο αλγόριθμος θα τερματίζει όταν ο χρήστης κάποια στιγμή εισάγει (απ το πληκτρολόγιο) μια κατάλληλη τιμή φρουρό.
Υπόδειξη: Προσοχή στον Μ.Ο. στη περίπτωση που δεν πέρασε κανένα φορτηγό.

- **Αθροίσματα άγνωστου πλήθους**
  - **Με έλεγχο σε άλλη μεταβλητή**

```
Αλγόριθμος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_σε_άλλη_μεταβλητή
sum←0
Εμφάνισε "Δώσε …"
Διάβασε μ
Όσο μ <>τιμή_φρουρός επανάλαβε  !π.χ. μ<>"Τ"
        Διάβασε χ
        sum←sum+x
        Εμφάνισε "Δώσε …"
        Διάβασε μ
Τέλος_επανάληψης
Εκτύπωσε " Άθροισμα=",sum
Τέλος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_σε_άλλη_μεταβλητή
```

Άσκηση 49: Να σχεδιάσετε το λογικό διάγραμμα του παραπάνω αλγορίθμου.

Άσκηση 50: Να γράψετε αλγόριθμο ο οποίος θα διαβάζει τα ονόματα δωρητών και το ποσό που δώρισαν σε μια φιλανθρωπική οργάνωση κατά τη διάρκεια ενός μήνα και θα τυπώνει λίστα με τα ονόματα και το ποσό καθώς και το συνολικό ποσό που συγκεντρώθηκε και το Μ.Ο. του ποσού.

- **Αθροίσματα άγνωστου πλήθους**
  - **Με έλεγχο στο άθροισμα**
    - **Να σταματά μόλις το άθροισμα ξεπεράσει μια τιμή**

```
Αλγόριθμος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στο_άθροισμα
            ! Να σταματά μόλις το άθροισμα φτάσει ή ξεπεράσει μια τιμή
sum←0
Εμφάνισε "Δώσε …"
Όσο sum < τιμή_φρουρός επανάλαβε  ! sum<80
      Εμφάνισε "Δώσε …"
      Διάβασε x
      sum←sum+x
Τέλος_επανάληψης
Εκτύπωσε " Άθροισμα=",sum
Τέλος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στο_άθροισμα
```

- **Αθροίσματα άγνωστου πλήθους**
  - **Με έλεγχο στο άθροισμα**
    - **Να σταματά πριν το άθροισμα ξεπεράσει μια τιμή**

```
Αλγόριθμος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στο_άθροισμα
            ! Να σταματά πριν το άθροισμα  ξεπεράσει μια τιμή
sum←0
Εμφάνισε "Δώσε …"
Διάβασε x
Όσο sum+x < τιμή_φρουρός επανάλαβε  ! sum+x<80
      sum←sum+x
      Εμφάνισε "Δώσε …"
      Διάβασε x
Τέλος_επανάληψης
sum←sum-x
Εκτύπωσε " Άθροισμα=",sum
Τέλος Άθροισμα_άγνωστου_πλήθους_με_έλεγχο_στο_άθροισμα
```

Άσκηση 51: Να σχεδιάσετε το λογικό διάγραμμα του παραπάνω αλγορίθμου.

Άσκηση 52: Ο Κώστας πήγε στο Μασούτη με 50^{E} στη τσέπη και μια λίστα με ψώνια. Να γράψετε αλγόριθμο που θα διαβάζει τις τιμές των προϊόντων που βάζει στο καλάθι, θα σταματά πριν συμπληρωθούν τα 50^{E} και θα τυπώνει το πλήθος των προϊόντων που αγόρασε καθώς και πόσα Ε περίσσεψαν.

Άσκηση 52α: Θέμα 4ο 2001
Άσκηση 52β: Θέμα 3ο 2007

Άσκηση 53: Θέμα 3ο εσπεριν επαναλ 2005

Άσκηση 54: Ένας καταναλωτής πηγαίνει στο πολυκατάστημα και έχει στη τσέπη του 5.000 ευρώ. Ξεκινά να αγοράζει διάφορα είδη και ταυτόχρονα κρατά το συνολικό ποσό στο οποίο έχει φθάσει κάθε στιγμή που αγοράζει κάποιο είδος. Οι τιμές των ειδών που αγοράζει είναι σε δραχμές και είναι δεδομένο ότι 1 ευρώ=330 δραχμές. Να γραφεί σε φυσική γλώσσα, με ακολουθία βημάτων και με διάγραμμα ροής ένας αλγόριθμος για τον υπολογισμό του ποσού από τα ψώνια που έγιναν και να σταματά η αγορά ειδών έτσι ώστε να μην ξεπεραστεί το ποσό που έχει διαθέσιμο ο καταναλωτής.

Άσκηση 55: 'Ο Κώστας έχει μόνο 50Ε αλλά θέλει να αγοράσει έναν υπολογιστή αξίας 1000^{E}. Πήγε σ ένα κατάστημα κι έκανε την εξής συμφωνία: Θα έπαιρνε σήμερα τον υπολογιστή με τα 50^{E} που είχε και στο τέλος κάθε μήνα θα έδινε όσο χαρτζιλίκι που του περίσσευε για την αποπληρωμή του υπολογιστή. Όμως για κάθε μήνα, στην αξία του υπολογιστή θα προσθέτονταν τόκος 2% στο υπόλοιπο ποσό που απέμενε απλήρωτο. Να γράψετε αλγόριθμο που θα υπολογίζει σε πόσους μήνες ξεχρέωσε, τι ποσό τελικά πλήρωσε και πόσο χαρτζιλίκι από τον τελευταίο μήνα του έμεινε αφού πλήρωσε και την τελευταία δόση.
Υπόδειξη: Αν σε δυσκολεύει ο τόκος, αρχικά κάνε τον αλγόριθμο όπως θα ήταν αν η τιμή έμενε σταθερή στα 1000^{E}. Στη συνέχεια τροποποίησε τον αλγόριθμο πως θα γινόταν αν αντί για τόκο είχαμε σταθερή προσαύξηση της τιμής του υπολογιστή κατά π.χ. 5^{E} τον μήνα. Τέλος κάνε την τελική τροποποίηση όπως ακριβώς το ζητά η άσκηση. Σε κάθε στάδιο να βάζεις δικές σου εικονικές τιμές να υπολογίζεις "με το χέρι" και να ελέγχεις τον αλγόριθμό σου αν λειτουργεί σωστά.

Άσκηση 56: Ένα ασανσέρ χωράει μέγιστο πλήθος ατόμων (ανεξαρτήτως του συνολικού βάρους των) 10 και σηκώνει μέγιστο βάρος (ανεξαρτήτως πλήθους ατόμων) 800 κιλά. Να γράψετε αλγόριθμο που θα διαβάζει το βάρος του κάθε ατόμου που μπαίνει μέσα θα μετρά το πλήθος τους και θα τελειώνει επιτρέποντας τόσα άτομα και τόσο συνολικό βάρος ώστε να είναι μέσα στα όρια τον προδιαγραφών του. Στο τέλος θα τυπώνει το πλήθος των ατόμων που μπήκαν και το συνολικό τους βάρος.

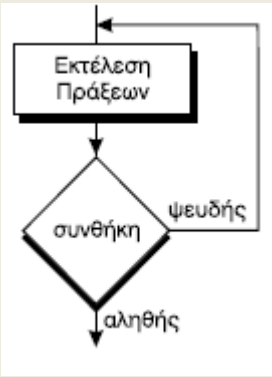
Άσκηση 57: Ο αλγόριθμος της άσκησης 56 έχει μια ατέλεια: Το ασανσέρ κάθεται και περιμένει μέχρι να γεμίσει ή να φτάσει τα όρια του βάρους του. Αν όμως υπάρχει π.χ. ένα μόνο άτομο και δεν αναμένονται άλλα, ο αλγόριθμος πρέπει να τελειώσει και το ασανσέρ να φύγει. Γι αυτό να τον τροποποιήσετε έτσι ώστε να τελειώνει αν διαβάζει αρνητικό βάρος επίσης να τελειώνει. Να κάνετε και μια δεύτερη τροποποίηση στην οποία δεν θα χρησιμοποιεί το αρνητικό βάρος αλλά κάθε φορά θα διαβάζει και μια άλλη μεταβλητή η οποία αν έχει πάρει την τιμή "Τ" αυτό σημαίνει ότι δεν υπάρχουν άλλα άτομα και το ασανσέρ φεύγει.

Εντολή αρχή_επανάληψης…μέχρις_όστου

Είναι μια άλλη εντολή επανάληψης διαδικασιών. Σύνταξη:

Εντολή/λές που δίνουν αρχικές τιμές στις μεταβλητές της συνθήκης
δεν είναι πάντα απαραίτητες
Αρχή_επανάληψης
…
…
εντολή/λες που αλλάζει την τιμή τουλάχιστον μιας μεταβλητής της συνθήκης

μέχρις_όστου *συνθήκη*



Η λειτουργία της εντολής είναι η εξής: **Επαναλαμβάνεται** η εκτέλεση των εντολών, **όσο** η συνθήκη είναι **ψευδής**. Όταν η συνθήκη γίνει αληθής, τότε ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το '*μέχρις_όστου'*.

Ας σημειωθεί ότι, στην εντολή αυτή ο βρόχος επανάληψης **θα εκτελεσθεί οπωσδήποτε τουλάχιστον μία φορά**. Η βασική διαφοροποίηση αυτής της μορφής επαναληπτικής διαδικασίας σε σχέση με την επαναληπτική διαδικασία της εντολής *'Όσο …*', οφείλεται στη θέση της λογικής συνθήκης στη ροή εκτέλεσης των εντολών.

Επαναλήψεις γνωστού πλήθους με αρχή_επανάληψης…

Γενικό σχήμα:

```
i ← αρχική_τιμή
Όσο i <= τελική_τιμή επανάλαβε
      …
      i ← i + βήμα
Τέλος_επανάληψης
```

Γενικό σχήμα:

```
i ← αρχική_τιμή
Αρχή_επανάληψης
      …
      i ← i + βήμα
μέχρις_όστου i > τελική_τιμή
```

<u>Σημαντική παρατήρηση</u>: Όταν μετατρέπουμε μια *Όσο…* σε *Αρχή_επανάληψης …* η συνθήκη γίνεται ακριβώς η **αντίθετή** της.

Π.χ. Συνθήκη της *Όσο…* : x<5 ΚΑΙ y>=8 Ή z=10
Συνθήκη της *Αρχή_επανάληψης* : x>=5 Ή y<8 ΚΑΙ z=10

Επαναλήψεις άγνωστου πλήθους με αρχή_επανάληψης…

Γενικό σχήμα:

Διάβασε x
Όσο x <> τιμή_φρουρός **επανάλαβε**
    …
    Διάβασε x
Τέλος_επανάληψης

Γενικό σχήμα:

Αρχή_επανάληψης
    Διάβασε x
    …
    …
μέχρις_ότου x = τιμή_φρουρός

Το πρόβλημα που πολλές φορές υπάρχει με την Αρχή_επανάληψης είναι ότι θα εκτελεστούν οπωσδήποτε οι εντολές ακόμα και για την *τιμή_φρουρό* για την οποία υποτίθεται δεν θα έπρεπε. Ακόμα κι αν κάνουμε την παρακάτω τροποποίηση:

Γενικό σχήμα:

Διάβασε x
Αρχή_επανάληψης
    …
    …
    Διάβασε x
μέχρις_ότου x = τιμή_φρουρός

το πρόβλημα θα εξακολουθεί να ισχύει αν η *τιμή_φρουρός* είναι η **πρώτη** που θα δώσουμε.
Το πρόβλημα αυτό λύνετε ως εξής:

Γενικό σχήμα:

Διάβασε x
Αν x<>τιμή_φρουρός **τότε**
    Αρχή_επανάληψης
        …
        …
        Διάβασε x
    μέχρις_ότου x = τιμή_φρουρός
Τέλος_αν

Γενικά αν δεν μας το ζητά η άσκηση προτιμούμε την *Όσο…* αντί για την *Αρχή_επανάληψης…*

Άσκηση 58: Να μετατρέψετε τους 5 βασικούς αλγορίθμους αθροισμάτων με χρήση της Αρχή_επανάληψης.

Έλεγχος ορθότητας εισαγωγής δεδομένων

Αν και γενικά είναι δύσχρηστη η *Αρχή_επανάληψης…* , σε μερικές περιπτώσεις είναι πολύ βολική. Μια από αυτές είναι ο έλεγχος ορθότητας των δεδομένων που δίνει ο χρήστης. Αν ο χρήστης δώσει μια παράλογη τιμή, έξω από τα αποδεκτά όρια, θέλουμε το πρόγραμμα να μην προχωρά και να ζητά ξανά και ξανά τιμή μέχρις ότου αυτή να βρίσκεται μέσα στα αποδεκτά όρια.

Γενικό σχήμα:

```
Αρχή_επανάληψης
        Εμφάνισε "Δώσε…"
        Διάβασε x
μέχρις_ότου x εντός ορίων
```

Παράδειγμα 9:
Έστω ότι τα δεδομένα σ ένα πρόγραμμα είναι: Όνομα, φύλο ("Α", "Κ"), βάρος (>0), βαθμός (>0 και <=20). Επιπλέον επειδή το φύλλο θα χρησιμοποιηθεί σαν τιμή_φρουρός αποδεκτή τιμή είναι και το "Τ" που δηλώνει το τέλος επαναλήψεων.
Σε ένα πρόγραμμα χωρίς έλεγχο ορθότητας δεδομένων θα εμφανίζονταν μια η περισσότερες φορές οι παρακάτω εντολές:

```
Εμφάνισε "Δώσε όνομα, φύλο, βάρος και βαθμό:"
Διάβασε ο,φ,β,βθ
```

με την παραπάνω μορφή ή και ξεχωριστά:

```
Εμφάνισε "Δώσε όνομα:"
Διάβασε ο
Εμφάνισε "Δώσε  φύλο:"
Διάβασε φ
Εμφάνισε "Δώσε  βάρος:"
Διάβασε β
Εμφάνισε "Δώσε βαθμό:"
Διάβασε βθ
```

Αν θέλουμε να κάνουμε έλεγχο ορθότητας δεδομένων, κάθε φορά που εμφανίζονται στο πρόγραμμα οι παραπάνω εντολές ενεργούμε ως εξής:

Σαν πρώτο βήμα χωρίζουμε τις εντολές δηλαδή **ποτέ δεν διαβάζουμε περισσότερες από μια μεταβλητές στην ίδια εντολή** *Διάβασε*.
Στη συνέχεια αντικαθιστούμε κάθε απλό ζεύγος Εμφάνισε-Διάβασε με το γενικό σχήμα που γράψαμε παραπάνω.

Άρα λοιπόν στο παράδειγμά μας θα έχουμε:

```
! στο όνομα δεν μπορούμε να κάνουμε έλεγχο
Εμφάνισε "Δώσε όνομα:"
Διάβασε ο

Αρχή_επανάληψης
Εμφάνισε "Δώσε  φύλο:"
Διάβασε φ
μέχρις_ότου φ="Α" ή φ="Κ" ή φ="Τ"

Αρχή_επανάληψης
Εμφάνισε "Δώσε  βάρος:"
Διάβασε β
μέχρις_ότου β>0

Αρχή_επανάληψης
Εμφάνισε "Δώσε βαθμό:"
Διάβασε βθ
μέχρις_ότου 0<βθ και 20<=βθ
```

Άσκηση 59: Να μετατρέψετε τους παραπάνω ελέγχους χρησιμοποιώντας την Όσο…

Άσκηση 60 ΘΕΜΑ 3^{ο} εσπερινΕΠΑΝΑΛ 2006
Άσκηση 61 ΘΕΜΑ 1^{ο} Δ εσπερινΕΠΑΝΑΛ 2002

Επαναληπτική εντολή: για...από...μέχρι...βήμα

Όταν ο αριθμός των φορών που θα εκτελεστεί μια επαναληπτική διαδικασία είναι γνωστός εκ των προτέρων, τότε είναι προτιμότερο να χρησιμοποιείται η εντολή Για...από...μέχρι.
Η σύνταξη της εντολής αυτής είναι:

```
…
Για μεταβλητή από α_τ μέχρι τ_τ με βήμα β
        …
        εντολές    ! απαγορεύεται εντολή που αλλάζει την τιμή της μεταβλητής
        …
Τέλος_επανάληψης
```

Η λειτουργία της εντολής φαίνεται καλύτερα από τον παρακάτω πίνακα όπου γίνετε αντιστοίχιση με την *Όσο…*

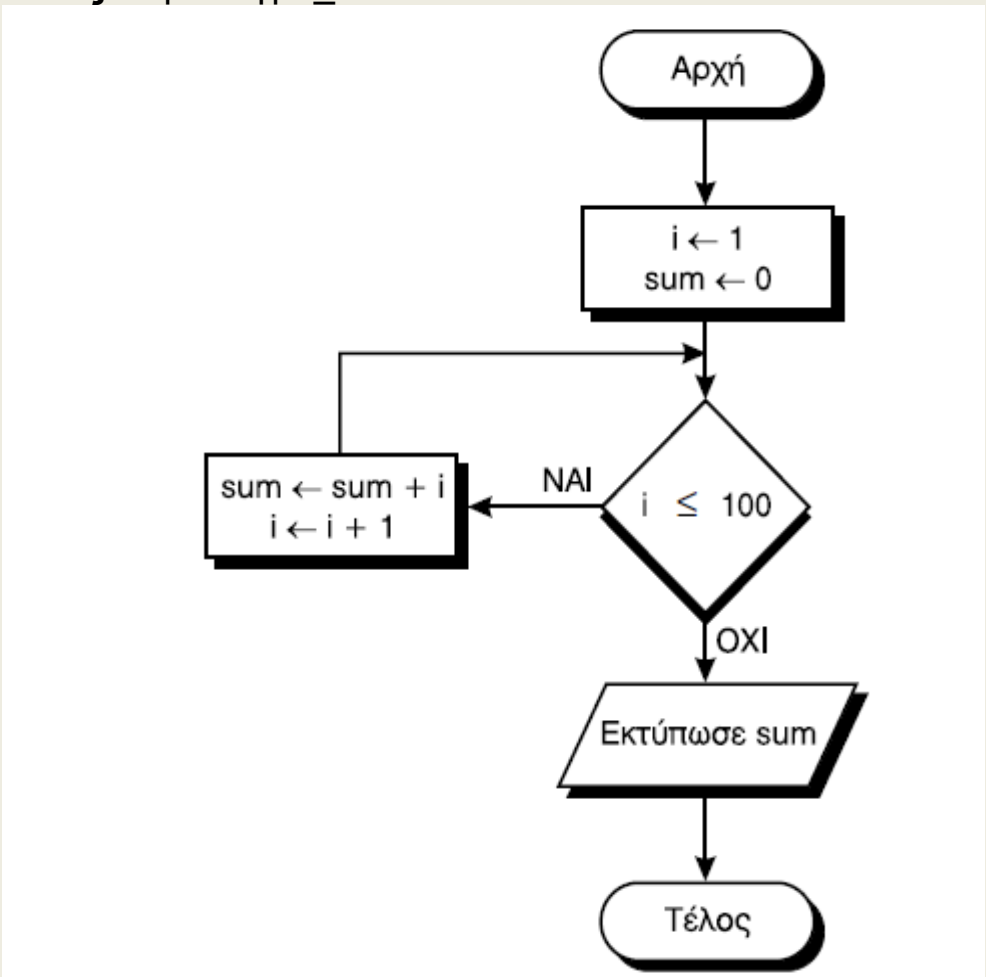
Για *μεταβλητή* **από** *α_τ*	*μεταβλητή* ← α_τ
μέχρι *τ_τ*	**Όσο** *μεταβλητή* <= τ_τ **επανάλαβε**
…	**…**
εντολές	*εντολές*
…	**…**
με βήμα *β*	*μεταβλητή* ← *μεταβλητή* + β
Τέλος_επανάληψης	**Τέλος_επανάληψης**

Από τα προηγούμενα γίνεται φανερός ο τρόπος χρήσης της εντολής Για…από…μέχρι. Ας σημειωθεί ωστόσο, ότι υπάρχουν κάποιες δεσμεύσεις μεταξύ των τιμών από, μέχρι και βήμα.

- Έτσι **το βήμα δεν μπορεί να είναι μηδέν**, γιατί τότε ο βρόχος εκτελείται επ' άπειρον.
- Είναι δυνατόν όμως το **βήμα να έχει αρνητική τιμή**, αρκεί η τιμή **από α_τ** να είναι **μεγαλύτερη** από την τιμή **μέχρι τ_τ**, όπως για παράδειγμα στην επόμενη εντολή:
  Για k **από** 100 **μέχρι** 0 **με_βήμα** -1
- Επίσης οι τιμές από, μέχρι και βήμα δεν είναι απαραίτητο να είναι ακέραιες. Μπορούν λάβουν οποιαδήποτε πραγματική τιμή. Για παράδειγμα, όταν ζητείται να βρεθούν διαδοχικές τιμές μιας συνάρτησης f(x) για x από 0 έως 1, τότε μπορεί να γραφεί η επόμενη εντολή:
  Για x **από** 0 **μέχρι** 1 **με_βήμα** 0,01
- Το βήμα μεταβολής της μεταβλητής υπονοείται και δεν σημειώνεται, όταν είναι 1.
- Ποτέ δεν αλλάζουμε την τιμή της μεταβλητής μέσα σε μια *για…*

Παράδειγμα 10. Υπολογισμός αθροίσματος αριθμών με επαναληπτική εντολή: για...από...μέχρι
Να βρεθεί και να εκτυπωθεί το άθροισμα των 100 ακεραίων από το 1 μέχρι το 100.

Αλγόριθμος Παράδειγμα_10
Sum ← 0
Για i **από** 1 **μέχρι** 100
    Sum ← Sum + i
Τέλος_επανάληψης
Εκτύπωσε Sum
Τέλος Παράδειγμα_10



Σχ. 2.6. Ο αλγόριθμος του παραδείγματος 10 με διάγραμμα ροής

Άσκηση 62: Να γράψετε τους αλγόριθμους των ασκήσεων 36,45,46 με χρήση της *για..από…μέχρι*.

Άσκηση 63 ΘΕΜΑ 1ο Β 2001
Άσκηση 64 ΘΕΜΑ 1ο Δ 2004
Άσκηση 65 ΘΕΜΑ 1ο Α2 και 1Γ 2005
Άσκηση 66 ΘΕΜΑ 2ο 2 2006
Άσκηση 67 ΘΕΜΑ 2Ο 2000
Άσκηση 68 ΘΕΜΑ 2Ο ΕΣΠΕΡΙΝ ΕΠΑΝΑΛ 2006
Άσκηση 69 ΘΕΜΑ 1Ο Β ΕΠΑΝΑΛ 2007
Άσκηση 70 ΘΕΜΑ 1Ο Β και 2ο ΕΣΠΕΡΙΝ 2007
Άσκηση 71 ΘΕΜΑ 1Ο Δ ΕΠΑΝΑΛ 2001
Άσκηση 72 ΘΕΜΑ 4Ο ΕΣΠΕΡΙΝ 2002

Αθροίσματα σειρών

Μια κατηγορία προβλημάτων τα οποία είναι συνήθως γνωστού πλήθους και χρησιμοποιείται η για…από…μέχρι είναι τα προβλήματα αθροίσματος σειρών. Έχουμε ήδη δει τέτοιο στο παράδείγμα 10.

Άσκηση 73: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
1+4+7+10+…+100

Άσκηση 74: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
1/1 +1/4 + 1/7 + 1/10 +…+ 1/100

Άσκηση 75: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
3/4 + 6/7 + 9/10 +…+ 99/100

Άσκηση 76: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
(2*3)/4 + (3*4)/5 + (4*5)/6 + … + (98*99)/100

Άσκηση 77: Να γραφεί ένας αλγόριθμος που θα βρίσκει το γινόμενο:
1*2*3*4*…*50

Άσκηση 78: Το παραγοντικό ενός μη αρνητικού ακέραιου αριθμού ορίζεται ως εξής: 0!=1, 1!=1, ν!=1*2*…*ν για κάθε ακέραιο ν>=2. Να γράψετε αλγόριθμο που θα βρίσκει το παραγοντικό ενός δεδομένου μη αρνητικού ακέραιου αριθμού.

Παράδειγμα 10β: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
1-2+3-4+5-6+…+99-100

Στα αθροίσματα αυτά που το πρόσημο εναλλάσσεται μια προφανής λύση είναι η εξής:

```
Αλγόριθμος Παράδειγμα_10β
Sum ← 0
Για i από 1 μέχρι 100
      Αν i MOD 2 <> 0 τότε
            Sum ← Sum + i
        αλλιώς
            Sum ← Sum – i
      Τέλος_Αν
Τέλος_επανάληψης
Εκτύπωσε Sum
Τέλος Παράδειγμα_10β
```

Μια πιο καλή λύση είναι η ακόλουθη:

```
Αλγόριθμος Παράδειγμα_10β
πρόσημο←1
Sum ← 0
Για i από 1 μέχρι 100
      Sum ← Sum + i*πρόσημο
      πρόσημο←πρόσημο*(-1)
Τέλος_επανάληψης
Εκτύπωσε Sum
Τέλος Παράδειγμα_10β
```

Άσκηση 79: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
-1/2 + 1/4 - 1/6 + 1/8 +... 1/100

Άσκηση 80: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
3/4 - 6/7 + 9/10 – 12/13 +... 99/100

Άσκηση 81: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
+1/1 + 1/2 - 1/3 + 1/4 + 1/5 - 1/6 + 1/7 +... 1/100

Άσκηση 82: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
(2*3)/4 - (3*4)/5 + (4*5)/6 - ... +(98*99)/100

Άσκηση 83: Να γραφεί αλγόριθμος που θα βρίσκει το άθροισμα :
2/1! - 3/2! + 4/3! - 5/4! + 6/5! - ... +30/29!

Άσκηση 84: Να τροποποιηθεί ο παραπάνω αλγόριθμος ώστε να βρίσκει το άθροισμα 2/1! - 3/2! + 4/3! - 5/4! + 6/5! - ... με ακρίβεια 3 δεκαδικών ψηφίων.

Εύρεση μέγιστου - ελάχιστου

Άσκηση 85: Να γράψετε αλγόριθμο (χωρίς χρήση καμίας επαναληπτικής διαδικασίας) που θα διαβάζει 3 αριθμούς και θα βρίσκει τον μεγαλύτερο. (Π.χ. 3,7,2 μεγαλύτερος το 7).

Ο αλγόριθμος της άσκησης 85 είναι βολικός για 3 αριθμούς αλλά είναι σχεδόν αδύνατον να τροποποιηθεί και να εφαρμόζεται σε ένα μεγάλο πλήθος αριθμών π.χ. 20 ή 100 κ.λ.π. Σ ένα τόσο μεγάλο πλήθος εφαρμόζουμε επαναληπτική διαδικασία και την ακόλουθη λογική: (Την ίδια λογική εφαρμόζουμε και με το μυαλό μας όταν π.χ. έχουμε μια λίστα από πολλούς αριθμούς και ψάχνουμε να βρούμε τον μεγαλύτερο ή τον μικρότερο).

- Θεωρούμε μια μεταβλητή (max ή min) η οποία κάθε στιγμή θα κρατά την "*μέχρι τώρα*" μεγαλύτερη ή μικρότερη τιμή.
- Πριν να αρχίσει η επαναληπτική διαδικασία δίνουμε στην μεταβλητή (max ή min) μια αρχική τιμή. Εδώ έχουμε 2 επιλογές:
    1. Αν ψάχνουμε την max δίνουμε σ αυτή μια υπερβολικά μικρή και παράλογη τιμή. Αν ψάχνουμε την min δίνουμε σ αυτήν μια υπερβολικά μεγάλη παράλογη τιμή.
    2. Αφού διαβάζουμε τον πρώτο αριθμό από τη λίστα μας (έξω από την επαναληπτική διαδικασία πριν αυτή αρχίσει) δίνουμε στην μεταβλητή (max ή min) την τιμή αυτή.
- Στην συνέχεια αρχίζει η επαναληπτική διαδικασία η οποία διαβάζει τις (υπόλοιπες) τιμές. Επιπλέον κάθε τιμή που διαβάζει την συγκρίνει με την "μέχρι τώρα" max ή min. Αν η τιμή αυτή είναι μεγαλύτερη ή μικρότερη αντίστοιχα τότε αλλάζουμε την τιμή της max ή min και της δίνουμε την τιμή του αριθμού που μόλις διαβάσαμε.
- Όταν τελειώσει η επαναληπτική διαδικασία η max ή min θα έχει την μεγαλύτερη ή μικρότερη τιμή αντίστοιχα.

Γενικό σχήμα:

```
…
max←a_τ
…
Εντολή αρχής επαναληπτικής διαδικασίας
        …
        Αν x>max τότε max←x
        …
Εντολή τέλους επαναληπτικής διαδικασίας
…
```

Γνωστού πλήθους:

```
Αλγόριθμος μέγιστο_α_τρόπος
Διάβασε Ν

max←-999999
Για i από 1 μέχρι Ν
        Διάβασε x
        Αν x>max τότε
             max←x
        Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε max
Τέλος μέγιστο_α_τρόπος
```

```
Αλγόριθμος μέγιστο_β_τρόπος
Διάβασε Ν
Διάβασε x
max←x
Για i από 2 μέχρι Ν
        Διάβασε x
        Αν x>max τότε
             max←x
        Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε max
Τέλος μέγιστο_β_τρόπος
```

Άγνωστου πλήθους με έλεγχο στη τιμή_φρουρό:

```
Αλγόριθμος ελάχιστο_α_τρόπος
min←99999
Διάβασε x
Όσο x <>99999 επανάλαβε
        Αν x<min τότε
             min←x
        Τέλος_αν
! Η Αν πριν την Διάβασε για να μην συγκρίνει
! και την τιμή φρουρό
        Διάβασε x
Τέλος_επανάληψης
Εκτύπωσε min
Τέλος ελάχιστο_α_τρόπος
```

```
Αλγόριθμος ελάχιστο_β_τρόπος
Διάβασε x
min←x
Όσο x <>99999 επανάλαβε
        Αν x<min τότε
             min←x
        Τέλος_αν
! Η Αν πριν την Διάβασε για να μην συγκρίνει
! και την τιμή φρουρό
        Διάβασε x
Τέλος_επανάληψης
Εκτύπωσε min
Τέλος ελάχιστο_β_τρόπος
```

Άσκηση 86: Να συμπληρώσετε τους γενικούς αλγορίθμους αθροισμάτων (6 αλγόριθμοι) ώστε να βρίσκουν εκτός από το άθροισμα και το ελάχιστο και μέγιστο (και τα 2) των αριθμών.

Πολλές φορές μαζί με την τιμή του μέγιστου ή του ελάχιστου χρειάζεται να βρούμε και "*ποιος*" έχει αυτό το μέγιστο ή ελάχιστο, δηλαδή είτε ένα όνομα είτε έναν αριθμό κατάταξης π.χ. ο τρίτος ή ο πέμπτος κατά σειρά εισαγωγής των δεδομένων. Φυσικά αν υπάρχουν περισσότεροι από ένας χωρίς πίνακες δεν μπορούμε να τους βρούμε όλους παρά μόνο έναν (τον πρώτο ή τον τελευταίο). Γι αυτό, προς το παρόν σε τέτοια προβλήματα υποθέτουμε πως όλες οι τιμές είναι διαφορετικές επομένως ένας είναι αυτός με την μέγιστη ή ελάχιστη τιμή. Με το παραπάνω σκεπτικό οι παραπάνω αλγόριθμοι τροποποιούνται ως εξής:

```
Αλγόριθμος μέγιστο_α_τρόπος
Διάβασε N

max←-999999


Για i από 1 μέχρι N
        Διάβασε ον,x
        Αν x>max τότε
            max←x
            ovmax←ον
            imax←i
        Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε max,imax,ovmax
Τέλος μέγιστο_α_τρόπος
```

```
Αλγόριθμος μέγιστο_β_τρόπος
Διάβασε N
Διάβασε ον,x
max←x
ovmax←ον
imax←1
Για i από 2 μέχρι N
        Διάβασε ον,x
        Αν x>max τότε
            max←x
            ovmax←ον
            imax←i
        Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε max,imax,ovmax
Τέλος μέγιστο_β_τρόπος
```

```
Αλγόριθμος ελάχιστο_α_τρόπος
Διάβασε  x
i←1
min←99999


Όσο x <>99999 επανάλαβε
        Διάβασε ον
        Αν x<min τότε
            min←x
            imin←i
            ovmin←ον
        Τέλος_αν
! Η Αν πριν την Διάβασε για να μην συγκρίνει
! και την τιμή φρουρό
        Διάβασε x
        i←i+1
Τέλος_επανάληψης
Εκτύπωσε min,imin,ovmin
Τέλος ελάχιστο_α_τρόπος
```

```
Αλγόριθμος ελάχιστο_β_τρόπος
Διάβασε ον,x
i←1
min←x
imin←i
ovmin←ον
Όσο x <>99999 επανάλαβε
        Αν x<min τότε
            min←x
            imin←i
            ovmin←ον
        Τέλος_αν
! Η Αν πριν την Διάβασε για να μην συγκρίνει
! και την τιμή φρουρό
        Διάβασε ον,x
        i←i+1
Τέλος_επανάληψης
Εκτύπωσε min
Τέλος ελάχιστο_β_τρόπος
```

Άσκηση 87 ΘΕΜΑ 3^{ο} εσπεριν 2002

Άσκηση 88: Στην είσοδο ενός τελωνείου υπάρχει μια πλαστιγγογέφυρα η οποία μετρά το βάρος των φορτηγών που μπαίνουν μέσα μεταφέροντας εμπορεύματα. Η τιμή αυτή μεταβιβάζεται αυτόματα σ έναν υπολογιστή. να γράψετε αλγόριθμο που θα υπολογίζει το συνολικό

βάρος όλων των φορτηγών, καθώς και τον μέσο όρο βάρους των φορτηγών που μπαίνουν στο τελωνείο κατά τη διάρκεια ενός 24ωρου. Επίσης θα βρίσκει το βάρος και την σειρά εισόδου (π.χ. $3^{ο}$ ή $7^{ο}$ κ.λ.π) του ελαφρύτερου και του βαρύτερου φορτηγού. Ο αλγόριθμος θα τερματίζει όταν ο χρήστης κάποια στιγμή εισάγει (απ το πληκτρολόγιο) μια κατάλληλη τιμή φρουρό.

Άσκηση 89: Ο Κώστας πήγε στο Μασούτη με 50^{E} στη τσέπη και μια λίστα με ψώνια. Να γράψετε αλγόριθμο που θα διαβάζει τις τιμές των προϊόντων που βάζει στο καλάθι, θα σταματά πριν συμπληρωθούν τα 50^{E} και θα τυπώνει το πλήθος των προϊόντων που αγόρασε, την τιμή του φτηνότερου προϊόντος, την σειρά με την οποία αυτό μπήκε στο καλάθι, καθώς και πόσα Ε περίσσεψαν.

Άσκηση 90: Να γράψετε αλγόριθμο που θα διαβάζει κάθε μάθημα (τίτλο μαθήματος) και τον αντίστοιχο βαθμό του ελέγχου και θα εκτυπώνει για κάθε μάθημα αύξοντα αριθμό, τίτλο μαθήματος, και βαθμό του ελέγχου ενός μαθητή και θα υπολογίζει και θα εκτυπώνει τον Μ.Ο. της βαθμολογίας του. Στο Μ.Ο. δεν προσμετράτε ο βαθμός της Γυμναστικής. Επίσης θα βρίσκει και θα εκτυπώνει το μάθημα (με εξαίρεση την Γυμναστική) με τον μικρότερο βαθμό και ποιος είναι αυτός.

Μερικές φορές χρειάζεται να βρούμε το **ελάχιστο** ή το **μέγιστο ξεχωριστών κατηγοριών** π.χ. ξεχωριστά το μέγιστο των αγοριών από αυτό των κοριτσιών. Στη περίπτωση αυτή χρησιμοποιούμε ξεχωριστά σετ μεταβλητών max, ovmax, imax κ.λπ. ένα για κάθε κατηγορία.

Άσκηση 91: Να γράψετε αλγόριθμο που θα διαβάζει (με τυχαία ανάμικτη σειρά) το όνομα, το φύλο (Α,Κ) και τον τελικό βαθμό στην Α Λυκείου των μαθητών της τάξης. Θα τυπώνει λίστα με αύξοντα αριθμό, όνομα και βαθμό και θα υπολογίσει και θα τυπώνει τον Μ.Ο. βαθμολογίας όλων των μαθητών καθώς και το Μ.Ο. βαθμολογίας των αγοριών και των κοριτσιών και μήνυμα ποιοι έχουν καλύτερες επιδόσεις τα αγόρια ή τα κορίτσια; Επίσης θα τυπώνει το όνομα του αγοριού και το όνομα του κοριτσιού με την καλύτερη επίδοση ώστε να είναι οι παραστάτες.

Άσκηση 92: Στα Βοδuline κάθε μέρα έρχεται για εγγραφή ένα πλήθος ανδρών και γυναικών. Να γράψετε αλγόριθμο ο οποίος θα διαβάζει το όνομα, το φύλλο και το βάρος του κάθε πελάτη/τισσας και στο τέλος της ημέρας θα τυπώνει το μέσο όρο βάρους των ανδρών, των γυναικών, το όνομα και το βάρος της χοντρότερης γυναίκας και το όνομα και το βάρος του χοντρότερου άνδρα. Επίσης θα τυπώνει το όνομα, το φύλο και το βάρος του ελαφρύτερου πελάτη ανεξαρτήτως φύλου.

Εμφωλευμένες επαναληπτικές δομές

Πολλές φορές είναι απαραίτητο μέσα σε μια επαναληπτική δομή να υπάρχει μια άλλη επαναληπτική δομή. Στις περιπτώσεις αυτές αν το πλήθος της κάθε επαναληπτικής δομής είναι γνωστό, οι εντολές που βρίσκονται μέσα στην εσωτερική επαναληπτική δομή θα εκτελεσθούν τόσες φορές όσες είναι το γινόμενο του πλήθους της κάθε μιας. ***Ποτέ δεν χρησιμοποιούμε την ίδια μεταβλητή ως μετρητή σε εμφωλευμένες επαναληπτικές δομές!***

Παράδειγμα 11. Διοφαντική ανάλυση
Να εκπονηθεί ένας αλγόριθμος για την εύρεση όλων των ακεραίων λύσεων της εξίσωσης 3x + 2y – 7z = 5 για τιμές των x, y, z μεταξύ των 0 και 100. Η επίλυση τέτοιων εξισώσεων με πολλές μεταβλητές που επιδέχονται πολλές λύσεις, ονομάζεται διοφαντική ανάλυση. Αλγοριθμικά το πρόβλημα αντιμετωπίζεται ως εξής:

```
Αλγόριθμος Διοφαντική
Για x από 0 μέχρι 100
      Για y από 0 μέχρι 100
            Για z από 0 μέχρι 100
                  Αν 3x+2y-7z=5 τότε Εκτύπωσε x,y,z
            Τέλος_επανάληψης
      Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος Διοφαντική
```

```
Αρχικά για x←0, y←0      ελέγχει όλες τις τιμές του z από 0 → 100,
μετά       x←0, y←1      ελέγχει όλες τις τιμές του z από 0 → 100,
μετά       x←0, y←2      ελέγχει όλες τις τιμές του z από 0 → 100,
μετά       x←0, …        ελέγχει όλες τις τιμές του z από 0 → 100
μετά       x←0, y←100    ελέγχει όλες τις τιμές του z από 0 → 100,
μετά       x←1, y←0      ελέγχει όλες τις τιμές του z από 0 → 100,
μετά       x←1, …        ελέγχει όλες τις τιμές του z από 0 → 100
μετά       x←1, y←100    ελέγχει όλες τις τιμές του z από 0 → 100,
…
μετά       x←100, y←100  ελέγχει όλες τις τιμές του z από 0 → 100,
```

Άσκηση 93: Να γράψετε αλγόριθμο που θα τυπώνει τον πίνακα της προπαίδειας του πολλαπλασιασμού.

Άσκηση 94 ΘΕΜΑ 2ο επαναλ 2001
Άσκηση 95 ΘΕΜΑ 3ο εσπεριν 2006
Άσκηση 96 ΘΕΜΑ 1ο Γ εσπεριν 2004
Άσκηση 97 ΘΕΜΑ 1ο Ε εσπεριν επαναλ 2005

Πολλαπλασιασμός αλά ρωσικά

Ας θεωρήσουμε την πράξη του πολλαπλασιασμού δύο ακεραίων αριθμών και ας θυμηθούμε πως αυτή υλοποιείται χειρωνακτικά.

```
  45
x 19
--------
  405
+ 45
--------
  855
```

Ωστόσο, η πράξη του πολλαπλασιασμού δεν εκτελείται από τον υπολογιστή με τον τρόπο αυτό.

Ας θυμηθούμε ότι:
Όσο και αν τυχόν ξαφνιάζει, **ο υπολογιστής** δεν μπορεί να **εκτελεί** παρά **μόνο τρεις λειτουργίες** :

_ **πρόσθεση**, η οποία αποτελεί τη **βασική αριθμητική πράξη**, δεδομένου ότι και οι άλλες αριθμητικές πράξεις μπορούν να αντιμετωπιστούν, σαν διαδικασίες πρόσθεσης
_ **σύγκριση**, η οποία συνιστά τη **βασική λειτουργία** για την επιτέλεση ό-λων των **λογικών πράξεων**,
_ **μεταφορά δεδομένων**, λειτουργία που προηγείται και έπεται της επε-ξεργασίας δεδομένων.

Με βάση αυτές τις τρεις λειτουργίες διεκπεραιώνει όλες τις εργασίες που του αναθέτονται και επιλύει όλα τα προβλήματα που αναλαμβάνει.

Με βάση τα παραπάνω ο υπολογιστής θα μπορούσε να εκτελέσει την πράξη του πολλαπλασιασμού κάνοντας πολλές προσθέσεις π.χ. προσθέτοντας 19 φορές το 45 στον εαυτό του.

Άσκηση 98: Να γράψετε αλγόριθμο που σαν είσοδο θα δέχεται 2 ακέραιους μη αρνητικούς αριθμούς και θα βρίσκει και θα τυπώνει το γινόμενό τους χωρίς την χρήση του αριθμητικού τελεστή * . Στη συνέχεια τροποποιήσετε και επεκτείνετε τον αλγόριθμο ώστε να λειτουργεί και για αρνητικούς ακέραιους.

Ο τρόπος αυτός όμως είναι πολύ χρονοβόρος για τον υπολογιστή καθώς αν π.χ. θέλαμε να βρούμε το γινόμενο: 996.500Χ998.561 θα έπρεπε να κάνει σχεδόν ένα εκατομμύριο προσθέσεις.

Για το λόγο αυτό ο χρησιμοποιούμενος τρόπος είναι ο λεγόμενος *πολλαπλασιασμός αλά ρωσικά*. Χωρίς βλάβη της γενικότητας θεωρούμε ότι οι ακέραιοι είναι θετικοί (μεγαλύτεροι του μηδενός), αλλά η μέθοδος μπορεί εύκολα να μετατραπεί, ώστε να περιγράψει και την περίπτωση των αρνητικών ακεραίων. Πως ακριβώς λειτουργεί η μέθοδος θα φανεί με

το επόμενο παράδειγμα, όπου περιγράφεται ο αλγόριθμος με ελεύθερο κείμενο.

Έστω, λοιπόν, ότι δίνονται δύο θετικοί ακέραιοι αριθμοί, οι αριθμοί 45 και 19. Οι αριθμοί γράφονται δίπλα-δίπλα και ο πρώτος διπλασιάζεται αγνοώντας το δεκαδικό μέρος, ενώ ο δεύτερος υποδιπλασιάζεται. Στο σχήμα 2.8 παρουσιάζεται η επαναλαμβανόμενη διαδικασία, που συνεχίζεται μέχρις ότου στη δεύτερη στήλη να προκύψει μονάδα. Τελικώς, το γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης, όπου αντίστοιχα στη δεύτερη στήλη υπάρχει περιττός αριθμός. Για το παράδειγμά μας, τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη.

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

Ο αλγόριθμος αυτός είναι αποδοτικός για τους εξής λόγους:

1. Χρειάζεται πολύ μικρότερος αριθμός πράξεων. Π.χ. εκεί που στο γινόμενο: 996.500Χ998.561 θα έπρεπε να κάνει σχεδόν ένα εκατομμύριο προσθέσεις τώρα χρειάζεται 20 διαιρέσεις με το 2, 20 πολλαπλασιασμούς με το 2, 20 συγκρίσεις αν ένας αριθμός είναι περιττός και το μάξιμουμ 20 προσθέσεις αν όλοι είναι περιττοί δηλαδή συνολικά μάξιμουμ 80 πράξεις!!!! (Ερώτηση: Πόσες πράξεις το μάξιμουμ θα χρειαζόμασταν αν ήταν να πολλαπλασιάσουμε περίπου 1 δις Χ 1 δις;)
2. Επιπλέον οι δύο πράξεις που απαιτούνται στη μέθοδο αυτή δηλαδή ο πολλαπλασιασμός επί 2 και η ακέραια διαίρεση δια 2 υλοποιούνται πολύ εύκολα και γρήγορα με τον υπολογιστή μέσω της **ολίσθησης** για τον ακόλουθο λόγο: Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή, δηλαδή 0 και 1, ανεξάρτητα από το πως τα ορίζει ο προγραμματιστής, όπως ακεραίους ή πραγματικούς σε δεκαδικό σύστημα, ή ακόμη χαρακτήρες κ.λπ. Έτσι ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό 00010001 του δυαδικού συστήματος, ο οποίος μπορεί να αποθηκευθεί σε ένα byte. Αν μετακινήσουμε τα ψηφία αυτά κατά μία θέση προς τα αριστερά, δηλαδή αν προσθέσουμε ένα 0 στο τέλος του αριθμού και αγνοήσουμε το αρχικό 0, τότε προκύπτει ο αριθμός 00100010 του δυαδικού συστήματος, που ισοδυναμεί με το αριθμό 34 του δεκαδικού συστήματος. Επίσης, με παρόμοιο τρόπο, αν μετακινήσουμε τα ψηφία κατά μία θέση δεξιά, δηλαδή αποκόψουμε το τελευταίο ψηφίο 1 και θεωρήσουμε ένα ακόμη αρχικό 0, τότε προκύπτει ο αριθμός 00001000 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος. Άρα η ολίσθηση προς τα αριστερά ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η ολίσθηση προς τα δεξιά ισοδυναμεί με την ακέραια διαίρεση δια 2.

Άσκηση 99: Να γράψετε αλγόριθμο που σαν είσοδο θα δέχεται 2 ακέραιους μη αρνητικούς αριθμούς και θα βρίσκει και θα τυπώνει με πολλαπλασιασμό αλλά ρώσικα το γινόμενό τους. Ο αλγόριθμος να γραφεί σε ψευδογλώσσα, σε διάγραμμα ροής και σε φυσική γλώσσα κατά βήματα. Στη συνέχεια τροποποιήσετε και επεκτείνετε τον αλγόριθμο ώστε να λειτουργεί και για αρνητικούς ακέραιους.

Επαναληπτικές ασκήσεις:

Άσκηση 100 ΘΕΜΑ $1^{Ο}$ Α, 2001
Άσκηση 101 ΘΕΜΑ $1^{Ο}$ Α,Β 2003
Άσκηση 102 ΘΕΜΑ $1^{Ο}$ Β 2004
Άσκηση 103 ΘΕΜΑ $1^{Ο}$ Α1 2005
Άσκηση 104 ΘΕΜΑ $1^{Ο}$ Δ 2006
Άσκηση 105 ΘΕΜΑ $1^{Ο}$ Γ 2007
Άσκηση 106 ΘΕΜΑ $1^{Ο}$ Γ1,Δ 2008
Άσκηση 107 ΘΕΜΑ $1^{Ο}$ Α, Β2, Γ1,Γ2, 200Ο
Άσκηση 108 ΘΕΜΑ $1^{ο}$ Α,Β,Γ εσπεριν επαναλ 2006
Άσκηση 109 ΘΕΜΑ $1^{ο}$ Γ1 επαναλ 2007
Άσκηση 110 ΘΕΜΑ $1^{ο}$ Α,Δ ΘΕΜΑ $3^{Ο}$ εσπεριν 2007
Άσκηση 111 ΘΕΜΑ $1^{ο}$ Α,Β,Ε ΘΕΜΑ $3^{Ο}$ ΕΠΑΝΑΛ 2001
Άσκηση 112 ΘΕΜΑ $1^{ο}$ Β εσπεριν 2006
Άσκηση 113 ΘΕΜΑ $1^{ο}$ Α ΘΕΜΑ $3^{Ο}$ ΘΕΜΑ $4^{Ο}$ Β εσπεριν 2000
Άσκηση 114 ΘΕΜΑ $1^{ο}$ Γ εσπεριν 2002
Άσκηση 115 ΘΕΜΑ $4^{Ο}$ εσπεριν 2004
Άσκηση 116 ΘΕΜΑ $1^{ο}$ Γ εσπεριν 2005
Άσκηση 117 ΘΕΜΑ $1^{ο}$ Α,Δ ΘΕΜΑ $2^{Ο}$ εσπεριν ΕΠΑΝΑΛ 2005

ΚΕΦΑΛΑΙΟ 3

3.1 Δεδομένα

Τα *δεδομένα* (data) είναι η αφαιρετική αναπαράσταση της πραγματικότητας και συνεπώς μία απλοποιημένη όψη της. Τα δεδομένα, λοιπόν, είναι ακατέργαστα γεγονότα, και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος.

Η **συλλογή** των ακατέργαστων δεδομένων και ο **συσχετισμός** τους δίνει ως αποτέλεσμα την ***πληροφορία*** (information). Δεν είναι εύκολο να δοθεί επακριβής ορισμός της έννοιας της πληροφορίας, αλλά μπορεί να θεωρηθεί ότι ο **αλγόριθμος είναι το μέσο για την παραγωγή πληροφορίας από τα δεδομένα.**

Με βάση τις δεδομένες **πληροφορίες λαμβάνονται διάφορες αποφάσεις και γίνονται ενέργειες**. Στη συνέχεια αυτές οι ενέργειες παράγουν νέα δεδομένα, νέες πληροφορίες, νέες αποφάσεις, νέες ενέργειες κοκ. **(κύκλος επεξεργασίας δεδομένων**).

Η **μέτρηση**, η **κωδικοποίηση**, η **μετάδοση** της **πληροφορίας** αποτελεί αντικείμενο μελέτης ενός ιδιαίτερου κλάδου, της ***Θεωρίας Πληροφοριών*** (Information Theory), που είναι ένα ιδιαίτερα σημαντικό πεδίο της Πληροφορικής.

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 κ.λ.π.

_ **Γλωσσών προγραμματισμού**. Οι γλώσσες προγραμματισμού υψηλού επιπέδου επιτρέπουν τη χρήση διάφορων τύπων μεταβλητών για να περιγράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης κάθε μεταβλητής στον Η/Υ.

_ **Δομών Δεδομένων**. Δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών.

_ **Ανάλυσης Δεδομένων**. Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των Βάσεων Δεδομένων, της Μοντελοποίησης Δεδομένων και της Αναπαράστασης Γνώσης ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

3.2 Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στη δευτερεύουσα μνήμη του. Η αποθήκευση αυτή δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή.

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

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

Οι **βασικές λειτουργίες** (ή αλλιώς **πράξεις**) επί των δομών δεδομένων είναι οι ακόλουθες:

_ ***Προσπέλαση*** (access), πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.

_ ***Εισαγωγή*** (insertion), δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.

_ ***Διαγραφή*** (deletion), που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.

_ ***Αναζήτηση*** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.

_ ***Ταξινόμηση*** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

_ ***Αντιγραφή*** (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.

_ ***Συγχώνευση*** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.

_ ***Διαχωρισμός*** (separation), κατά την οποία μία ενιαία δομή χωρίζεται σε δύο ή περισσότερες δομές και αποτελεί την αντίστροφη πράξη της συγχώνευσης.

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

αφ' ενός την ύπαρξη διαφορετικών δομών, και αφ' ετέρου τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.

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

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

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

Με τον όρο ***στατική δομή δεδομένων*** εννοείται ότι το ακριβές **μέγεθος** της απαιτούμενης **κύριας μνήμης** καθορίζεται κατά τη στιγμή του **προγραμματισμού**, και κατά συνέπεια κατά τη στιγμή της μετάφρασης του προγράμματος και **όχι κατά τη στιγμή της εκτέλεσης** του προγράμματος.
Μία άλλη σημαντική διαφορά σε σχέση με τις δυναμικές δομές είναι ότι τα στοιχεία των στατικών δομών αποθηκεύονται σε **συνεχόμενες θέσεις** μνήμης.

Οι ***δυναμικές δομές*** **δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης** αλλά στηρίζονται στην τεχνική της λεγόμενης *δυναμικής παραχώρησης μνήμης* (dynamic memory allocation). Με άλλα λόγια, οι δομές αυτές **δεν έχουν σταθερό μέγεθος**, αλλά **το πλήθος των κόμβων** τους **μεγαλώνει και μικραίνει κατά τη στιγμή της εκτέλεσης** του προγράμματος καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα. Όλες οι σύγχρονες γλώσσες προγραμματισμού προσφέρουν τη δυνατότητα δυναμικής παραχώρησης μνήμης.

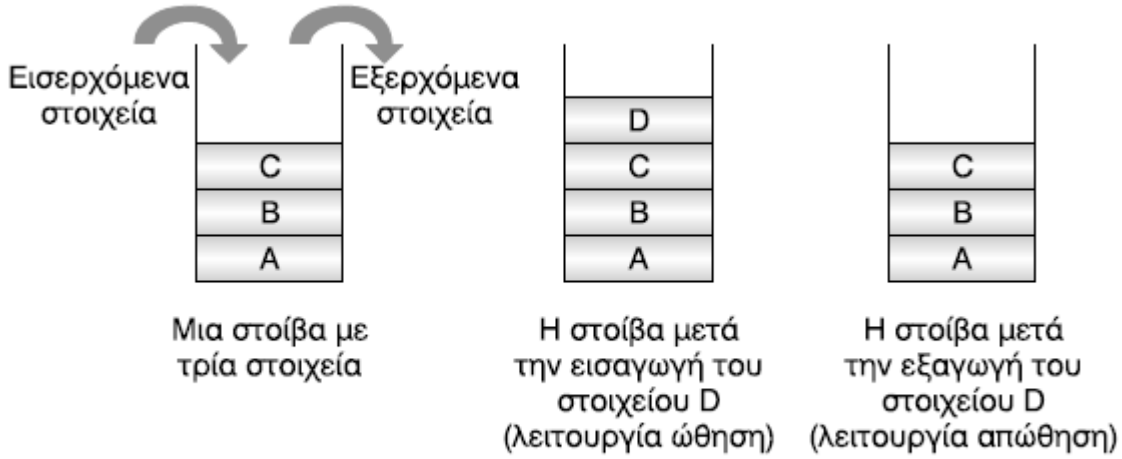
Επίσης διακρίνονται σε ομογενείς δομές δεδομένων (όλα τα δεδομένα είναι του ίδιου τύπου) όπως π.χ. ένας *πίνακας* και ετερογενείς δομές δεδομένων (τα δεδομένα μπορεί να είναι διαφορετικού τύπου) όπως π.χ. μια εγγραφή (αρχείου).

3.3 Πίνακες

Βλέπε στα επόμενα κεφάλαια του προγραμματισμού

3.4 Στοίβα

Μία στοίβα δεδομένων μοιάζει με μία στοίβα από πιάτα. Για παράδειγμα, κάθε πιάτο που πλένεται τοποθετείται στην κορυφή (top) της στοίβας των πιάτων, ενώ για σκούπισμα λαμβάνεται και πάλι το πιάτο της κορυφής. Αντίστοιχα, τα δεδομένα που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία. Αυτή η μέθοδος επεξεργασίας ονομάζεται *Τελευταίο μέσα, πρώτο έξω* ή απλούστερα με την αγγλική συντομογραφία LIFO (Last-In-First-Out).



Σχ. 3.2. *Λειτουργίες στοίβας.*

Η στοίβα είναι μια δυναμική ομογενής δομή δεδομένων η οποία χρησιμοποιεί την μέθοδος επεξεργασίας *Τελευταίο μέσα, πρώτο έξω* ή απλούστερα με την αγγλική συντομογραφία LIFO (Last-In-First-Out). Σύμφωνα με τη μέθοδο αυτή τα δεδομένα που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία.

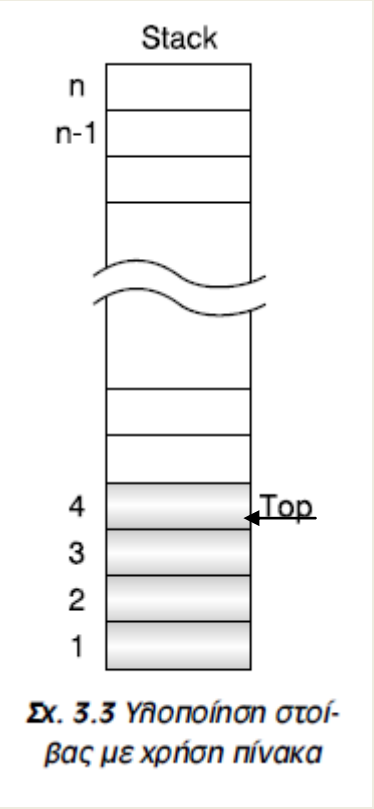
Δύο είναι οι κύριες λειτουργίες σε μία στοίβα:

_ η ***ώθηση*** (**push**) στοιχείου στην κορυφή της στοίβας, και

_ η ***απώθηση*** (**pop**) στοιχείου από τη στοίβα.

Η διαδικασία της **ώθησης** πρέπει οπωσδήποτε να **ελέγχει**, αν η στοίβα είναι γεμάτη, οπότε λέγεται ότι συμβαίνει ***υπερχείλιση*** (**overflow**) της στοίβας.
Αντίστοιχα, η διαδικασία **απώθησης ελέγχει**, αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα, δηλαδή ελέγχει αν γίνεται ***υποχείλιση*** (**underflow**) της στοίβας.

Μια στοίβα μπορεί να υλοποιηθεί πολύ εύκολα με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα 3.3. Μια βοηθητική μεταβλητή (με όνομα συνήθως **top**) χρησιμοποιείται για να δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας. Για την **εισαγωγή** ενός νέου στοιχείου στη στοίβα (**ώθηση**) αρκεί να **αυξηθεί** η μεταβλητή **top** κατά **ένα** και **στη θέση αυτή να εισέλθει** το στοιχείο. Αντίθετα για την **εξαγωγή** ενός στοιχείου από τη στοίβα (**απώθηση**) **εξέρχεται πρώτα το στοιχείο** που δείχνει η μεταβλητή **top** και στη συνέχεια η **top μειώνεται** κατά **ένα** για να δείχνει τη **νέα κορυφή**.



Σχ. 3.3 Υλοποίηση στοίβας με χρήση πίνακα

3.5 Ουρά

Οι ουρές είναι καθημερινό φαινόμενο. Για παράδειγμα, ουρές δημιουργούνται όταν άνθρωποι, αυτοκίνητα, εργασίες, προγράμματα κ.λπ. περιμένουν για να εξυπηρετηθούν.
Το θέμα είναι τόσο σημαντικό και με τέτοιες πρακτικές επιπτώσεις, ώστε ένας ιδιαίτερος κλάδος των Μαθηματικών, η **Επιχειρησιακή Έρευνα** (Operations Research), και ιδιαίτερα η **Θεωρία Ουρών** (Queueing Theory), μελετά τη **συμπεριφορά** και την **επίδοση** των ουρών.
Σε μία ουρά αναμονής με ανθρώπους, συμβαίνει να εξυπηρετείται εκείνος που στάθηκε στην ουρά πρώτος από όλους τους άλλους (αν και υπάρχουν εξαιρέσεις που όμως δεν θα εξετασθούν στο βιβλίο αυτό). Η μέθοδος αυτή επεξεργασίας ονομάζεται *Πρώτο μέσα, πρώτο έξω* ή απλούστερα ακολουθώντας την αγγλική συντομογραφία FIFO (First-In-First-Out).

Η ουρά είναι μια **δυναμική ομογενής δομή δεδομένων** η οποία χρησιμοποιεί την μέθοδος επεξεργασίας *Πρώτο μέσα, πρώτο έξω* ή απλούστερα ακολουθώντας την αγγλική συντομογραφία **FIFO** (First-In-First-Out).Σύμφωνα με τη μέθοδο αυτή τα δεδομένα που εισήχθησαν πρώτα στην ουρά λαμβάνονται πρώτα, ενώ αυτά που εισήχθησαν τελευταία λαμβάνονται τελευταία.

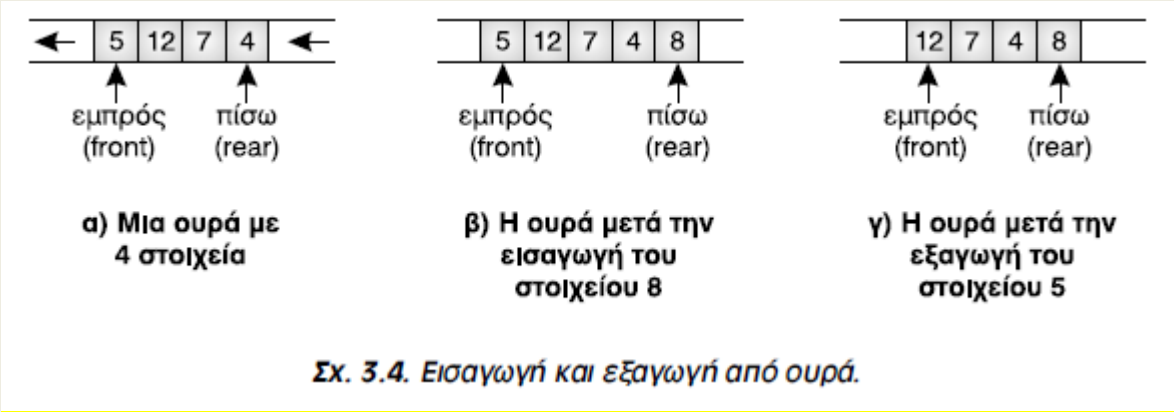
Δύο είναι οι κύριες λειτουργίες που εκτελούνται σε μία ουρά:

_ η ***εισαγωγή*** (enqueue) στοιχείου στο **πίσω** άκρο της ουράς, και

_ η ***εξαγωγή*** (dequeue) στοιχείου από το **εμπρός** άκρο της ουράς.

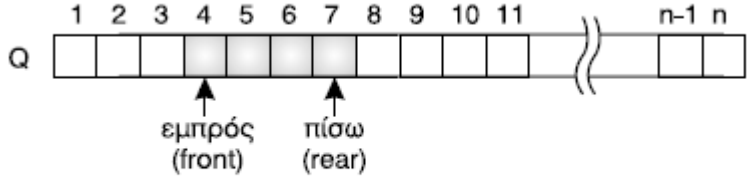
Άρα, σε αντίθεση με τη δομή της στοίβας, στην περίπτωση της ουράς απαιτούνται **δύο δείκτες**: ο ***εμπρός*** (**front**) και ο ***πίσω*** (**rear**) δείκτης, που μας δίνουν τη θέση του στοιχείου που σε πρώτη ευκαιρία **θα εξαχθεί** και τη θέση του στοιχείου που **μόλις εισήλθε**.

Στο σχήμα 3.4 φαίνεται μια ουρά με τέσσερα στοιχεία (α), στην οποία εισάγεται ένα νέο στοιχείο (β) και ακολούθως εξάγεται ένα στοιχείο.



Σχ. 3.4. Εισαγωγή και εξαγωγή από ουρά.

Μια ουρά μπορεί να υλοποιηθεί με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα 3.5. Για την **εισαγωγή** ενός νέου στοιχείου στην ουρά **αυξάνεται** ο δείκτης **rear** κατά **ένα** και **στη θέση αυτή αποθηκεύεται** το στοιχείο. Αντίστοιχα για τη λειτουργία της **εξαγωγής**, εξέρχεται **το στοιχείο που δείχνει ο δείκτης front**, ο οποίος στη συνέχεια **αυξάνεται** κατά **ένα**, για να δείχνει το **επόμενο** στοιχείο που πρόκειται να εξαχθεί.
Σε κάθε περίπτωση όμως, πρέπει να **ελέγχεται** πριν από οποιαδήποτε ενέργεια, **αν υπάρχει ελεύθερος χώρος** στον πίνακα για την εισαγωγή και **αν υπάρχει ένα τουλάχιστον στοιχείο** για την εξαγωγή.



Σχ. 3.5 Υλοποίηση ουράς με χρήση πίνακα

Άσκηση 1 ΘΕΜΑ 1° Α,Β 2002
Άσκηση 2 ΘΕΜΑ 1° Α,Β ΕΣΠΕΡΙΝ 2004
Άσκηση 3 ΘΕΜΑ 1° Β ΕΣΠΕΡΙΝ ΕΠΑΝΑΛ 2005

ΚΕΦΑΛΑΙΟ 6

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.

Ο **προγραμματισμός** ασχολείται με το τρίτο αυτό στάδιο, τη δημιουργία του προγράμματος δηλαδή του συνόλου των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος. Το πρόγραμμα, το οποίο γράφεται σε κάποια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγορίθμου, αλλά βασικό στοιχείο του είναι τα **δεδομένα** και οι **δομές δεδομένων** επί των οποίων ενεργεί. Αναφέρθηκε ήδη ότι οι αλγόριθμοι και οι δομές δεδομένων είναι μια αδιάσπαστη ενότητα.
Ο προγραμματισμός είναι αυτός που δίνει την εντύπωση ότι, οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν τα πολύπλοκα προβλήματα. Η εντύπωση αυτή όμως είναι απλώς μία ψευδαίσθηση. Ο υπολογιστής, ως γνωστόν, είναι μία μηχανή που καταλαβαίνει μόνο δύο καταστάσεις, οι οποίες αντιπροσωπεύονται με δύο αριθμούς το μηδέν και το ένα, τα ψηφία του δυαδικού συστήματος. Το μόνο πράγμα που κάνει ο υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολουθίες αυτών των δύο ψηφίων, αλλά αυτές τις ενέργειες τις εκτελεί μα ασύλληπτη ταχύτητα. Ο υπολογιστής μπορεί απλά να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.

6.2 Ιστορική αναδρομή

6.2.1 ***Γλώσσες μηχανής (1ης γενιάς)***

Αρχικά για να μπορέσει ο υπολογιστής να εκτελέσει μία οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες ακολουθίες από 0 και 1, δηλαδή εντολές σε μορφή κατανοητή από τον υπολογιστή αλλά ακατανόητες από τον άνθρωπο. Ο τρόπος αυτός ήταν επίπονος και ελάχιστοι μπορούσαν να τον υλοποιήσουν, αφού απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή. Ο πρώτος υπολογιστής ο περίφημος ENIAC για να “προγραμματιστεί”, ώστε να εκτελέσει κάποιους υπολογισμούς, έπρεπε να αλλάξουν θέση εκατοντάδες διακόπτες και να ρυθμιστούν αντίστοιχα όλες οι καλωδιώσεις, διαδικασία εξαιρετικά επίπονη και χρονοβόρα. Ο “προγραμματισμός” των πρώτων αυτών υπολογιστών, δεν ήταν ουσιαστικά προγραμματισμός με τη σημερινή έννοια του όρου. Ο υπολογιστής αναδιαρθρωνόταν, ώστε να εκτελέσει τους απαιτούμενους υπολογισμούς και στη συνέχεια έπρεπε να αλλάξει πάλι η διάρθρωσή του, ώστε να εκτελέσει έναν άλλο υπολογισμό. Οι εντολές ενός προγράμματος και σήμερα μετατρέπονται σε ακολουθίες που αποτελούνται από 0 και 1, τις εντολές σε γλώσσα μηχανής, όπως ονομάζονται, οι οποίες εκτελούνται από τον υπολογιστή.

Η **γλώσσα μηχανής** είναι η "μητρική" γλώσσα του υπολογιστή την ο οποία ο υπολογιστής "καταλαβαίνει" από κατασκευής του.

Μορφή του προγράμματος: Ένα πρόγραμμα σε **γλώσσα μηχανής** είναι μια ακολουθία δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες.

Πως εκτελεί ο υπολογιστής το πρόγραμμα: Άμεσα καθώς η γλώσσα μηχανής είναι η μόνη που "καταλαβαίνει" ο υπολογιστής εκ κατασκευής του.

Μειονεκτήματα:

1. Δύσκολο στον άνθρωπο να απομνημονεύσει τις εντολές της που είναι **ακολουθία δυαδικών ψηφίων** (0,1) δηλαδή **δυσκολία** στην **εκμάθηση** και **εκπαίδευση**.
2. **Δεν έχουν εντολές πιο σύνθετων λειτουργιών** οδηγώντας έτσι σε μακροσκελή προγράμματα, που είναι **δύσκολο** να **γραφούν** και κύρια να **συντηρηθούν** (διόρθωση λαθών, αλλαγές**)**.
3. Είναι στενά συνδεδεμένες με την **αρχιτεκτονική** του κάθε υπολογιστή την οποία θα πρέπει να γνωρίζει ο προγραμματιστής
4. τα προγράμματα δεν μπορούν να μεταφερθούν σε άλλον διαφορετικού τύπου υπολογιστή, ακόμη και του ίδιου κατασκευαστή δηλαδή δεν υπάρχει η δυνατότητα της **μεταφερσιμότητας**.
5. Μεγάλος **χρόνος** και **κόστος** παραγωγής προγραμμάτων

Πλεονεκτήματα:

- Είναι η μόνη γλώσσα που "καταλαβαίνει" ο υπολογιστής από κατασκευής του.

*6.2.2 **Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου (2^{ης} γενιάς)***

Από τα πρώτα χρόνια άρχισαν να γίνονται προσπάθειες για τη δημιουργία μίας συμβολικής γλώσσας, η οποία ενώ θα έχει έννοια για τον άνθρωπο, θα μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1. Για παράδειγμα η λέξη ADD (πρόσθεσε) ακολουθούμενη από δύο αριθμούς, είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μία ακολουθία δυαδικών ψηφίων και στη συνέχεια μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο **συμβολομεταφραστής** (assembler).

Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου είναι οι πρώτες γλώσσες προγραμματισμού που σαν εντολές έχουν συμβολικές λέξεις κατανοητές από τον άνθρωπο.

Μορφή του προγράμματος: Ένα πρόγραμμα σε ***συμβολική γλώσσα ή γλώσσα χαμηλού επιπέδου*** είναι μια ακολουθία εντολών γραμμένες με συμβολικές λέξεις κατανοητές από τον άνθρωπο.

Πως εκτελεί ο υπολογιστής το πρόγραμμα: Οι εντολές μεταφράζονται από την μορφή των συμβολικών λέξεων της συμβολικής γλώσσας σε εντολές της γλώσσας μηχανής, δηλαδή ακολουθίες δυαδικών ψηφίων και μετά εκτελούνται από τον υπολογιστή. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο **συμβολομεταφραστής** (assembler).

Μειονεκτήματα:

1. **Δεν έχουν εντολές πιο σύνθετων λειτουργιών** οδηγώντας έτσι σε μακροσκελή προγράμματα, που είναι **δύσκολο** να **γραφούν** και κύρια να **συντηρηθούν**.
2. Είναι στενά συνδεδεμένες με την **αρχιτεκτονική** του κάθε υπολογιστή την οποία θα πρέπει να γνωρίζει ο προγραμματιστής
3. τα προγράμματα δεν μπορούν να μεταφερθούν σε άλλον διαφορετικού τύπου υπολογιστή, ακόμη και του ίδιου κατασκευαστή δηλαδή δεν υπάρχει η δυνατότητα της **μεταφερσιμότητας**.
4. Μεγάλος **χρόνος** και **κόστος** παραγωγής προγραμμάτων

Πλεονεκτήματα (σε σχέση με τις γλώσσες μηχανής):

- Είναι πιο εύκολο στον άνθρωπο να απομνημονεύσει τις εντολές οι οποίες είναι συμβολικές λέξεις έναντι των εντολών της γλώσσας μηχανής οι οποίες είναι ακολουθίες δυαδικών ψηφίων.

*6.2.3 **Γλώσσες υψηλού επιπέδου 3ης γενιάς***

Γλώσσες υψηλού επιπέδου είναι γλώσσες προγραμματισμού που σαν εντολές έχουν φράσεις κατανοητές από τον άνθρωπο.

Μορφή του προγράμματος: Ένα πρόγραμμα σε ***γλώσσα υψηλού επιπέδου*** είναι μια ακολουθία εντολών γραμμένες με φράσεις κατανοητές από τον άνθρωπο που θυμίζουν πολύ την φυσική ανθρώπινη γλώσσα.

Πως εκτελεί ο υπολογιστής το πρόγραμμα: Οι εντολές μεταφράζονται σε εντολές της γλώσσας μηχανής, δηλαδή ακολουθίες δυαδικών ψηφίων και μετά εκτελούνται από τον υπολογιστή. Το έργο της μετάφρασης μπορεί να γίνει από δύο διαφορετικές κατηγορίες προγραμμάτων τους **διερμηνείς** (interpreters) ή τους **μεταγλωττιστές** (compilers).

Πλεονεκτήματα (σε σχέση με τις γλώσσες μηχανής και συμβολικές):

1. Ο **φυσικότερος και πιο "ανθρώπινος" τρόπος έκφρασης** των προβλημάτων. Τα προγράμματα σε γλώσσα υψηλού επιπέδου είναι πιο κοντά στα προβλήματα που επιλύουν.
2. Η **ανεξαρτησία από τον τύπο του υπολογιστή και την αρχιτεκτονική του**. Προγράμματα σε μία γλώσσα υψηλού επιπέδου μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή με ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της **μεταφερσιμότητας** των προγραμμάτων είναι σημαντικό προσόν.
3. Η **ευκολία της εκμάθησης και εκπαίδευσης** ως απόρροια των προηγουμένων.
4. Η **διόρθωση λαθών και η συντήρηση** προγραμμάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο.
5. Συνολικά οι γλώσσες υψηλού επιπέδου ελάττωσαν σημαντικά το **χρόνο και το κόστος παραγωγής** νέων προγραμμάτων, αφού λιγότεροι προγραμματιστές μπορούν σε μικρότερο χρόνο να αναπτύξουν προγράμματα που χρησιμοποιούνται σε περισσότερους υπολογιστές.

Με τον όρο **οπτικό προγραμματισμό** εννοούμε τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της εφαρμογής για παράδειγμα τα πλαίσια διαλόγου ή τα μενού.

Με τον όρο **οδηγούμενο από το γεγονός προγραμματισμό** εννοούμε τη δυνατότητα να ενεργοποιούνται λειτουργίες του προγράμματος με την εκτέλεση ενός γεγονότος, για παράδειγμα την επιλογή μίας εντολής από ένα μενού ή το κλικ του ποντικιού.

*6.2.4 **Γλώσσες 4ης γενιάς***

Οι γλώσσες υψηλού επιπέδου (γλώσσες 3ης γενιάς) γνώρισαν μεγάλη επιτυχία λόγω των πλεονεκτημάτων που παρουσιάζουν. Ωστόσο απευθύνονται μόνο σε προγραμματιστές. Ο χρήστης ενός υπολογιστή δεν είχε τη δυνατότητα να επιφέρει αλλαγές σε κάποιο πρόγραμμα, προκειμένου να ικανοποιήσει μια νέα ανάγκη του.
Σταδιακά όμως πολλές γλώσσες εφοδιάστηκαν με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο χρήστης μπορεί να επιλύει μόνος του μικρά προβλήματα εφαρμογών. Αυτή η αυξανόμενη τάση απόκρυψης της αρχιτεκτονικής του υλικού και της τεχνικής του προγραμματισμού οδήγησε στις γλώσσες 4ης γενιάς.
Στις γλώσσες αυτές ο χρήστης ενός υπολογιστή έχει τη δυνατότητα, σχετικά εύκολα, να υποβάλει ερωτήσεις στο σύστημα ή να αναπτύσσει εφαρμογές που ανακτούν πληροφορίες από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών.

Ποια είναι η καλύτερη γλώσσα προγραμματισμού;

Μπορούμε να ισχυριστούμε με βεβαιότητα ότι μία γλώσσα προγραμματισμού που να είναι αντικειμενικά καλύτερη από τις άλλες δεν υπάρχει, ούτε πρόκειται να υπάρξει. Σε διαφορετικούς τομείς διαφορετικές γλώσσες είναι καταλληλότερες.
Η επιλογή της γλώσσας για την ανάπτυξη μιας εφαρμογής εξαρτάται από το **είδος της εφαρμογής**, το **υπολογιστικό περιβάλλον** στο οποίο θα **εκτελεστεί**, τα **προγραμματιστικά περιβάλλοντα που διαθέτουμε** και κυρίως τις **γνώσεις του προγραμματιστή.**

Γλώσσα	**Τομείς εξειδίκευσης**	**Τύπος προγραμματισμού**	**Άλλα χαρακτηριστικά**
FORTRAN	Επιστημονικός	Διαδικασιακός αλγοριθμικός	
COMBOL	Εμπορικός	Διαδικασιακός αλγοριθμικός	
ALGOL	Γενικής χρήσης	Διαδικασιακός αλγοριθμικός	Ελάχιστη πρακτική εφαρμογή
PL/1	Γενικής χρήσης	Διαδικασιακός αλγοριθμικός	Χωρίς επιτυχία
LISP	Τεχνητή νοημοσύνη	Μη διαδικαστικός Συναρτησιακός	Πολύ υψηλού επιπέδου
PROLOG	Τεχνητή νοημοσύνη	Μη διαδικαστικός Δηλωτικός	Πολύ υψηλού επιπέδου
BASIC	Εκπαίδευση Γενικής χρήσης	Διαδικασιακός αλγοριθμικός	
PASCAL	Γενικής χρήσης Εκπαίδευση	Διαδικασιακός αλγοριθμικός	Δομημένος προγραμματισμός
ADA	Γενικής χρήσης	Διαδικασιακός αλγοριθμικός	Δομημένος προγραμματισμός
C	Προγραμματισμού συστημάτων	Διαδικασιακός αλγοριθμικός	Δομημένος προγραμματισμός UNIX
C++	Γενικής χρήσης Προγραμματισμού συστημάτων	Αντικειμενοστραφής	Λειτουργικά συστήματα
JAVA	Διαδίκτυο	Αντικειμενοστραφής Κατανεμημένος	
VISUAL BASIC	Γενικής χρήσης	Αντικειμενοστραφής Οδηγούμενος από γεγονός Οπτικός	Γραφικά περιβάλλοντα
VISUAL C++	Γενικής χρήσης Προγραμματισμού συστημάτων	Αντικειμενοστραφής Οδηγούμενος από γεγονός Οπτικός	Γραφικά περιβάλλοντα
SQL	Βάσεις δεδομένων	Γλώσσα ερωταπαντήσεων	4ης γενιάς

6.3 Φυσικές και τεχνητές γλώσσες

Οι γλώσσες προγραμματισμού χρησιμοποιούνται δηλαδή για την επικοινωνία του ανθρώπου και της μηχανής, όπως αντίστοιχα οι φυσικές γλώσσες χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων. Οι γλώσσες προγραμματισμού, πού είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες.

Μία γλώσσα προσδιορίζεται από το **αλφάβητό** της, το **λεξιλόγιό** της, τη **γραμματική** της και τέλος τη **σημασιολογία** της.

Το αλφάβητο

Αλφάβητο μίας γλώσσας καλείται το **σύνολο** των **στοιχείων** που χρησιμοποιείται από τη γλώσσα.

Το λεξιλόγιο

Το λεξιλόγιο αποτελείται από ένα **υποσύνολο** όλων των **ακολουθιών** που δημιουργούνται από τα **στοιχεία** του **αλφαβήτου**, που είναι **δεκτές** από την γλώσσα και ονομάζονται **λέξεις**.

Η Γραμματική

Η Γραμματική αποτελείται από το **τυπικό** ή **τυπολογικό** (accidence) και το **συντακτικό** (syntax).

- **Τυπικό** είναι το **σύνολο των κανόνων** που ορίζει τις **μορφές** με τις οποίες μία **λέξη** είναι αποδεκτή.
- **Συντακτικό** είναι το **σύνολο των κανόνων** που καθορίζει τη νομιμότητα της **διάταξης** και της **σύνδεσης** των **λέξεων** της γλώσσας για τη δημιουργία **προτάσεων** (στις φυσικές γλώσσες) ή **εντολών** (στις τεχνητές γλώσσες του προγραμματισμού).

Η σημασιολογία

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

Στις γλώσσες προγραμματισμού οι οποίες είναι τεχνητές γλώσσες, ο δημιουργός της γλώσσας αποφασίζει τη σημασιολογία των λέξεων της γλώσσας.

Διαφορές φυσικών και τεχνητών γλωσσών.

Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η **δυνατότητα εξέλιξής τους**.

Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημιουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν με την πάροδο του χρόνου και αυτό γιατί η γλώσσα χρησιμοποιείται για την επικοινωνία μεταξύ ανθρώπων, που εξελίσσονται.

Αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα, αφού κατασκευάζονται συνειδητά για ένα συγκεκριμένο σκοπό. Ωστόσο συχνά οι γλώσσες προγραμματισμού βελτιώνονται και μεταβάλλονται από τους δημιουργούς τους, με σκοπό να διορθωθούν αδυναμίες ή να καλύψουν μεγαλύτερο εύρος εφαρμογών ή τέλος να ακολουθήσουν τις νέες εξελίξεις. Οι γλώσσες προγραμματισμού αλλάζουν σε επίπεδο **διαλέκτου** (για παράδειγμα GW-Basic και QuickBasic) ή σε επίπεδο **επέκτασης** (για παράδειγμα Basic και Visual Basic).

6.4 Τεχνικές σχεδίασης προγραμμάτων

6.4.1 ***Ιεραρχική σχεδίαση προγράμματος***

Η τεχνική της **ιεραρχικής σχεδίασης** και επίλυσης ή η διαδικασία σχεδίασης "**από επάνω προς τα κάτω**" όπως συχνά ονομάζεται (top-down program design) περιλαμβάνει τον καθορισμό των **βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο**, και στη συνέχεια τη **διάσπαση των λειτουργιών** αυτών σε όλο και μικρότερες λειτουργίες, μέχρι το τελευταίο επίπεδο που οι λειτουργίες είναι πολύ απλές, ώστε να επιλυθούν εύκολα.
Σκοπός της ιεραρχικής σχεδίασης είναι η **διάσπαση λοιπόν του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα**, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.
Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες **διαγραμματικές τεχνικές**.

6.4.2 Τμηματικός προγραμματισμός

Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον τμηματικό προγραμματισμό. Μετά την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα (module), που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.
Ο τμηματικός προγραμματισμός **διευκολύνει τη δημιουργία** του προγράμματος, **μειώνει τα λάθη** και επιτρέπει την **ευκολότερη παρακολούθηση**, **κατανόηση** και **συντήρηση** του προγράμματος από τρίτους.

6.4.3 ***Δομημένος προγραμματισμός***

Ο **δομημένος προγραμματισμός** στηρίζεται στη χρήση **τριών** και μόνο στοιχειωδών λογικών δομών, τη **δομή της ακολουθίας**, τη **δομή της επιλογής** και τη **δομή της επανάληψης**. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο **μία είσοδο** και μόνο **μία έξοδο**.
Αν και ο δομημένος προγραμματισμός αρχικά εμφανίστηκε σαν μία προσπάθεια **περιορισμού των εντολών GOTO**, σήμερα αποτελεί τη **βασική μεθοδολογία προγραμματισμού**. είναι μία μεθοδολογία σύνταξης προγραμμάτων που έχει σκοπό να βοηθήσει τον προγραμματιστή στην **ανάπτυξη σύνθετων προγραμμάτων**, να **μειώσει τα λάθη**, να εξασφαλίσει την **εύκολη κατανόηση** των

προγραμμάτων και να **διευκολύνει τις διορθώσεις και τις αλλαγές** σε αυτά.
Ο δομημένος προγραμματισμός ενθαρρύνει και βοηθάει την ανάλυση του προγράμματος σε επί μέρους τμήματα, έτσι ώστε σήμερα ο όρος **δομημένος προγραμματισμός περιέχει** τόσο την **ιεραρχική σχεδίαση** όσο και τον **τμηματικό προγραμματισμό**.

Πλεονεκτήματα του δομημένου προγραμματισμού:

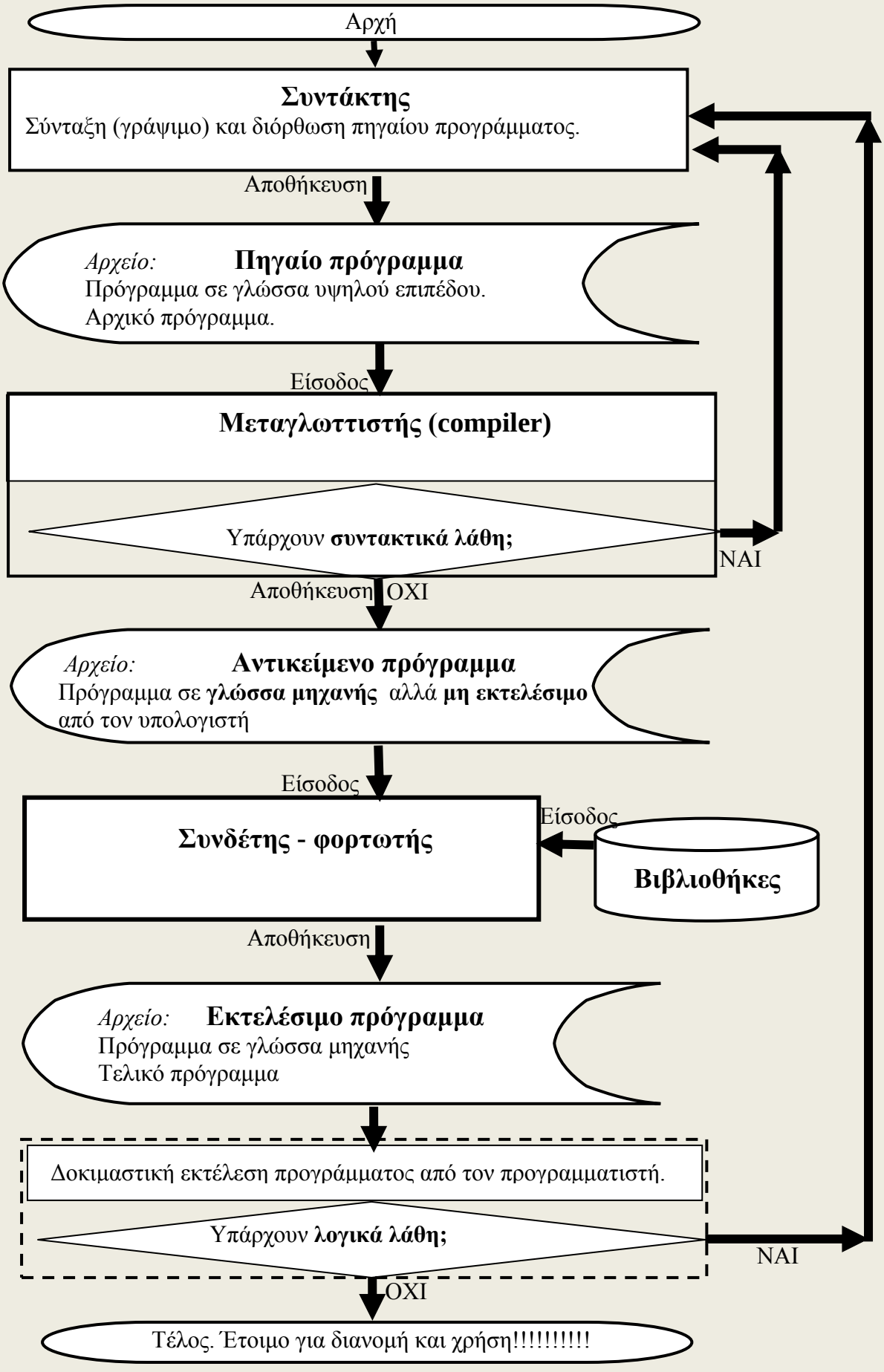
1. Δημιουργία **απλούστερων** προγραμμάτων.
2. Άμεση μεταφορά των **αλγορίθμων** σε **προγράμματα**.
3. Διευκόλυνση **ανάλυσης** του προγράμματος σε τμήματα.
4. Περιορισμός των **λαθών** κατά την ανάπτυξη του προγράμματος.
5. Διευκόλυνση στην ανάγνωση και **κατανόηση** του προγράμματος από τρίτους.
6.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

6.7 Προγραμματιστικά περιβάλλοντα

Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού, πρέπει να **μετατραπεί** σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές **γλώσσας μηχανής**.
Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν 2 μεγάλες κατηγορίες τέτοιων προγραμμάτων, οι **μεταγλωττιστές** (compilers) και οι **διερμηνευτές** (interpreters).

Μεταγλωττιστές.
Ο μεταγλωττιστής δέχεται στην **είσοδο** ένα **πρόγραμμα** σε μια **γλώσσα υψηλού επιπέδου** και παράγει ένα **ισοδύναμο πρόγραμμα σε γλώσσα μηχανής**. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι **τελείως ανεξάρτητο από το αρχικό** πρόγραμμα.
Για την αρχική σύνταξη των προγραμμάτων και τη διόρθωσή τους στη συνέχεια χρησιμοποιείται ένα ειδικό πρόγραμμα που ονομάζεται **συντάκτης** (editor). Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου, με δυνατότητες όμως που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων.
Το αρχικό πρόγραμμα λέγεται **πηγαίο** (source), ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται **αντικείμενο** πρόγραμμα (object). Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά συνήθως **δεν είναι σε θέση να εκτελεστεί**. Χρειάζεται να **συμπληρωθεί** και να **συνδεθεί** με άλλα **τμήματα προγράμματος** απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις **βιβλιοθήκες** (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται **συνδέτης – φορτωτής** (linker-loader).

Το αποτέλεσμα του συνδέτη είναι η παραγωγή του **εκτελέσιμου προγράμματος** (executable), το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται μεταγλώττιση και σύνδεση.



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

Λάθη στον προγραμματισμό

Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση, που το **αρχικό πρόγραμμα δεν περιέχει (συντακτικά) λάθη**. Τις περισσότερες φορές κάθε πρόγραμμα αρχικά θα έχει λάθη. Τα λάθη του προγράμματος είναι γενικά δύο ειδών, **λογικά** και **συντακτικά**.

Συντακτικά λάθη:

- Τα συντακτικά λάθη **οφείλονται** σε **ανορθογραφία** των ονομάτων εντολών, **παράληψη δήλωσης** δεδομένων, παράληψη **λέξεων** σε μια φράση εντολής κ.λ.π. και πρέπει πάντα να διορθωθούν, ώστε να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.
- Τα συντακτικά λάθη **εμφανίζονται** στο στάδιο της **μεταγλώττισης**.
- Τα συντακτικά λάθη **ανιχνεύονται** από τον **μεταγλωττιστή** ή τον **διερμηνευτή** οι οποίοι εμφανίζουν κατάλληλα διαγνωστικά μηνύματα.

Το στάδιο που ακολουθεί είναι η **διόρθωση** των λαθών. Το διορθωμένο πρόγραμμα επαναϋποβάλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαλειφθούν πλήρως όλα τα συντακτικά λάθη.

Λογικά λάθη:

- Τα λογικά λάθη **οφείλονται** σε **σφάλματα** κατά την υλοποίηση του **αλγορίθμου** και είναι τα πλέον σοβαρά και δύσκολα στη διόρθωση τους.
- Τα λογικά λάθη **εμφανίζονται** στο στάδιο της **εκτέλεσης** του προγράμματος.
- Τα λογικά λάθη **ανιχνεύονται** από τον **άνθρωπο (προγραμματιστή** ή **χρήστη)** κατά το στάδιο δοκιμαστικών ή πραγματικών εκτελέσεων αντίστοιχα.

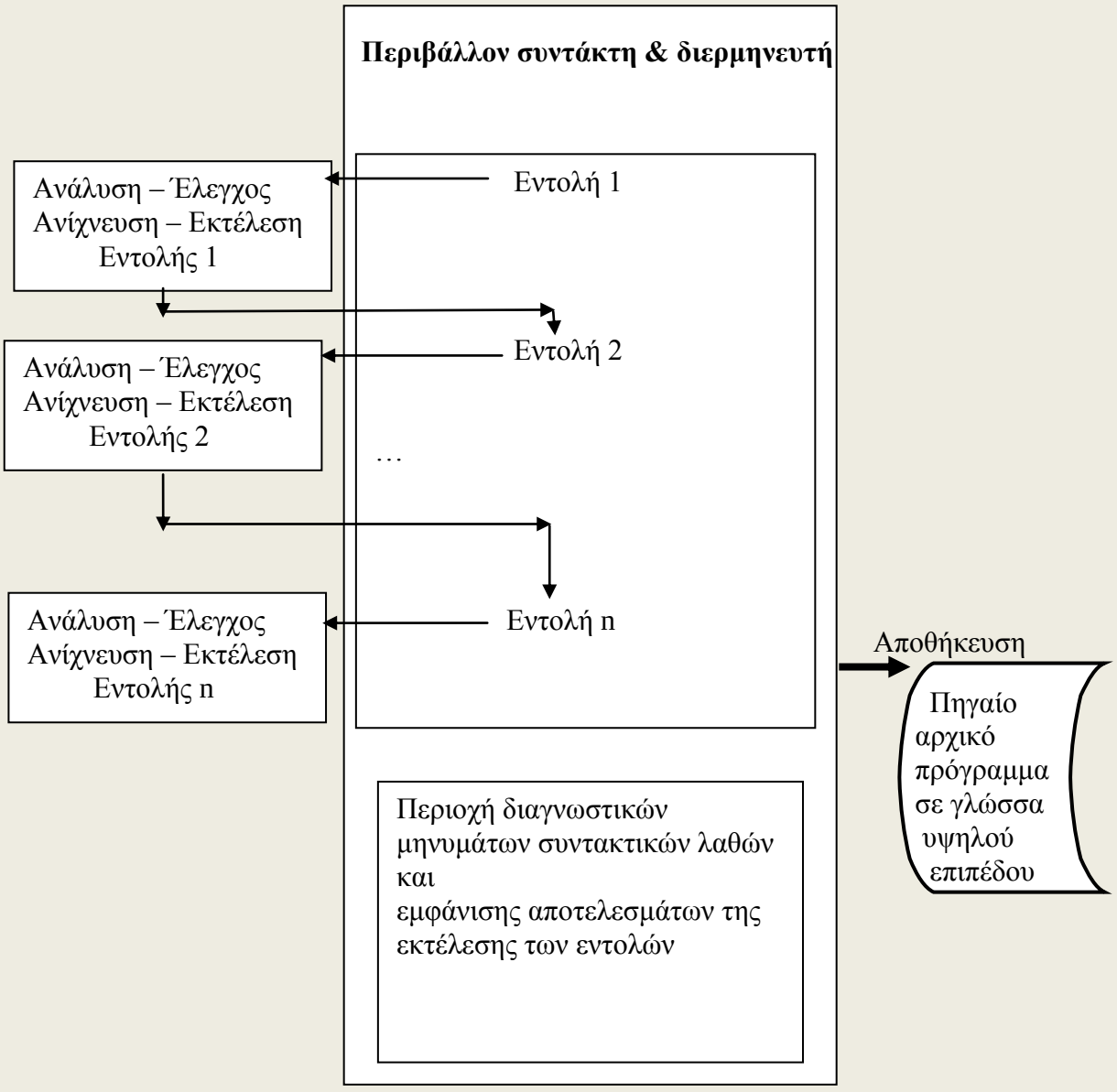
Το στάδιο που ακολουθεί τις δοκιμαστικές εκτελέσεις είναι η **διόρθωση** των λογικών λαθών. Το διορθωμένο πρόγραμμα επαναϋποβάλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαλειφθούν πλήρως όλα τα λογικά λάθη.

Διερμηνευτές

Αντίθετα με τον μεταγλωττιστή όπου πρώτα μεταφράζονται όλες οι εντολές και μετά εκτελούνται όλες μαζί, ο διερμηνευτής διαβάζει **μία προς μία τις εντολές του αρχικού προγράμματος** και **για κάθε μια εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής**.

Στη περίπτωση του διερμηνευτή, **συντάκτης και διερμηνευτής** αποτελούν πάντα ένα **ενιαίο πρόγραμμα** σε ένα **ενιαίο περιβάλλον** μέσα στο οποίο γίνεται και η εκτέλεση των εντολών και εμφανίζονται τα αποτελέσματα της εκτέλεσης.

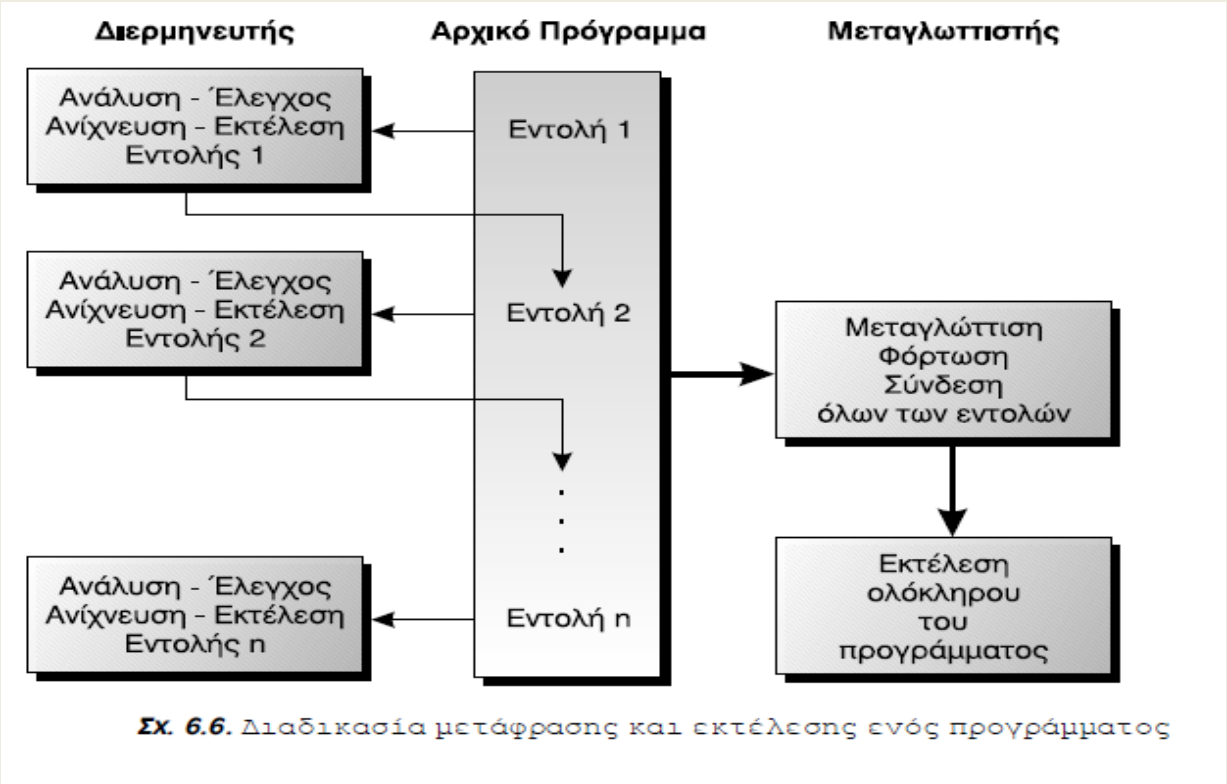
Το μόνο πρόγραμμα το οποίο **αποθηκεύεται** σε αρχείο στη περίπτωση του διερμηνευτή είναι το **αρχικό πηγαίο πρόγραμμα** γραμμένο στη γλώσσα υψηλού επιπέδου.



Σύγκριση πλεονεκτήματα μειονεκτήματα μεταγλωττιστή – διερμηνευτή

- Στη φάση **ανάπτυξης** του προγράμματος όπου γίνονται και οι έλεγχοι για την διόρθωση των λαθών (συντακτικών και λογικών):
  Η χρήση **μεταγλωττιστή** έχει το **μειονέκτημα**, ότι προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης.
  Από την άλλη μεριά η χρήση **διερμηνευτή** έχει το **πλεονέκτημα** της άμεσης εκτέλεσης και συνεπώς και της άμεσης διόρθωσης.

- Στη φάση **εκτέλεσης** του προγράμματος:
  Η χρήση του **μεταγλωττιστή** έχει το **πλεονέκτημα** της πολύ πιο **γρήγορης εκτέλεσης**, καθώς επίσης και το πλεονέκτημα ότι το τελικό εκτελέσιμο πρόγραμμα μπορεί να **διανεμηθεί και να εκτελεσθεί σε άλλους υπολογιστές** οι οποίοι **δεν διαθέτουν τον συντάκτη, μεταφραστή και συνδέτη** της γλώσσας προγραμματισμού.
  Η χρήση του **διερμηνευτή** έχει το **μειονέκτημα** της **αργής εκτέλεσης**, καθώς επίσης και το μειονέκτημα ότι η **διανομή και εκτέλεση** του προγράμματος σε άλλους υπολογιστές **απαιτεί να διαθέτουν και εκείνοι τον διερμηνευτή** της γλώσσας προγραμματισμού.



Σχ. 6.6. Διαδικασία μετάφρασης και εκτέλεσης ενός προγράμματος

Άσκηση 1 ΘΕΜΑ 1ο Γ 2002, ΘΕΜΑ 1ο ΣΤ 2003
Άσκηση 2 ΘΕΜΑ 1ο Α 2004 , ΘΕΜΑ 1ο Ε 2005
Άσκηση 3 ΘΕΜΑ 1ο Β1, Δ 2007
Άσκηση 4 ΘΕΜΑ 1ο Β1, Β2 2008

ΚΕΦΑΛΑΙΟ 7

7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ

Το αλφάβητο της **ΓΛΩΣΣΑΣ** αποτελείται από τα γράμματα του ελληνικού και του λατινικού αλφαβήτου, τα ψηφία, καθώς και από ειδικά σύμβολα, που χρησιμοποιούνται για προκαθορισμένες ενέργειες.
Συγκεκριμένα

Γράμματα

Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)
Πεζά ελληνικού αλφαβήτου (α-ω)
Κεφαλαία λατινικού αλφαβήτου (A-Z)
Πεζά λατινικού αλφαβήτου (a-z)

Ψηφία

0-9

Ειδικοί χαρακτήρες

+ - * / = ^ ( ) . , ‘ ! & κενός χαρακτήρας

7.2 Τύποι δεδομένων

Οι τύποι δεδομένων που υποστηρίζει η **ΓΛΩΣΣΑ** είναι οι **αριθμητικοί**, που περιλαμβάνουν τους **ακέραιους** και τους **πραγματικούς** αριθμούς, οι **χαρακτήρες** και τέλος οι **λογικοί**.

- **Ακέραιος τύπος**. Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι θετικοί, αρνητικοί ή μηδέν. Παραδείγματα ακεραίων είναι οι αριθμοί 1, 3409, 0, -980.
- **Πραγματικός τύπος**. Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι αριθμοί 3.14159, 2.71828, -112.45, 0.45 είναι πραγματικοί αριθμοί. Και οι πραγματικοί αριθμοί μπορούν να είναι θετικοί, αρνητικοί ή μηδέν.
- **Χαρακτήρας**. Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και μία σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου μπορούν να περιέχουν οποιοδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο. Παραδείγματα χαρακτήρων είναι ‘Κ’, ‘Κώστας’, ‘σήμερα είναι Τετάρτη’, ‘Τα πολλαπλάσια του 15 είναι’, '25'.
  Οι χαρακτήρες πρέπει υποχρεωτικά να βρίσκονται μέσα σε απλά εισαγωγικά, ‘ ‘. Τα δεδομένα αυτού του τύπου, επειδή περιέχουν τόσο αλφαβητικούς όσο και αριθμητικούς χαρακτήρες, ονομάζονται συχνά **αλφαριθμητικά**.
- **Λογικός**. Αυτός ο τύπος δέχεται μόνο δύο τιμές ΑΛΗΘΗΣ και ΨΕΥΔΗΣ. Οι τιμές αντιπροσωπεύουν αληθείς ή ψευδείς συνθήκες.

7.3 Σταθερές

Οι σταθερές (constants) είναι προκαθορισμένες τιμές που δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι σταθερές είναι αντίστοιχου τύπου δεδομένων, δηλαδή ακέραιες, πραγματικές, αλφαριθμητικές ή λογικές. Π.χ. 5, 3.14, 'Κώστας', Αληθής.

Συμβολικές σταθερές

Η **ΓΛΩΣΣΑ** επιτρέπει την αντιστοίχιση σταθερών τιμών με ονόματα, εφόσον αυτά δηλωθούν στην αρχή του προγράμματος (**στο τμήμα δήλωσης σταθερών**, βλέπε παρακάτω).

Σύνταξη

ΣΤΑΘΕΡΕΣ

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

Παραδείγματα

ΣΤΑΘΕΡΕΣ

ΠΙ=3.14
ΦΠΑ=0.18
g=9,81

Λειτουργία

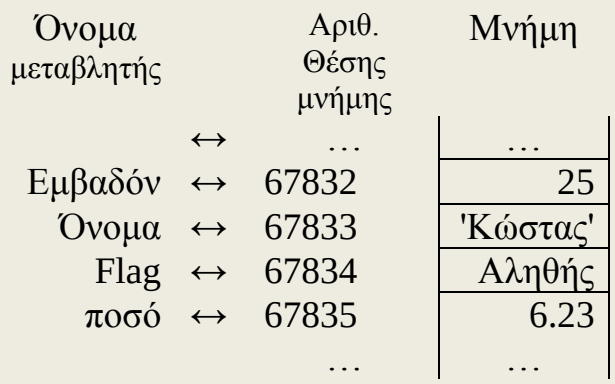
Αποδίδει ονόματα σε σταθερές τιμές. Κάθε ένα από αυτά τα ονόματα μπορεί να χρησιμοποιηθεί οπουδήποτε στο πρόγραμμα, αλλά δεν είναι δυνατή η μεταβολή της τιμής κατά τη διάρκεια εκτέλεσης του προγράμματος.

Τις συμβολικές σταθερές τις χρησιμοποιούμε σε τιμές που δεν πρόκειται να αλλάξουν κατά την διάρκεια εκτέλεσης του προγράμματος, όπως π.χ. το π, για 3 λόγους:

1. Γίνετε πιο ευανάγνωστο το πρόγραμμα.
2. Κάνουμε οικονομία στο γράψιμο. Αν π.χ. το π χρειάζεται να το χρησιμοποιήσουμε 15 φορές μέσα στο πρόγραμμα, άλλο είναι να γράφουμε 15 φορές π, και άλλο 15 φορές 3.14.
3. Γίνονται πιο εύκολα και με μικρότερο κίνδυνο για λάθη αλλαγές στο πρόγραμμα. Π.χ. έστω ότι δεν χρησιμοποιήσαμε συμβολική σταθερά και χρησιμοποιήσαμε 15 φορές το 3.14 μέσα στο πρόγραμμα. Αν μετά αποφασίσουμε ότι τελικά θέλουμε μεγαλύτερη ακρίβεια στους υπολογισμούς και θέλουμε αντί για 3.15 να χρησιμοποιήσουμε την τιμή 3.14159, τότε και 15 αλλαγές θα πρέπει να κάνουμε και κινδυνεύουμε να μην προσέξουμε και να μην αλλάξουμε κάποια τιμή. Αντίθετα με την χρήση συμβολικής σταθεράς χρειάζεται μια μόνο αλλαγή.

7.4 Μεταβλητές

Μια μεταβλητή λοιπόν, παριστάνει μία ποσότητα που η τιμή της μπορεί να μεταβάλλεται.



Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα, αντιστοιχούνται από το μεταγλωττιστή σε συγκεκριμένες θέσεις μνήμης του υπολογιστή. Η τιμή της μεταβλητής είναι η τιμή που βρίσκεται στην αντίστοιχη θέση μνήμης και όπως αναφέρθηκε μπορεί να μεταβάλλεται κατά τη διάρκεια της εκτέλεσης του προγράμματος.

Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, αυτό που μένει υποχρεωτικά αναλλοίωτο είναι ο τύπος της μεταβλητής.
Η **ΓΛΩΣΣΑ** επιτρέπει τη χρήση μεταβλητών των τεσσάρων τύπων που αναφέρθηκαν, δηλαδή ακεραίων, πραγματικών, χαρακτήρων και λογικών ενώ η δήλωση του τύπου κάθε μεταβλητής γίνεται υποχρεωτικά στο **τμήμα δήλωσης μεταβλητών**.

Σύνταξη
ΜΕΤΑΒΛΗΤΕΣ
τύπος-1*: Λίστα-μεταβλητών-1*
τύπος-2*: Λίστα-μεταβλητών-2*
…
Τύπος-ν: *Λίστα-μεταβλητών-ν*

Παραδείγματα
ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: Εμβαδόν, Α
ΑΚΕΡΑΙΕΣ: ΤΙΜΗ, Ν
ΧΑΡΑΚΤΗΡΕΣ: Όνομα
ΛΟΓΙΚΕΣ: Έλεγχος

Λειτουργία
Δηλώνει τον τύπο όλων των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα.

Το όνομα κάθε μεταβλητής, (όπως και των συμβολικών σταθερών και τα ονόματα των προγραμμάτων) ακολουθεί τους **κανόνες δημιουργίας ονομάτων:**
Κάθε **πρόγραμμα**, καθώς και τα **δεδομένα** που χρησιμοποιεί (**συμβολικές σταθερές** και **μεταβλητές**) έχουν ένα **όνομα**, με το οποίο αναφερόμαστε σε αυτά. Τα ονόματα αυτά μπορούν να αποτελούνται από **γράμματα** πεζά ή κεφαλαία του ελληνικού ή του λατινικού αλφαβήτου (Α-

Ω, Α-Ζ), **ψηφία** (0-9) καθώς και τον **χαρακτήρα κάτω παύλα** (underscore) (_), ενώ πρέπει υποχρεωτικά να **αρχίζουν με γράμμα**.
Επειδή μερικές λέξεις χρησιμοποιούνται από την ίδια τη **ΓΛΩΣΣΑ** για συγκεκριμένους λόγους, όπως οι λέξεις ΠΡΟΓΡΑΜΜΑ, ΑΚΕΡΑΙΟΣ, ΠΡΑΓΜΑΤΙΚΟΣ, ΑΝ κ.λπ, αυτές οι λέξεις **δεν μπορούν να χρησιμοποιηθούν** ως ονόματα. Οι λέξεις αυτές αποκαλούνται **δεσμευμένες**.
Παραδείγματα ονομάτων που είναι αποδεκτά από τη **ΓΛΩΣΣΑ** είναι: Α, Όνομα, Τιμή, Τυπική_Απόκλιση, Α100, ΦΠΑ, μέγιστο.
Παραδείγματα ονομάτων που δεν είναι αποδεκτά είναι: 100Α, Μέση Τιμή, Κόστος$, Για, Αν, τότε.

7.5 Αριθμητικοί τελεστές

Οι αριθμητικοί τελεστές που υποστηρίζονται από τη **ΓΛΩΣΣΑ** καλύπτουν τις βασικές πράξεις. Οι πράξεις και οι αντίστοιχοι αριθμητικοί τελεστές είναι:

Αριθμητικός τελεστής	Πράξη
^	Ύψωση σε δύναμη
/	Διαίρεση
DIV	Ακέραια διαίρεση
MOD	Υπόλοιπο ακέραιας διαίρεσης
*	Πολλαπλασιασμός
+	Πρόσθεση
-	Αφαίρεση

7.6 Συναρτήσεις

Πολλές γνωστές συναρτήσεις από τα μαθηματικά χρησιμοποιούνται συχνά και περιέχονται στη **ΓΛΩΣΣΑ**. Οι συναρτήσεις αυτές είναι:

ΗΜ(Χ) Υπολογισμός ημίτονου
ΣΥΝ(Χ) Υπολογισμός συνημίτονου
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

Όταν μια **τιμή** προκύπτει από **υπολογισμό**, τότε αναφερόμαστε σε **εκφράσεις** (expressions). Για τη **σύνταξη** μιας αριθμητικής έκφρασης χρησιμοποιούνται **αριθμητικές σταθερές**, **μεταβλητές**, **συναρτήσεις**, **αριθμητικοί τελεστές** και **παρενθέσεις**. Οι αριθμητικές εκφράσεις **υλοποιούν** απλές ή σύνθετες **μαθηματικές πράξεις**.
Κάθε έκφραση παριστάνει μια συγκεκριμένη αριθμητική τιμή, η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι απαραίτητο όλες οι μεταβλητές, που εμφανίζονται σε μια έκφραση να έχουν οριστεί προηγούμενα, δηλαδή να έχουν ήδη κάποια τιμή.

Ιεραρχία

1. Ύψωση σε δύναμη (^)
2. Πολλαπλασιασμός και διαίρεση (*, / , DIV, MOD)
3. Πρόσθεση και αφαίρεση (+ , -)

Όταν η ιεραρχία είναι ίδια, τότε οι πράξεις εκτελούνται από τ' αριστερά προς τα δεξιά. Σε πολλές όμως περιπτώσεις είναι απαραίτητο να προηγηθεί μια πράξη χαμηλότερης ιεραρχίας. Αυτό επιτυγχάνεται με την **εισαγωγή των παρενθέσεων**.

7.8 Εντολή εκχώρησης

Η εντολή εκχώρησης χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.

Σύνταξη

Όνομα-Μεταβλητής **<-** *έκφραση*

Παραδείγματα

```
Α <- 132
ΜΗΝΑΣ <- 'Ιανουάριος'
ΕΜΒΑΔΟΝ <- Α*Β
```

Λειτουργία

Υπολογίζεται η τιμή της έκφρασης στη δεξιά πλευρά και εκχωρείται η τιμή αυτή στη μεταβλητή, που αναφέρεται στην αριστερή πλευρά.

7.9 Εντολές εισόδου-εξόδου

Σύνταξη
ΔΙΑΒΑΣΕ λίστα-μεταβλητών

Παραδείγματα
ΔΙΑΒΑΣΕ Ποσότητα, Τιμή

Λειτουργία
Η εκτέλεση της εντολής οδηγεί στην είσοδο τιμών από το πληκτρολόγιο και την εκχώρηση τους στις μεταβλητές που αναφέρονται.

Σύνταξη
ΓΡΑΨΕ *λίστα-στοιχείων*

Παραδείγματα
ΓΡΑΨΕ 'Η τετραγωνική ρίζα του', Α,' είναι: ',ΡΙΖΑ

Λειτουργία
Χρησιμοποιείται για την εμφάνιση σταθερών τιμών καθώς και των τιμών των μεταβλητών που αναφέρονται στη λίστα.

7.10 Δομή προγράμματος

Η πρώτη εντολή κάθε προγράμματος είναι υποχρεωτικά η επικεφαλίδα του προγράμματος, η οποία είναι η λέξη **ΠΡΟΓΡΑΜΜΑ** ακολουθούμενη από το όνομα του προγράμματος.
Στη συνέχεια ακολουθεί το **τμήμα δήλωσης των συμβολικών σταθερών** του προγράμματος, (αν υπάρχουν).
Αμέσως μετά είναι το τμήμα **δήλωσης μεταβλητών**, όπου δηλώνονται υποχρεωτικά τα **ονόματα** όλων των μεταβλητών καθώς και ο **τύπος** τους.
Ακολουθεί το κύριο μέρος του προγράμματος, που περιλαμβάνει όλες τις **εκτελέσιμες εντολές**. Οι εντολές αυτές περιλαμβάνονται υποχρεωτικά ανάμεσα στις λέξεις **ΑΡΧΗ** και **ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ**.
Τέλος αν το πρόγραμμα χρησιμοποιεί **διαδικασίες** (βλ. κεφ. 10), αυτές γράφονται **μετά** το **ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ**.
Κάθε εντολή γράφεται σε **ξεχωριστή γραμμή**. Αν μία εντολή πρέπει να **συνεχιστεί** και στην **επόμενη γραμμή**, τότε ο πρώτος χαρακτήρας αυτής της γραμμής πρέπει να είναι ο **χαρακτήρας &**.
Αν ο πρώτος χαρακτήρας είναι το **θαυμαστικό** (!), σημαίνει ότι αυτή η γραμμή περιέχει **σχόλια** και όχι εκτελέσιμες εντολές.

Παράδειγμα
Το επόμενο πρόγραμμα υπολογίζει το συνολικό κόστος παραγγελιών υπολογιστών. Το πρόγραμμα διαβάζει από το πληκτρολόγιο την ποσότητα της παραγγελίας και την τιμή του ενός υπολογιστή, υπολογίζει και γράφει το συνολικό κόστος καθώς και το αντίστοιχο κόστος του ΦΠΑ. Ο συντελεστής ΦΠΑ είναι 18%.

ΠΡΟΓΡΑΜΜΑ Κόστος_Υπολογιστών
ΣΤΑΘΕΡΕΣ
ΦΠΑ=0.18
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: Ποσότητα, Τιμή_μονάδας, Κόστος
ΠΡΑΓΜΑΤΙΚΕΣ: Αξία_ΦΠΑ, Συνολικό_κόστος
ΑΡΧΗ
! Εισαγωγή δεδομένων
ΓΡΑΨΕ 'Δώσε την ποσότητα της παραγγελίας'
ΔΙΑΒΑΣΕ Ποσότητα
ΓΡΑΨΕ 'Δώσε την τιμή του υπολογιστή'
ΔΙΑΒΑΣΕ Τιμή_μονάδας
! Υπολογισμοί
Κόστος <- Ποσότητα* Τιμή_μονάδας
Αξία_ΦΠΑ <- Κόστος*ΦΠΑ
Συνολικό_κόστος <- Κόστος+Αξία_ΦΠΑ
! Εμφάνιση αποτελεσμάτων
ΓΡΑΨΕ 'Το κόστος των',Ποσότητα,'υπολογ. είναι ',Κόστος
ΓΡΑΨΕ ' Η αξία του ΦΠΑ είναι', Αξία_ΦΠΑ
ΓΡΑΨΕ 'Το συνολικό κόστος είναι', Συνολικό_κόστος
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Κόστος_Υπολογιστών

Άσκηση 1 ΘΕΜΑ 1ο Γ 2001, ΘΕΜΑ 1ο Δ 2005
Άσκηση 2 ΘΕΜΑ 1ο Α,Β επαναλ 2001
Άσκηση 3 ΘΕΜΑ 4ο (ΜΕ ΠΡΌΓΡΑΜΜΑ) ΕΣΠΕΡ 2000

Άσκηση 4: Να αντιστοιχίσετε τα παρακάτω:

1	ΛΟΓΙΚΗ ΣΤΑΘΕΡΑ	Α	5
2	ΧΑΡΑΚΤΗΡΑΣ	Β	'ΑΛΗΘΗΣ'
3	ΑΚΕΡΑΙΑ ΣΤΑΘΕΡΑ	Γ	DIV
4	ΠΡΑΓΜΑΤΙΚΗ ΣΤΑΘΕΡΑ	Δ	'58.76'
5	ΑΡΙΘΜΗΤΙΚΟΣ ΤΕΛΕΣΤΗΣ	Ε	'MOD'
		Ζ	ΨΕΥΔΗΣ
		Η	+
		Θ	'-'
			'ΚΩΣΤΑΣ'

(Περισσότερα από ένα ή και κανένα της 2ης στήλης μπορούν να αντιστοιχηθούν στην πρώτη.)

Άσκηση 5: Τι τύπου μεταβλητές πρέπει να χρησιμοποιήσετε για τα παρακάτω στοιχεία; Γράψετε το αντίστοιχο τμήμα δηλώσεων.
1. Το όνομα ενός μαθητή.
2. Ο αριθμός μαθητολογίου του μαθητή.
3. Τη βαθμολογία του μαθητή.
4. Το τηλέφωνο ενός μαθητή.
5. Τη διεύθυνση ενός μαθητή.
6. Το φύλο ενός μαθητή (πώς μπορεί να οριστεί με χρήση λογικής μεταβλητής;)

Άσκηση 6:
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει έναν αριθμό δευτερολέπτων και θα τον μετατρέπει σε ημέρες, ώρες, λεπτά και δευτερόλεπτα. Π.χ. 442930 δευτερόλεπτα → 5 ημέρες, 3 ώρες, 2 λεπτά, 10 δευτερόλεπτα.

Άσκηση 7:
Να γράψετε ένα πρόγραμμα που θα διαβάζει ένα ποσό σε Ευρώ και θα βρίσκει τον ελάχιστο αριθμό χαρτονομισμάτων και κερμάτων που θα το αντιπροσωπεύουν. Χαρτονομίσματα (50,20,10,5) κέρματα (2,1).
Π.χ. 434^{E}→ 8 των 50 + 1 των 20 + 1 των 10 + 2 των 2 + 0 του 1.

ΚΕΦΑΛΑΙΟ 8

8.1 Εντολές Επιλογής

Λογική Έκφραση

Για τη σύνταξη μιας **λογικής έκφρασης** ή **συνθήκης** χρησιμοποιούνται **σταθερές**, **μεταβλητές**, **αριθμητικές παραστάσεις**, **συγκριτικοί** και **λογικοί τελεστές**, καθώς και **παρενθέσεις**. Στις λογικές εκφράσεις γίνεται σύγκριση της τιμής μίας έκφρασης, που βρίσκεται αριστερά από το συγκριτικό τελεστή με την τιμή μιας άλλης έκφρασης που βρίσκεται δεξιά. Το αποτέλεσμα είναι μία λογική τιμή **ΑΛΗΘΗΣ** ή **ΨΕΥΔΗΣ**.

Συγκριτικοί τελεστές
=
<>
>
>=
<
<=

Οι συγκρίσεις γίνονται σε δεδομένα αριθμητικά, αλφαριθμητικά και λογικά.

- Η σύγκριση μεταξύ δύο **αριθμών** γίνεται με προφανή τρόπο.
- Η σύγκριση **ατομικών χαρακτήρων** στηρίζεται στην αλφαβητική σειρά, για παράδειγμα το ‘α’ θεωρείται μικρότερο από το ‘β’.
- Η σύγκριση **αλφαριθμητικών δεδομένων** βασίζεται στη σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρις ότου βρεθεί κάποια διαφορά, για παράδειγμα η λέξη ‘κακός’ θεωρείται μικρότερη από τη λέξη ‘καλός’ αφού το γράμμα κ προηγείται του γράμματος λ.
- Η σύγκριση **λογικών** έχει έννοια μόνο στην περίπτωση του ίσου (=) και του διάφορου (<>), αφού οι τιμές που μπορούν να έχουν είναι ΑΛΗΘΗΣ και ΨΕΥΔΗΣ.

Σύνθετες Εκφράσεις

Σε πολλά προβλήματα οι επιλογές δεν αρκεί να γίνονται με απλές λογικές παραστάσεις όπως αυτές οι οποίες αναφέρθηκαν, αλλά χρειάζεται να συνδυαστούν μία ή περισσότερες λογικές παραστάσεις. Αυτό επιτυγχάνεται με τη χρήση των τριών βασικών λογικών τελεστών ΟΧΙ, ΚΑΙ, ‘Η.

Παραδείγματα

0<Χ<5	Χ>0 ΚΑΙ Χ<5
Χ=1 ή 2 ή 3	Χ=1 ‘Η Χ=2 ‘Η Χ=3

Η **ιεραρχία** των **λογικών** τελεστών είναι **μικρότερη** των **αριθμητικών**.

8.1.1 Εντολή ΑΝ

Η δομή επιλογής υλοποιείται στη **ΓΛΩΣΣΑ** με την εντολή **ΑΝ**. Η εντολή **ΑΝ** εμφανίζεται με τρεις διαφορετικές μορφές.

- Την απλή εντολή **ΑΝ...ΤΟΤΕ**,
- την εντολή **ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ** και τέλος
- την εντολή **ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ_ΑΝ.**

Κάθε εντολή **ΑΝ** πρέπει να κλείνει με **ΤΕΛΟΣ_ΑΝ**.

Σύνταξη

ΑΝ *συνθήκη* **ΤΟΤΕ**
εντολή-1
εντολή-2
…
εντολή-ν
ΤΕΛΟΣ_ΑΝ

Παράδειγμα

ΑΝ *αριθμός > 0* **ΤΟΤΕ**
ΓΡΑΨΕ *‘Ο αριθμός είναι θετικός’*
Πλήθος_θετικών <- Πλήθος_θετικών+1
ΤΕΛΟΣ_ΑΝ

Λειτουργία

Αν η συνθήκη ισχύει (είναι ΑΛΗΘΗΣ), τότε εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων **ΤΟΤΕ** και **ΤΕΛΟΣ_ΑΝ**, σε αντίθετη περίπτωση οι εντολές αυτές αγνοούνται. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση **ΤΕΛΟΣ_ΑΝ**

Σύνταξη

ΑΝ *συνθήκη* **ΤΟΤΕ**
εντολή-1
εντολή-2
…
εντολή-ν
ΑΛΛΙΩΣ
εντολή-1
εντολή-2
…
εντολή-ν
ΤΕΛΟΣ_ΑΝ

Παράδειγμα

ΑΝ αριθμός > 0 **ΤΟΤΕ**
ΓΡΑΨΕ ‘Ο αριθμός είναι θετικός’
Πλήθος_θετικών <- Πλήθος_θετικών+1
ΑΛΛΙΩΣ
ΓΡΑΨΕ ‘Ο αριθμός είναι αρνητικός ή 0’
Πλήθος_μη_θετικών <- Πλήθος_μη_θετικών +1
ΤΕΛΟΣ_ΑΝ

Λειτουργία

Αν η συνθήκη ισχύει, τότε εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων **ΤΟΤΕ** και **ΑΛΛΙΩΣ**, διαφορετικά εκτελούνται οι εντολές μεταξύ **ΑΛΛΙΩΣ** και **ΤΕΛΟΣ_ΑΝ**. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση **ΤΕΛΟΣ_ΑΝ**

Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από τις δύο, τότε μπορούν να χρησιμοποιηθούν πολλές εντολές **ΑΝ** η μία μέσα στην άλλη, οι **εμφωλευμένες** εντολές **ΑΝ**, όπως ονομάζονται. Π.χ.:

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

Πολύ συχνά οι εντολές που έχουν γραφεί με εμφωλευμένα **ΑΝ,** μπορούν να γραφούν πιο απλά χρησιμοποιώντας σύνθετες εκφράσεις ή την εντολή επιλογής **ΑΝ_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
```

Σύνταξη

```
ΑΝ συνθήκη-1 ΤΟΤΕ
        εντολή-1
        εντολή-2
        …
        εντολή-ν
  ΑΛΛΙΩΣ_ΑΝ συνθήκη-2 ΤΟΤΕ
        εντολή-1
        εντολή-2
        …
        εντολή-ν
  …
  ΑΛΛΙΩΣ
        εντολή-1
        εντολή-2
        …
        εντολή-ν
ΤΕΛΟΣ_ΑΝ
```

Παράδειγμα

```
ΑΝ αριθμός > 0 ΤΟΤΕ
        ΓΡΑΨΕ 'Ο αριθμός είναι θετικός'
        Πλήθος-θετικών <- Πλήθος_θετικών+1
  ΑΛΛΙΩΣ_ΑΝ αριθμός <0 ΤΟΤΕ
        ΓΡΑΨΕ 'Ο αριθμός είναι αρνητικός '
        Πλήθος_αρνητικών <- Πλήθος_ αρνητικών +1
  ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Ο αριθμός είναι 0'
        Πλήθος_0 <- Πλήθος_0 +1
ΤΕΛΟΣ_ΑΝ
```

Λειτουργία

Εκτελούνται οι εντολές που βρίσκονται στο αντίστοιχο τμήμα, όταν η συνθήκη είναι αληθής. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση **ΤΕΛΟΣ_ΑΝ .**

Παράδειγμα 1

Στο πρόγραμμα του προηγούμενου κεφαλαίου (πωλήσεις υπολογιστών) υποθέτουμε ότι η τιμή των υπολογιστών εξαρτάται από την ποσότητα παραγγελίας. Συγκεκριμένα ισχύουν οι διπλανές τιμές αγοράς υπολογιστών.

ΠΟΣΟΤΗΤΑ	ΤΙΜΗ ΜΟΝΑΔΟΣ
1-50	200.000
51-100	180.000
101-200	160.000
>200	150.000

Ο υπολογισμός με χρήση εμφωλευμένων εντολών ΑΝ είναι:

```
ΑΝ Ποσότητα=<50 ΤΟΤΕ
        Κόστος <— Ποσότητα*200000
  ΑΛΛΙΩΣ
        ΑΝ Ποσότητα =< 100 ΤΟΤΕ
                Κόστος <— Ποσότητα*180000
          ΑΛΛΙΩΣ
                ΑΝ Ποσότητα =< 200 ΤΟΤΕ
                        Κόστος <— Ποσότητα*160000
                 ΑΛΛΙΩΣ
                        Κόστος <— Ποσότητα*150000
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
```

Το ίδιο πρόγραμμα με τη χρήση της εντολής **ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ_ΑΝ:**

```
ΑΝ Ποσότητα=<50 ΤΟΤΕ
        Κόστος <— Ποσότητα*200000
  ΑΛΛΙΩΣ_ΑΝ Ποσότητα =<100 ΤΟΤΕ
        Κόστος <— Ποσότητα*180000
  ΑΛΛΙΩΣ_ΑΝ Ποσότητα =<200 ΤΟΤΕ
        Κόστος <— Ποσότητα*160000
  ΑΛΛΙΩΣ
        Κόστος <— Ποσότητα*150000
ΤΕΛΟΣ_ΑΝ
```

Στο προηγούμενο παράδειγμα για το οποίο θεωρούμε ότι η ποσότητα είναι θετικός αριθμός, ένα παράδειγμα περιττών ελέγχων είναι το ακόλουθο:

ΑΝ Ποσότητα<=50 **ΤΟΤΕ**
    Κόστος <— Ποσότητα*200000
ΑΛΛΙΩΣ_ΑΝ Ποσότητα>50 **ΚΑΙ** Ποσότητα =<100 **ΤΟΤΕ**
    Κόστος <— Ποσότητα*180000
ΑΛΛΙΩΣ_ΑΝ Ποσότητα>100 **ΚΑΙ** Ποσότητα =<200 **ΤΟΤΕ**
    Κόστος <— Ποσότητα*160000
ΑΛΛΙΩΣ
    Κόστος <— Ποσότητα*150000
ΤΕΛΟΣ_ΑΝ

8.1.2 Εντολή ΕΠΙΛΕΞΕ

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 ανήκουν 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.
Π.χ.: ΠΕΡΙΠΤΩΣΗ 10..100 (από 10 έως και 100)

8.2 Εντολές επανάληψης

Η τρίτη βασική δομή είναι η δομή επανάληψης, ο βρόχος, η οποία επιτρέπει την εκτέλεση εντολών περισσότερες από μία φορά. Οι επαναλήψεις ελέγχονται πάντοτε από κάποια συνθήκη, η οποία καθορίζει την έξοδο από το βρόχο.

8.2.1 Εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ

Σύνταξη

ΟΣΟ *συνθήκη* **ΕΠΑΝΑΛΑΒΕ**
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Παράδειγμα

Άθροισμα<—0
ΟΣΟ *Άθροισμα<1000* **ΕΠΑΝΑΛΑΒΕ**
ΔΙΑΒΑΣΕ *Α*
Άθροισμα<— Άθροισμα+Α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Λειτουργία

Ελέγχεται η συνθήκη και αν είναι Αληθής, εκτελούνται οι εντολές που βρίσκονται ανάμεσα στις **ΟΣΟ_ΕΠΑΝΑΛΑΒΕ** και **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ.** Στη συνέχεια ελέγχεται πάλι η συνθήκη και αν ισχύει, εκτελούνται πάλι οι ίδιες εντολές. ‘Οταν η λογική έκφραση γίνει Ψευδής, τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά το **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**. Εφόσον μετά από κάθε επανάληψη ελέγχεται εκ νέου η συνθήκη, πρέπει υποχρεωτικά μέσα στο βρόχο να υπάρχει μία εντολή, η οποία να μεταβάλει την τιμή της μεταβλητής που ελέγχεται με τη συνθήκη. Σε αντίθετη περίπτωση η επανάληψη δε θα τερματίζεται και θα εκτελείται συνεχώς. Χρησιμοποιείται για άγνωστο πλήθος επαναλήψεων.

8.2.2 Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ

Σύνταξη

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
εντολή-1
εντολή-2
…
εντολή-ν
ΜΕΧΡΙΣ_ΟΤΟΥ *λογική-έκφραση*

Παράδειγμα

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
ΔΙΑΒΑΣΕ Α
Άθροισμα <- Άθροισμα +Α
ΜΕΧΡΙΣ_ΟΤΟΥ *Άθροισμα >= 1000*

Λειτουργία

Εκτελούνται οι εντολές μεταξύ των **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** και **ΜΕΧΡΙΣ_ΟΤΟΥ. Στ**η συνέχεια ελέγχεται η λογική έκφραση και αν δεν ισχύει (είναι ψευδής), τότε οι εντολές που βρίσκονται ανάμεσα στις **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** και **ΜΕΧΡΙΣ_ΟΤΟΥ,** εκτελούνται πάλι. Ελέγχεται ξανά η λογική έκφραση και αν δεν ισχύει, επαναλαμβάνεται η εκτέλεση των ίδιων εντολών. Όταν η λογική έκφραση γίνει Αληθής τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά από την **ΜΕΧΡΙΣ_ΟΤΟΥ**.
Γενικά σε περιπτώσεις όπου η επανάληψη θα συμβεί υποχρεωτικά μία φορά, είναι προτιμότερη η χρήση της **ΜΕΧΡΙΣ_ΟΤΟΥ**.
Χαρακτηριστική περίπτωση όπου προτιμάται η εντολή **ΜΕΧΡΙΣ_ΟΤΟΥ** είναι στον έλεγχο αποδεκτών τιμών καθώς και στην επιλογή από προκαθορισμένες απαντήσεις ή μενού.

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ! Έλεγχος δεδομένων θετικοί αριθμοί μόνο
    ΓΡΑΨΕ ‘Δώσε Αριθμό’
    ΔΙΑΒΑΣΕ Χ
    ΑΝ Χ=<0 ΤΟΤΕ
        ΓΡΑΨΕ ‘Λάθος Αριθμός, Παρακαλώ δώστε ξανά...’
    ΤΕΛΟΣ_ΑΝ
    ! Αν το Χ δεν είναι θετικό εισάγουμε νέο αριθμό
ΜΕΧΡΙΣ_ΟΤΟΥ Χ>0
```

8.2.3 Εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ

Σύνταξη

ΓΙΑ *μεταβλητή* **ΑΠΟ** *τιμή1* **ΜΕΧΡΙ** *τιμή2* **ΜΕ ΒΗΜΑ** *τιμή3*
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Παράδειγμα

ΓΙΑ Αριθμό **ΑΠΟ** 1 **ΜΕΧΡΙ** 100 **ΜΕ ΒΗΜΑ** 2
Άθροισμα <- Άθροισμα+Αριθμό
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Λειτουργία

Οι εντολές του βρόχου εκτελούνται για όλες τις τιμές της μεταβλητής από την αρχική τιμή μέχρι την τελική τιμή (έως ότου η τιμή ελέγχου γίνει μεγαλύτερη της τελικής τιμής), αυξανόμενες με την τιμή του βήματος. Αν το βήμα είναι ίσο με 1, τότε παραλείπεται. Χρησιμοποιείται μόνο για γνωστό πλήθος επαναλήψεων.

Στη χρήση των **εμφωλευμένων βρόχων** ισχύουν συγκεκριμένοι κανόνες που πρέπει να ακολουθούνται αυστηρά για την σωστή λειτουργία των προγραμμάτων.
Συγκεκριμένα:

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

Ποτέ μη χρησιμοποιείς εντολές που αλλάζουν την αρχική τιμή, την τελική τιμή, το βήμα ή τη μεταβλητή που ελέγχει την επανάληψη μέσα σε ένα βρόχο ΓΙΑ. Αν και μερικές γλώσσες προγραμματισμού επιτρέπουν αυτές τις αλλαγές, να τις αποφεύγεις, γιατί οδηγούν σε προγράμματα δυσνόητα και συνήθως λανθασμένα.

Άσκηση 1:

Έστω το παρακάτω τμήμα προγράμματος:

```
Κ <- 0
ΓΙΑ Ι ΑΠΟ 0 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 5
        Α <- Ι^3
        Κ <- Κ+Α
        ΓΡΑΨΕ Ι, Α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Κ
ΓΡΑΨΕ Ι,Α
```

Πόσες φορές θα εκτελεστεί ο βρόχος; Ποια η λειτουργία των εντολών; Τι θα τυπώσει η τελευταία ΓΡΑΨΕ;
Γράψτε τις παραπάνω εντολές χρησιμοποιώντας την εντολή επανάληψης ΟΣΟ και την εντολή επανάληψης ΜΕΧΡΙΣ_ΟΤΟΥ. Ποιον από τους τρεις τρόπους προτιμάς και γιατί.

Άσκηση 2:

Διάβασε προσεκτικά τα παρακάτω τμήματα προγράμματος. Ποια είναι τα λάθη; Διόρθωσέ τα, ώστε να λειτουργούν σωστά.

Α.

```
ΔΙΑΒΑΣΕ Μισθός
ΟΣΟ Μισθός <>0 ΕΠΑΝΑΛΑΒΕ
    Άθροισμα <- 0
    ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
        Μέγιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
        Ελάχιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    Άθροισμα <- Άθροισμα+Μισθός
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Β.

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    Άθροισμα <- 0
    ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
        Μέγιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
        Ελάχιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    Άθροισμα <- Άθροισμα+Μισθός
    ΔΙΑΒΑΣΕ Μισθός
ΜΕΧΡΙΣ_ΟΤΟΥ Μισθός<>0
```

Γ.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
    Άθροισμα <- 0
    ΔΙΑΒΑΣΕ Μισθός
    ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
        Μέγιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
        Ελάχιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    Άθροισμα <- Άθροισμα+Μισθός
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εκτέλεσε εικονικά τις εντολές στο χαρτί και σημείωνε τα αποτελέσματα που προκύπτουν. Με αυτόν τον τρόπο θα δεις τα λάθη και στη συνέχεια θα κάνεις τις διορθώσεις.

Άσκηση 3:

Πόσες φορές θα εκτελεστεί η παρακάτω επανάληψη

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2 ΜΕ_ΒΗΜΑ 3
        ΓΡΑΨΕ 'Μήνυμα'
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Α. 2 Β. 0 Γ. 1 Δ. Άπειρες

Άσκηση 4:

Να γραφεί πρόγραμμα το οποίο θα εκτελεί κάποια από τις βασικές πράξεις πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση ανάμεσα σε δύο ακέραιους αριθμούς και θα εμφανίζει το αποτέλεσμα στην οθόνη. Το πρόγραμμα θα ελέγχεται από το παρακάτω μενού επιλογής και θα σταματάει όταν ο χρήστης επιλέξει από το μενού την επιλογή έξοδο.

1. Πρόσθεση
2. Αφαίρεση
3. Πολλαπλασιασμό
4. Διαίρεση
5. Έξοδος

Δώσε επιλογή:__

Άσκηση 5:

Η φορολογία εισοδήματος φυσικών προσώπων υπολογίζεται από τις αρμόδιες υπηρεσίες του υπουργείου των Οικονομικών κλιμακωτά, με τη βοήθεια του παρακάτω πίνακα.

ΚΛΙΜΑΚΑ ΥΠΟΛΟΓΙΣΜΟΥ ΦΟΡΟΥ ΕΙΣΟΔΗΜΑΤΟΣ ΦΥΣΙΚΩΝ ΠΡΟΣΩΠΩΝ
ΟΙΚΟΝ. ΕΤΟΥΣ 1999

Κλιμάκιο εισοδήματος	Φορολογικός συντελεστής	Φόρος κλιμακίου	Σύνολο	
			εισοδήματος	φόρου
1.055.000	0	0	1.055.000	0
1.582.500	5	79.125	2.637.500	79.125
1.582.500	15	237.375	4.220.000	316.500
3.165.000	30	949.500	7.385.000	1.266.000
8.440.000	40	3.376.000	15.825.000	4.642.000
Υπερβάλλον	45			

Για κάθε φορολογούμενο δίνονται τα εξής στοιχεία: αριθμός φορολογικού μητρώου (ΑΦΜ), όνομα φορολογούμενου, φορολογητέο εισόδημα Να γραφτεί πρόγραμμα το οποίο:

Να διαβάζει τα στοιχεία των φορολογουμένων, να υπολογίζει και να τυπώνει το φόρο που τους αντιστοιχεί. Το πρόγραμμα θα διαβάζει τα στοιχεία πολλών φορολογουμένων και θα τελειώνει όταν διαβάζει για ΑΦΜ τον αριθμό 0.

Άσκηση 6:

Να γραφεί ένα πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό και να τον αναλύει σε γινόμενο πρώτων παραγόντων.

Άσκηση 7:

Ένας τρόπος υπολογισμού των τριγωνομετρικών συναρτήσεων, που χρησιμοποιείται συχνά από τους υπολογιστές είναι με τον υπολογισμό των παρακάτω σειρών:

$$\eta\mu x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$\sigma\upsilon\nu x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Να γράψεις πρόγραμμα το οποίο να διαβάζει τη γωνία x σε μοίρες και να υπολογίζει το ημίτονο και το συνημίτονο της σύμφωνα με τους παραπάνω τύπους. Ποια μπορεί να είναι τα κριτήρια για διακοπή των επαναλήψεων;
Υπόδειξη: Να μετατρέψεις αρχικά τη γωνία x σε ακτίνια.

Άσκηση 8 ΘΕΜΑ 3^{ο} 2006
Άσκηση 9 ΘΕΜΑ 2^{ο} 2008
Άσκηση 10 ΘΕΜΑ 2^{ο} εσπεριν 2004

ΚΕΦΑΛΑΙΟ 9 (+3)

9.1. Μονοδιάστατοι πίνακες. & 3.3 Πίνακες

Ένα πρόγραμμα το οποίο διαβάζει τις θερμοκρασίες διαφόρων ημερών του μήνα, έστω 30, και υπολογίζει τη μέση θερμοκρασία, μπορεί πολύ απλά να γραφεί ως εξής:

….
Σύνολο <- 0
ΓΙΑ Ημέρα **ΑΠΟ** 1 **ΜΕΧΡΙ** 30
    ΔΙΑΒΑΣΕ Θερμοκρασία
    Σύνολο <- Σύνολο+Θερμοκρασία
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
Μέση <- Σύνολο /30
…..

Χρησιμοποιώντας λοιπόν μόνο μία μεταβλητή, τη μεταβλητή *Θερμοκρασία*, το πρόβλημα λύνεται πολύ απλά και το αντίστοιχο πρόγραμμα είναι σύντομο και κατανοητό.

Αν όμως στο προηγούμενο πρόγραμμα ζητείται και το πλήθος των ημερών που η θερμοκρασία ήταν κατώτερη της μέσης, τότε η σύγκριση αυτή πρέπει να γίνει μετά τον υπολογισμό της μέσης θερμοκρασίας. Αυτό σημαίνει ότι όλες οι θερμοκρασίες πρέπει να **επαναεισαχθούν** για να συγκριθούν με τη μέση.

Μία άλλη λύση είναι να καταχωρηθεί κάθε θερμοκρασία σε **διαφορετική μεταβλητή**, έτσι ώστε κάθε τιμή που εισάγεται να διατηρείται στη μνήμη και να μπορεί να συγκριθεί με τη μέση, αφού αυτή υπολογιστεί. Τότε όμως πρέπει να δημιουργηθούν 30 διαφορετικές μεταβλητές Θερμοκρασία1, Θερμοκρασία2,..., Θερμοκρασία30. Για να γραφεί το πρόγραμμα χρειάζονται τριάντα εντολές ΔΙΑΒΑΣΕ και τριάντα εντολές ΑΝ.

Η καλύτερη λύση στο πρόβλημα αυτό είναι η χρήση **μεταβλητής** με **δείκτες**, έννοια που είναι γνωστή από τα μαθηματικά και υλοποιείται στον προγραμματισμό με τη δομή δεδομένων του **πίνακα.** Χρησιμοποιείται λοιπόν μόνο **ένα** όνομα *Θερμοκρασία***,** που αναφέρεται και στις **τριάντα** διαφορετικές θερμοκρασίες.

Το **όνομα** του πίνακα καθορίζει μία ομάδα **διαδοχικών θέσεων** στη μνήμη. Κάθε συγκεκριμένη θέση μνήμης καλείται **στοιχείο** του πίνακα και προσδιορίζεται από την τιμή ενός **δείκτη**, όπως φαίνεται στο σχήμα 9.1.

Σχ. 9.1. Ο πίνακας θερμοκρασία

Μπορούμε να ορίσουμε τον **πίνακα** ως μια **στατική δομή δεδομένων** που περιέχει στοιχεία του **ίδιου τύπου** (δηλαδή ακέραιους, πραγματικούς κ.λπ). Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται. Όμως, γενικά η **αναφορά στα στοιχεία** ενός πίνακα γίνεται με τη χρήση του **συμβολικού ονόματος** του πίνακα ακολουθούμενου από την **τιμή** ενός ή περισσότερων ***δεικτών*** (indexes) σε παρένθεση ή αγκύλη.

Οι πίνακες που χρησιμοποιούν **ένα μόνο δείκτη** για την αναφορά των στοιχείων τους, ονομάζονται **μονοδιάστατοι** πίνακες.

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 [και]. Το στοιχείο *Θερμοκρασία[2],* εκφράζει τη θερμοκρασία της δεύτερης ημέρας, αναφέρεται στο δεύτερο στοιχείο του πίνακα *Θερμοκρασία* και έχει την τιμή 27. Γενικότερα το στοιχείο Θερμοκρασία[i] αναφέρεται στο i-στό στοιχείο του 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 το **πλήθος 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. Π.χ.:

ΜΕΤΑΒΛΗΤΕΣ

ΠΡΑΓΜΑΤΙΚΕΣ:Θερμοκρασία [30]

Πίνακας είναι ένα **σύνολο αντικειμένων ίδιου τύπου**, τα οποία αναφέρονται με ένα **κοινό όνομα**. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται **στοιχείο** του πίνακα. Η **αναφορά** σε ατομικά στοιχεία του πίνακα γίνεται με το **όνομα** του πίνακα ακολουθούμενο από ένα **δείκτη**.

Παράδειγμα 1

Χρησιμοποιώντας μεταβλητές με δείκτες για το προηγούμενο παράδειγμα έχουμε το εξής πρόγραμμα:

```
ΠΡΟΓΡΑΜΜΑ Θερμοκρασίες
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: Θερμοκρασία[30], Μέση, Σύνολο
        ΑΚΕΡΑΙΕΣ: i, Ημέρες
ΑΡΧΗ
Σύνολο <- 0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 30
        ΓΡΑΨΕ ‘Δώσε τη θερμοκρασία’
        ΔΙΑΒΑΣΕ Θερμοκρασία[i]
        Σύνολο <- Σύνολο+ Θερμοκρασία[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
Μέση <- Σύνολο/30
Ημέρες <- 0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ Θερμοκρασία[i] < Μέση ΤΟΤΕ
                Ημέρες <- Ημέρες+1
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ‘Μέση Θερμοκρασία:’, Μέση
ΓΡΑΨΕ ‘Ημέρες με μικρότερη θερμοκρασία’, Ημέρες
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Με τη χρήση του πίνακα, όλα τα δεδομένα καταχωρούνται κάτω από το ίδιο όνομα μεταβλητής, στο παράδειγμα Θερμοκρασία. Η ανάγνωση όλων των δεδομένων απλοποιείται, αφού είναι μόνο μία εντολή, **ΔΙΑΒΑΣΕ** Θερμοκρασία[i], η οποία βρίσκεται μέσα σε ένα βρόχο και επαναλαμβάνεται όσες φορές απαιτείται όπως και ο υπολογισμός του αθροίσματος, Σύνολο<-Σύνολο+ Θερμοκρασία[i].

Οι τιμές μετά τον υπολογισμό της μέσης τιμής δεν χάνονται, αφού βρίσκονται στα στοιχεία του πίνακα και ξαναχρησιμοποιούνται για την εύρεση του αριθμού των ημερών, που η θερμοκρασία είναι μικρότερη από τη μέση. Ο υπολογισμός των ημερών είναι μόνο μία εντολή ΑΝ, η οποία βρίσκεται σε ένα βρόχο και επαναλαμβάνεται 30 φορές.

Βασικοί αλγόριθμοι μονοδιάστατων πινάκων

Στη συνέχεια θα δούμε τους βασικούς αλγορίθμους των βασικών λειτουργιών (πράξεων) επί των μονοδιάστατων πινάκων καθώς και τους αλγόριθμους των συνηθέστερων τυπικών επεξεργασιών πινάκων (θα τις αναφέρουμε σε επόμενο κεφάλαιο).

Οι **βασικές λειτουργίες** (ή αλλιώς **πράξεις**) επί των δομών δεδομένων είναι οι ακόλουθες:

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

Οι τυπικές επεξεργασίες είναι:

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

🛠 **Εισαγωγή, Διαγραφή : Οι πράξεις αυτές δεν υφίστανται στους πίνακες.** Αυτό διότι από το τμήμα δηλώσεων δηλώνουμε το μέγεθος του πίνακα π.χ. 100 θέσεις μνήμης. Οι θέσεις αυτές δεσμεύονται και το μέγεθος του πίνακα δεν μπορεί να αλλάξει άρα δεν μπορεί να γίνει εισαγωγή νέου κόμβου ή διαγραφή κάποιου άλλου. Μπορούμε μόνο να αλλάξουμε το περιεχόμενο ενός κόμβου δίνοντας μια (νέα) τιμή μέσω της πράξης της προσπέλασης.

- ***Προσπέλαση*** (access), πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.

Βασικό πρόγραμμα προσπέλασης:

```
ΠΡΟΓΡΑΜΜΑ Προσπέλαση
ΜΕΤΑΒΛΗΤΕΣ
        ! δήλωση μεγέθους και τύπου του πίνακα
        ΠΡΑΓΜΑΤΙΚΕΣ: Π[1000], Χ
        ΑΚΕΡΑΙΕΣ: i

ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε πλήθος στοιχείων πίνακα (1-1000)'
ΔΙΑΒΑΣΕ Ν
! προσπέλαση τροποποίησης περιεχομένου κόμβων
! (διάβασμα - εισαγωγή στοιχείων του πίνακα)

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΓΡΑΨΕ 'Δώσε το', i, 'ο στοιχείο του πίνακα'
        ΔΙΑΒΑΣΕ Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Π[Ν]←8

! προσπέλαση εξέτασης περιεχομένου κόμβων
!(ανάθεση τιμών τους σε άλλη μεταβλητή εκτυπώσεις περιεχομένου κόμβων κ.λ.π. )
Χ←Π[Ν]+5

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΓΡΑΨΕ Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ Π[2], Π[2+3], Π[Ν-5]

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Άσκηση 1:

Να γράψετε τις δηλώσεις των παρακάτω πινάκων, καθώς και τις εντολές με τις οποίες εκχωρούνται οι τιμές σε αυτά.

Α. Πίνακας 5 στοιχείων που κάθε στοιχείο έχει την τιμή του δείκτη του.

Β. Πίνακας που θα περιέχει τα ψηφία.

Γ. Πίνακας που περιέχει τα ονόματα των συμμαθητών σου.

Δ. Πίνακας με 10 στοιχεία, πρώτο στοιχείο τον αριθμό 500 και κάθε επόμενο στοιχείο να είναι το μισό του προηγούμενο, δηλαδή το δεύτερο 250, το τρίτο 125 κοκ.

🛠 Υπολογισμός **αθροισμάτων** στοιχείων του πίνακα:

Βασικό πρόγραμμα προσπέλασης:

```
ΠΡΟΓΡΑΜΜΑ Άθροισμα
ΜΕΤΑΒΛΗΤΕΣ
        ! δήλωση μεγέθους και τύπου του πίνακα
        ΠΡΑΓΜΑΤΙΚΕΣ: Π[1000], Sum
        ΑΚΕΡΑΙΕΣ: i

ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε πλήθος στοιχείων πίνακα (1-1000)'
ΔΙΑΒΑΣΕ N

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
        ΓΡΑΨΕ 'Δώσε το', i, 'ο στοιχείο του πίνακα'
        ΔΙΑΒΑΣΕ Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Sum←0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
        Sum←Sum + Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ 'Το άθροισμα των στοιχείων του πίνακα είναι:', Sum

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Άσκηση 2:
Να γράψετε ένα πρόγραμμα το οποίο διαβάζει 2 ακέραιους πίνακες 100 θέσεων, και σε έναν τρίτο πίνακα θέτει σε κάθε θέση ως τιμή το άθροισμα των τιμών των αντίστοιχων θέσεων του πίνακα που διάβασε.

🛠 ***Αναζήτηση*** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.

Η πιο απλή μορφή αναζήτησης στοιχείου σε πίνακα είναι η ***σειριακή*** (sequential) ή ***γραμμική*** (linear) μέθοδος. Έτσι για τον επόμενο αλγόριθμο Sequential_Search υποτίθεται ότι αναζητείται η τιμή key στο μη ταξινομημένο πίνακα table. Μετά την εκτέλεση του αλγορίθμου η μεταβλητή position επιστρέφει την τιμή 0, αν η αναζήτηση είναι ανεπιτυχής, ενώ αν η αναζήτηση είναι επιτυχής, τότε επιστρέφει τη θέση του στοιχείου στον πίνακα (δηλαδή, έναν αριθμό από 1 ως n).

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

Ερώτηση: Μπορείς να αποφύγεις την χρήση της done σ αυτόν τον αλγόριθμο;

Όπως αναφέρθηκε, τα στοιχεία που περιέχονται στον πίνακα table δεν είναι ταξινομημένα. Επίσης, ο προηγούμενος αλγόριθμος ισχύει για την περίπτωση όπου κάθε στοιχείο υπάρχει μία μόνο φορά στον πίνακα. Αν κάποιο στοιχείο εμφανίζεται στον πίνακα περισσότερο από μία φορές, τότε ο αλγόριθμος πρέπει να τροποποιηθεί κατά το εξής: η μεταβλητή done είναι περιττή και η αναζήτηση συνεχίζεται μέχρι το τέλος του πίνακα ελέγχοντας με τη συνθήκη i <= n. Εξ άλλου, αν τα στοιχεία του πίνακα είναι ταξινομημένα, τότε ο αλγόριθμος πρέπει να σταματήσει, μόλις συναντήσει κάποιο στοιχείο που είναι μεγαλύτερο από το αναζητούμενο.

Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερη αποτελεσματική μέθοδος αναζήτησης. Έτσι, δικαιολογείται η χρήση της μόνο σε περιπτώσεις όπου:

1. _ ο πίνακας είναι μη ταξινομημένος,
2. _ ο πίνακας είναι μικρού μεγέθους (για παράδειγμα, n <= 20),
3. _ η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,

Μία αποτελεσματικότερη μέθοδος αναζήτησης, η **δυαδική αναζήτηση**.

Το παρακάτω πρόγραμμα παρουσιάζει έναν πλήρη αλγόριθμο αναζήτησης ο οποίος βρίσκει αν υπάρχει ένα στοιχείο, πόσες φορές υπάρχει, και όλες του τις θέσεις τις οποίες αποθηκεύει σε ένα πίνακα θέσεων.

```
ΠΡΟΓΡΑΜΜΑ Σειριακή_αναζήτηση_πλήρης
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: Π[1000], Ψ
        ΑΚΕΡΑΙΕΣ: i,φ, Θ[1000]

ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε πλήθος στοιχείων πίνακα (1-1000)'
ΔΙΑΒΑΣΕ Ν

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΓΡΑΨΕ 'Δώσε το', i, 'ο στοιχείο του πίνακα'
        ΔΙΑΒΑΣΕ Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ 'Δώσε το στοιχείο που ψάχνεις'
ΔΙΑΒΑΣΕ Ψ

φ←0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΑΝ Ψ= Π[i] ΤΟΤΕ
                φ←φ+1
                Θ[φ]←i
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΑΝ φ=0 ΤΟΤΕ
        ΓΡΑΨΕ 'Δεν βρέθηκε'
  ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Βρέθηκε ',φ,' φορές στις παρακάτω θέσεις:'
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ φ
                ΓΡΑΨΕ Θ[i]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Άσκηση 3:

Να γράψετε πρόγραμμα το οποίο σε τρεις πίνακες, στις αντίστοιχες θέσεις θα διαβάζει και θα αποθηκεύει (με τυχαία σειρά) το όνομα, φύλλο, και βαθμό των μαθητών μιας τάξης. Στη συνέχεια θα βρίσκει τον Μ.Ο. των αγοριών και τον Μ.Ο. των κοριτσιών. Κατόπιν θα βρίσκει και θα τυπώνει πόσα και ποια κορίτσια είχαν βαθμολογία πάνω από το μέσο όρο των κοριτσιών και πόσα και ποια αγόρια είχαν βαθμολογία πάνω από το Μ.Ο. των αγοριών.

Άσκηση 4:
Να τροποποιήσετε τον **Αλγόριθμος** Sequential_Search υποθέτοντας ότι εφαρμόζεται σε ταξινομημένο πίνακα και να τον κάνετε να σταματά μόλις βρει τιμή μεγαλύτερη από την τιμή που ψάχνει.

🛠 Εύρεση του **μέγιστου** ή του **ελάχιστου** στοιχείου.

```
ΠΡΟΓΡΑΜΜΑ μέγιστο
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: Π[1000], max
        ΑΚΕΡΑΙΕΣ: i

ΑΡΧΗ
ΓΡΑΨΕ ‘Δώσε πλήθος στοιχείων πίνακα (1-1000)’
ΔΙΑΒΑΣΕ Ν

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΓΡΑΨΕ ‘Δώσε το’, i, 'ο στοιχείο του πίνακα'
        ΔΙΑΒΑΣΕ Π[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ


max←Π[1]
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν  !είναι πιο λογικό να γράψουμε ΑΠΟ 2 αλλά δουλεύει σωστά και
                      !έτσι και επιπλέον βολεύει αν θέλουμε π.χ. να υπολογίσουμε
                      ! ταυτόχρονα και το άθροισμα.
        ΑΝ  Π[i] > max ΤΟΤΕ
                max← Π[i]
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ 'Μέγιστο=',max

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Άσκηση 5:
Να γράψετε πρόγραμμα που διαβάζει τα στοιχεία ενός πίνακα και στη συνέχεια βρίσκει και τυπώνει: Μ.Ο., μέγιστο και ελάχιστο στοιχείο, πόσες φορές εμφανίζεται το καθένα από αυτά και σε ποιες θέσεις του πίνακα. Σε μια έκδοση του προγράμματος οι θέσεις θα εκτυπώνονται απ ευθείας, σε μια άλλη θα αποθηκεύονται σε ένα πίνακα θέσεων και μετά θα εκτυπώνονται.

Άσκηση 6:
Έχουμε δύο πίνακες, ο ένας με τα μοντέλα των υπολογιστών και ο δεύτερος με τις τιμές τους. Να γράψετε τις εντολές που βρίσκουν και τυπώνουν το φθηνότερο μοντέλο καθώς και το ακριβότερο.

Άσκηση 7:
Σε μία κατασκήνωση υπάρχουν 300 παιδιά και καθένα από αυτά έχει μοναδικό αριθμό από το 1 έως και το 300 που του αντιστοιχεί. Για κάθε παιδί είναι γνωστή η ηλικία του. Να χρησιμοποιηθεί η δομή του πίνακα για να αποθηκεύονται οι ηλικίες των παιδιών και να βρεθεί ο κατάλληλος αλγόριθμος υπολογισμού του μικρότερου και μεγαλύτερου σε ηλικία παιδιού και να εκτυπώνεται τόσο η ηλικία όσο και ο κωδικός του μικρότερου και μεγαλύτερου παιδιού.

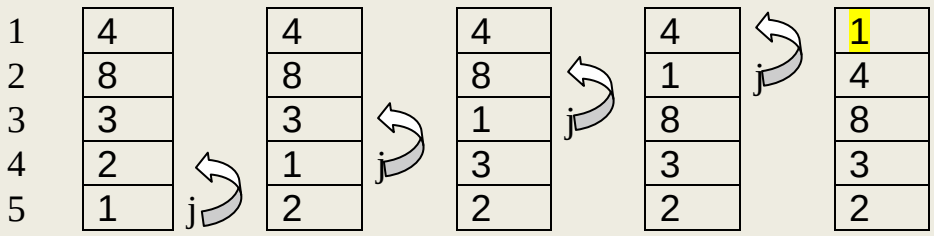
Οι παραπάνω πίνακες λέγονται **παράλληλοι**. Δύο οι περισσότεροι πίνακες λέγονται παράλληλοι, αν σε αυτούς έχουμε αποθηκεύσει τα χαρακτηριστικά οντοτήτων με τέτοιο τρόπο ώστε τα δεδομένα κάθε οντότητας να βρίσκονται σε στοιχεία με την ίδια τιμή δείκτη.

- ***Ταξινόμηση*** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

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

Η μέθοδος της *ταξινόμησης* ***ευθείας ανταλλαγής*** (straight exchange sort) βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων, μέχρις ότου διαταχθούν όλα τα στοιχεία. Η μέθοδος είναι γνωστή ως ***ταξινόμηση φυσσαλίδας*** (bubblesort).

Παράδειγμα: έστω ένας πίνακας 5 ακεραίων: 4,8,3,2,1. Ξεκινάμε από το τελευταίο στοιχείο και το συγκρίνουμε με το προηγούμενό του. Αν είναι μικρότερο τα ανταλλάσσουμε θέση αν όχι τα αφήνουμε όπως είναι και συνεχίζουμε με το προτελευταίο με το προηγούμενό του κ.λ.π.



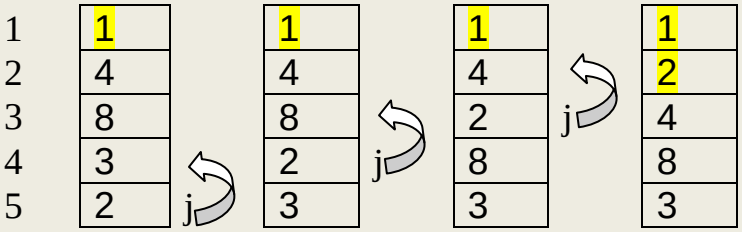
Τελειώνοντας αυτό τον "*πρώτο γύρο*" ο πίνακας δεν ταξινομήθηκε βέβαια ακόμα αλλά το μικρότερο στοιχείο του ανέβηκε σαν *φυσαλίδα* στην πρώτη θέση.
Σε κάθε μια σύγκριση (έστω της θέσης j με τη θέση j-1) κάναμε το εξής:

Αν table[j-1] > table[j] **τότε**
αντιμετάθεσε table[j-1], table[j]
Τέλος_αν

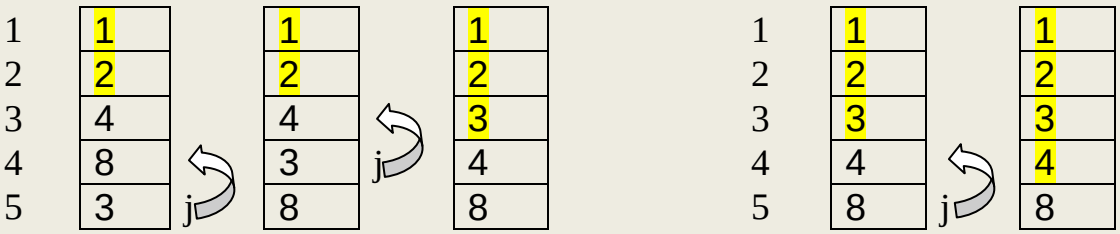
Αυτή η εντολή εκτελέστηκε 4 φορές (βλέπε τα 4 βέλη η βάση δείχνει το j και η μύτη το j-1) για τις τιμές του j από 5 (που είναι το μέγεθος του πίνακα) μέχρι 2.
Άρα αυτός ο πρώτος γύρος περιγράφεται ολόκληρος βάζοντας την Αν μέσα σε μια επαναληπτική διαδικασία:

Για j **από** 5 **μέχρι** 2 **με_βήμα** –1
Αν table[j-1] > table[j] **τότε**
αντιμετάθεσε table[j-1], table[j]
Τέλος_αν
Τέλος_επανάληψης

Στη συνέχεια αγνοούμε την πρώτη θέση του πίνακα (η οποία ήδη περιέχει την μικρότερη τιμή) και επαναλαμβάνουμε την διαδικασία σε έναν "***δεύτερο γύρο***" για τιμές του j από 5 (που είναι το μέγεθος του πίνακα) μέχρι 3.



"***Τρίτος γύρος***": "*Τέταρτος γύρος*"



Αφού λοιπόν έχουμε 4 "γύρους" το τμήμα του αλγορίθμου που περιγράφει ένα γύρο πρέπει να μπει κι αυτό μέσα σε μια επαναληπτική διαδικασία:

Για i από 1 μέχρι 4
Για j **από** 5 **μέχρι** 2 **με_βήμα** –1
Αν table[j-1] > table[j] **τότε**
αντιμετάθεσε table[j-1], table[j]
Τέλος_αν
Τέλος_επανάληψης
Τέλος_επανάληψης

Αν όμως παρατηρήσουμε προσεκτικά, οι επόμενοι 3 γύροι δεν τελειώνουν στο 2 όπως ο πρώτος, αλλά στο 3, 4 και στο 5. Επομένως στην εντολή **Για** j **από** 5 **μέχρι** 2 **με_βήμα** -1 στη θέση του 2 πρέπει να βάλουμε μια μεταβλητή με τιμές από 2 μέχρι 5. Παρατηρούμε ότι η μεταβλητή i δεν χρησιμοποιείται πουθενά αλλού παρά μόνο να μετρήσει 4 επαναλήψεις. Μπορούμε να χρησιμοποιήσουμε αυτή τη μεταβλητή με 2 τρόπους: Είτε γράφοντας **Για** j **από** 5 **μέχρι** i+1 **με_βήμα** -1 είτε όπως παρακάτω στον ολοκληρωμένο αλγόριθμο όπου επιπλέον αντί για 5 γράφουμε n όπου n το μέγεθος οποιουδήποτε πίνακα προς ταξινόμηση:

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

Στον αλγόριθμο αυτό ως είσοδος δίνεται η μεταβλητή table με n ακεραίους που πρέπει να ταξινομηθούν. Φυσικά η επιλογή του ακέραιου τύπου για το κλειδί είναι αυθαίρετη, αφού μπορεί να χρησιμοποιηθεί οποιοσδήποτε άλλος τύπος, όπου ορίζεται μία συνάρτηση διάταξης, όπως για παράδειγμα ο τύπος του χαρακτήρα.
Σημειώνεται ότι η εντολή “αντιμετάθεσε table[j-1], table[j]” ανταλλάσσει το περιεχόμενο δύο θέσεων με τη βοήθεια μίας βοηθητικής θέσης. Εναλλακτικά αυτό μπορεί να αυτό μπορεί να γίνει με εξής τρεις εντολές:

```
temp ← table[j-1]
table[j-1] ← table[j]
table[j] ← temp
```

Αν ο πίνακας "συνοδεύεται" από κάποιον άλλο πίνακα που περιέχει τιμές που αντιστοιχούν, τότε αν ταξινομούμε τον ένα οι ανταλλαγές της ταξινόμησης πρέπει να εφαρμόζονται και στον άλλο. Π.χ.:

Πριν την ταξινόμηση

Όνομα	*Βάρος*
Κώστας	65
Χαρά	70
Βασίλης	90

(Όνομα)

Μετά την ταξινόμηση

Όνομα	*Βάρος*
Βασίλης	90
Κώστας	65
Χαρά	70

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

Άσκηση 8:
Μία οικολογική οργάνωση διαθέτει στοιχεία για το ποσοστό δασών για 50 διαφορετικές χώρες. Χρειάζεται να πάρει απόφαση για να διοργανώσει μία εκδήλωση διαμαρτυρίας στις 10 χώρες που έχουν το χαμηλότερο ποσοστό δασών. Να δοθεί αλγόριθμος που θα ταξινομεί τα ποσοστά δασών των χωρών με χρήση της μεθόδου της ευθείας ανταλλαγής και θα εκτυπώνει τις 10 χώρες στις οποίες θα διοργανωθούν οι εκδηλώσεις.

Άσκηση 9:
Ο αλγόριθμος της φυσαλίδας όπως διατυπώθηκε έχει το μειονέκτημα ότι δεν είναι αρκετά “έξυπνος” ώστε να διαπιστώνει στην αρχή ή στο μέσο της διαδικασίας αν ο πίνακας είναι ταξινομημένος. Να σχεδιασθεί μία παραλλαγή του αλγορίθμου αυτού που να σταματά όταν διαπιστωθεί ότι τα στοιχεία του πίνακα είναι ήδη ταξινομημένα. Για πίνακα μεγέθους Ν πόσες επαναλήψεις απαιτεί η κλασική φυσαλίδα και πόσες η έξυπνη;
Υπόδειξη: Να χρησιμοποιήσετε μία βοηθητική μεταβλητή που να ελέγχει το τέλος κάθε επανάληψης του εξωτερικού βρόχου (”Για i από 2 μέχρι n”) αν για την τρέχουσα τιμή του i έγιναν αντιμεταθέσεις στοιχείων.

Άσκηση 10:
Έστω ότι θέλουμε να διατάξουμε τους μαθητές μίας τάξης κατά φθίνουσα σειρά ύψους. Η τεχνική είναι η εξής. Αρχικά, τοποθετούμε τους μαθητές σε μία τυχαία σειρά. Κατόπιν συγκρίνουμε το δεύτερο με τον πρώτο και αν χρειασθεί τους αντιμεταθέτουμε ώστε πρώτος να είναι ο ψηλότερος. Στη συνέχεια θεωρούμε τον τρίτο και τον τοποθετούμε στη σωστή σειρά σε σχέση μεν πρώτο και το δεύτερο. Κατ’ αυτόν τον τρόπο συνεχίζουμε μέχρι να τοποθετήσουμε στη σωστή σειρά όλους τους μαθητές. Να σχεδιασθεί ένας αλγόριθμος που να υλοποιεί αυτή τη μέθοδο ταξινόμησης.

Άσκηση 11:
Ένας μαθητής έχει μία συλλογή από CD και για κάθε CD έχει καταγράψει στον υπολογιστή τον τίτλο και την χρονιά έκδοσής του. Να ταξινομηθούν τα CD με βάση την χρονιά τους και να υπολογισθεί ο αριθμός των CD που έχει ο μαθητής με χρονολογία έκδοσης πριν από το 1995.

Σε πολλές περιπτώσεις χρειάζεται να γίνει ταξινόμηση και με βάση τα στοιχεία του δεύτερου πίνακα, αν τα στοιχεία του πρώτου είναι ίδια. Π.χ.:

Πριν την ταξινόμηση

Επώνυμο	*Όνομα*
Παππάς	Γεώργιος
Κουκούδης	Κώστας
Δήμου	Νίκος
Κουκούδης	Βασίλης
Αντωνίου	Πέτρος

Μετά την ταξινόμηση

Επώνυμο	*Όνομα*
Αντωνίου	Πέτρος
Δήμου	Νίκος
Κουκούδης	Βασίλης
Κουκούδης	Κώστας
Παππάς	Γεώργιος

Παρατηρήστε ότι το όνομα συμμετέχει στην ταξινόμηση μόνο στη περίπτωση ίδιου επωνύμου.

```
Αλγόριθμος Φυσαλίδα
Δεδομένα // ε,ο, n //
Για i από 2 μέχρι n
    Για j από n μέχρι i με_βήμα –1
        Αν ε[j-1] > ε[j] τότε
            αντιμετάθεσε ε[j-1], ε[j]
            αντιμετάθεσε ο[j-1], ο[j]
        αλλιώς_αν ε[j-1] = ε[j] τότε
            Αν ο[j-1] > ο[j] τότε
                αντιμετάθεσε ο[j-1], ο[j]
            Τέλος_αν
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // table //
Τέλος Φυσαλίδα
```

Άσκηση 12:
Σε 3 πίνακες Επώνυμο, Όνομα, Πατρώνυμο καταγράφουμε τα στοιχεία των μαθητών μιας τάξης. Να ταξινομηθούν όπως στον τηλεφωνικό κατάλογο. Όπου υπάρχει συνωνυμία σε επώνυμο και όνομα ταξινομούμε με βάση το πατρώνυμο.

- ***Αντιγραφή*** (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.

Άσκηση 13:
Να γράψετε πρόγραμμα το οποίο αφού διαβάσει τα στοιχεία ενός πίνακα ακεραίων Ν θέσεων τα αντιγράφει σε έναν άλλο πίνακα Ν θέσεων.

- ***Διαχωρισμός*** (separation), κατά την οποία μία ενιαία δομή χωρίζεται σε δύο ή περισσότερες δομές και αποτελεί την αντίστροφη πράξη της συγχώνευσης.

Άσκηση 14:
Να γράψετε πρόγραμμα το οποίο αφού διαβάσει τα στοιχεία ενός πίνακα Π ακεραίων 100 θέσεων αντιγράφει τα 30 πρώτα σ ένα πίνακα Π2 και τα υπόλοιπα σ έναν άλλο πίνακα Π3.

Άσκηση 15:
Να γράψετε πρόγραμμα το οποίο διαβάζει το ονοματεπώνυμο και το φύλλο 95 παιδιών και τα καταχωρίζει στους πίνακες Μ και Φ αντίστοιχα. Στη συνέχεια διαχωρίζει τον Μ στους πίνακες Α και Κ οι οποίοι περιέχουν τα ονοματεπώνυμα των αγοριών και των κοριτσιών αντίστοιχα.

- ***Συγχώνευση*** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.

Άσκηση 16:
(Απλή συγχώνευση-συνένωση). Να γράψετε πρόγραμμα το οποίο σε διαβάζει και καταχωρίζει: σε έναν πίνακα Α τα ονόματα των 15 αγοριών ενός τμήματος, σε ένα πίνακα Κ τα ονόματα των 17 κοριτσιών ενός τμήματος. Στη συνέχεια συνενώνει τους δύο πίνακες σε ένα πίνακα Μ.

Συγχώνευση δύο πινάκων.
Η συγχώνευση είναι μία από τις βασικές λειτουργίες σε πίνακες.
Σκοπός της είναι η δημιουργία από τα στοιχεία δύο (ή περισσότερων) **ταξινομημένων** πινάκων ενός άλλου, που είναι και αυτός **ταξινομημένος**.
Δίνονται δύο ταξινομημένοι κατά αύξουσα σειρά μονοδιάστατοι πίνακες, ακεραίων αριθμών. Να γραφεί πρόγραμμα το οποίο να συγχωνεύει τους δύο πίνακες σε ένα τρίτο ο οποίος να είναι επίσης ταξινομημένος κατά αύξουσα σειρά. Οι δύο αρχικοί πίνακες δεν μπορούν να περιέχουν περισσότερα από 100 στοιχεία ο καθένας.
Θεωρείται ότι στην είσοδο του αλγορίθμου συγχώνευσης δίνονται δύο ταξινομημένοι, κατά αύξουσα σειρά, πίνακες Α και Β, μεγέθους Ν και Μ στοιχείων αντίστοιχα, ενώ στην έξοδο προκύπτει ένας τρίτος πίνακας Γ με Ν+Μ ταξινομημένα στοιχεία επίσης κατά αύξουσα σειρά.
Στο πρόγραμμα Συγχώνευση που ακολουθεί οι μεταβλητές i, j και k είναι δείκτες για την κίνηση μέσα στους πίνακες Α, Β και Γ. Η μέθοδος προχωρεί ως εξής:
Το μικρότερο στοιχείο από τους πίνακες Α και Β τοποθετείται στον πίνακα Γ με ταυτόχρονη αύξηση του αντίστοιχου δείκτη. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου τελειώσουν τα στοιχεία του ενός πίνακα.
Στη συνέχεια τα υπόλοιπα στοιχεία του άλλου πίνακα μεταφέρονται στον πίνακα Γ.

```
ΠΡΟΓΡΑΜΜΑ Συγχώνευση
        ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ : Α[100], Β[100], Γ[200], Ι, J, Κ, Ν, Μ, Λ
        ! Α και Β αρχικοί πίνακες Γ τελικός πίνακας
ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε το πλήθος των στοιχείων του πίνακα Α (<100)'
ΔΙΑΒΑΣΕ Ν
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΔΙΑΒΑΣΕ Α[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'Δώσε το πλήθος των στοιχείων του πίνακα Β(<100)'
ΔΙΑΒΑΣΕ Μ
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΔΙΑΒΑΣΕ Β[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
! Συγχώνευση πινάκων Ι είναι ο δείκτης για  Α, J είναι ο δείκτης για τον  Β, Κ είναι ο δείκτης για τον Γ
Ι <- 1
J <- 1
Κ <- 1
ΟΣΟ Ι <= Ν ΚΑΙ J <= Μ ΕΠΑΝΑΛΑΒΕ
! Όσο και τα δύο έχουν στοιχεία
        ΑΝ Α[Ι] < Β[J] ΤΟΤΕ
                Γ[Κ] <- Α[Ι]
                Κ <- Κ+1
                Ι <- Ι+1
          ΑΛΛΙΩΣ
                Γ[Κ] <- Β[J]
                Κ <- Κ+1
                J <- J +1
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
! Μεταφορά των υπολοίπων στοιχείων του Α ή του Β
ΑΝ Ι > Ν ΤΟΤΕ
        ΓΙΑ Λ ΑΠΟ Κ ΜΕΧΡΙ Ν+Μ
                Γ[Λ] <- Β[J]
                J <- J +1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΑΛΛΙΩΣ
        ΓΙΑ Λ ΑΠΟ Κ ΜΕΧΡΙ Ν+Μ
                Γ[Λ] <- Α[Ι]
                Ι <- Ι+1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΑΝ
! Εκτύπωση τελικού πίνακα
ΓΙΑ Λ ΑΠΟ 1 ΜΕΧΡΙ Ν+Μ
        ΓΡΑΨΕ Γ[Λ]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Συγχώνευση
```

Άσκηση 17:
Στο προηγούμενο πρόγραμμα χρησιμοποιήσαμε 3 επαναληπτικές διαδικασίες. Η πρώτη άγνωστου πλήθους (αφού δεν ξέρουμε πότε θα εξαντληθεί κάποιος και ποιος πίνακας) μεταφέρει στοιχεία στο πίνακα Γ μέχρι να εξαντληθεί ένας από τους δύο πίνακες. Οι άλλες δύο (εκ των οποίων εκτελείται μόνο η μια) μεταφέρουν τα υπόλοιπα στοιχεία από τον ανεξάντλητο πίνακα στον Γ. Μια άλλη προσέγγιση έχει ως εξής:
Αφού πρέπει να "γεμίσει" ο πίνακας Γ με μέγεθος (Μ+Ν) η όλη διαδικασία είναι **μια** επαναληπτική διαδικασία ΓΙΑ γνωστού πλήθους Μ+Ν. Μέσα σ αυτή τη διαδικασία και με εμφωλευμένες ΑΝ μεταφέρουμε τα κατάλληλα στοιχεία στον Γ. Χρησιμοποιούμε τον δείκτη της ΓΙΑ για τον πίνακα Γ και δυο άλλους δείκτες για τους Α και Β. Να γράψετε πρόγραμμα που υλοποιεί τη συγχώνευση με το παραπάνω σκεπτικό.

Άσκηση 18 ΘΕΜΑ 2^{ο} 2004
Άσκηση 19 ΘΕΜΑ 3^{ο} 2005
Άσκηση 20 ΘΕΜΑ 1^{ο} Δ , ΘΕΜΑ 2^{Ο} επαναλ 2007
Άσκηση 21 ΘΕΜΑ 4^{ο} ΕΣΠΕΡ 2007
Άσκηση 22 ΘΕΜΑ 2^{ο} ΕΣΠΕΡ 2002
Άσκηση 23 ΘΕΜΑ 3^{ο} ΕΣΠΕΡ 2004
Άσκηση 24 ΘΕΜΑ 3^{ο} ΕΣΠΕΡ 2005

Άσκηση 25:
Θεωρείστε ότι έχετε έναν ήδη ταξινομημένο πίνακα. Ο πίνακας είναι 100 θέσεων αλλά έχει 99 ταξινομημένα στοιχεία στις πρώτες 99 θέσεις. Η 100στη θέση είναι άτιμη. Να γράψετε πρόγραμμα το οποίο διαβάζει μια τιμή και πρέπει να την εισάγει στον πίνακα στη σωστή θέση ώστε να παραμείνει ένας ταξινομημένος πίνακας 100 θέσεων 100 τιμών. Μια πρώτη σκέψη θα ήταν να διαβαστεί το στοιχείο στην 100στη θέση και μετά να γίνει ταξινόμηση. Μπορείτε να σκεφτείτε έναν πιο αποδοτικό αλγόριθμο; Να γράψετε το πρόγραμμα.

Άσκηση 26:
Με βάση το σκεπτικό της προηγούμενης άσκησης να γράψετε πρόγραμμα το οποίο ταξινομεί έναν πίνακα βάζοντας κάθε στοιχείο που διαβάζει στη σωστή του θέση την ώρα που το διαβάζει.

9.2. Πότε πρέπει να χρησιμοποιούνται πίνακες

Η χρήση πινάκων είναι ένας βολικός τρόπος για τη διαχείριση πολλών δεδομένων ιδίου τύπου, αλλά συχνά η χρήση τους είναι περιττή και επιζήμια στην ανάπτυξη του προγράμματος.
Πέρα από τα πλεονεκτήματα που αναφέρθηκαν, υπάρχουν και δύο **μειονεκτήματα** από τη 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 μεγάλων πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.

2. ***Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος.*** Όταν, χρησιμοποιούμε πίνακες υπάρχει ανώτατο όριο στο πλήθος των δεδομένων. Αυτό γιατί οι πίνακες είναι στατικές δομές και το μέγεθος τους πρέπει να δηλώνεται στην αρχή του προγράμματος, ενώ παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση του προγράμματος.

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

9.3. Πολυδιάστατοι πίνακες

Ένας πίνακας μπορεί να είναι μονοδιάστατος, αλλά στη γενικότερη περίπτωση μπορεί να είναι δισδιάστατος, τρισδιάστος και γενικά ν-διάστατος πίνακας. Όσον αφορά στους δισδιάστους πίνακες σημειώνεται ότι αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται *τετραγωνικός* (square) και γενικά συμβολίζεται ως πίνακας n x n.
Στους δισδιάστατους πίνακες χρησιμοποιούμε 2 δείκτες. Ο πρώτος αναφέρετε στην γραμμή στην οποία βρίσκετε το στοιχείο και ο δεύτερος στη στήλη. Π.χ.:

ΓΡΑΜΜΕΣ i (7)	ΣΤΗΛΕΣ j (5)				
	1	2	3	4	5
1	8	11	6	4	3
2	9	12	15	6	2
3	7	13	12	7	6
4	5	9	11	3	3
5	10	10	8	2	5
6	7	12	11	6	7
7	12	12	10	5	4

Π.χ.: Το στοιχείο Π[2,3] έχει τιμή 15 ενώ το Π[3,2] έχει τιμή 13.

Για την επεξεργασία των θερμοκρασιών π.χ. μπορεί να χρησιμοποιηθεί ένας *δισδιάστατος* πίνακας, στον οποίο ο πρώτος δείκτης δείχνει τη γραμμή (στο παράδειγμα την ημέρα) και ο δεύτερος τη στήλη (την πόλη). Τα ονόματα των πόλεων και των ημερών μπορεί να είναι αποθηκευμένα σε άλλους μονοδιάστατους πίνακες:

Η	Θ	Π: Αθήνα	Βόλος	Πάτρα	Λαμία	Χίος
		1	2	3	4	5
Δευτέρα	1	8	11	6	4	3
Τρίτη	2	9	12	15	6	2
Τετάρτη	3	7	13	12	7	6
Πέμπτη	4	5	9	11	3	3
Παρασκευή	5	10	10	8	2	5
Σαββάτο	6	7	12	11	6	7
Κυριακή	7	12	12	10	5	4

Πίνακες: Ημέρες, Πόλεις, Θερμοκρασίες (Η,Π,Θ)

- ***Προσπέλαση*** (access), πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.

Βασικό πρόγραμμα προσπέλασης:

```
ΠΡΟΓΡΑΜΜΑ Προσπέλαση
ΜΕΤΑΒΛΗΤΕΣ
        ! δήλωση μεγέθους και τύπου του πίνακα
        ΠΡΑΓΜΑΤΙΚΕΣ: Π[100,100], Χ
        ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
ΓΡΑΨΕ 'Δώσε πλήθος στοιχείων πίνακα (1-100, 1-100)'
ΔΙΑΒΑΣΕ Μ,Ν
! προσπέλαση τροποποίησης περιεχομένου κόμβων
! (διάβασμα - εισαγωγή στοιχείων του πίνακα)
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
                ΓΡΑΨΕ 'Δώσε το', i,j, ' στοιχείο του πίνακα'
                ΔΙΑΒΑΣΕ Π[i,j]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
Π[1,1]←8
! προσπέλαση εξέτασης περιεχομένου κόμβων
!(ανάθεση τιμών τους σε άλλη μεταβλητή εκτυπώσεις περιεχομένου κόμβων κ.λ.π. )
Χ←Π[1,1]+5
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
                ΓΡΑΨΕ  Π[i,j]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Π[2,2], Π[2+3,5], Π[Ν-5,1]
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Άσκηση 27:

Να γράψετε πρόγραμμα που θα διαβάζει τους πίνακες Η,Π,Θ με τα ονόματα των Ημερών, των Πόλεων και των Θερμοκρασιών. Όταν διαβάζει τις Θερμοκρασίες η Γράψε θα βγάζει το μήνυμα: Π.χ.
Δώσε θερμοκρασία στη πόλη Βόλος για την ημέρα Τρίτη:
Το πρόγραμμα σε μια του έκδοση θα διαβάζει πρώτα για την ίδια ημέρα τις θερμοκρασίες όλων των πόλεων, μετά για την επόμενη ημέρα τις θερμοκρασίες όλων των πόλεων κ.λ.π.
Σε μια δεύτερη έκδοση θα διαβάζει πρώτα για την ίδια πόλη τις θερμοκρασίες όλων των ημερών, μετά για την επόμενη πόλη κ.λ.π.

🛠 Υπολογισμός **αθροισμάτων** στοιχείων του πίνακα:

Άσκηση 28:
Να συμπληρώσετε το πρόγραμμα της άσκησης 27 ώστε να βρίσκει τον Μ.Ο. όλων των θερμοκρασιών όλων των ημερών όλων των πόλεων.

Πολλές φορές χρειάζεται να βρούμε ξεχωριστά τα αθροίσματα κάθε γραμμής ή κάθε σειράς. Τα αθροίσματα αυτά τα αποθηκεύουμε σε ξεχωριστούς μονοδιάστατους πίνακες:

ΓΡΑΜΜΕΣ i (7)	Π	ΣΤΗΛΕΣ j (5) 1	2	3	4	5	ΑΓ
	1	8	11	6	4	3	32
	2	9	12	15	6	2	44
	3	7	13	12	7	6	45
	4	5	9	11	3	3	31
	5	10	10	8	2	5	35
	6	7	12	11	6	7	43
	7	12	12	10	5	4	43
	ΑΣ	58	79	73	33	30	

Τμήμα προγράμματος που βρίσκει τα αθροίσματα κατά σειρά και κατά στήλη των στοιχείων ενός δισδιάστατου πίνακα.

! προσοχή μην ξεχνάς τον αρχικό μηδενισμό των πινάκων των αθροισμάτων

```
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΑΣ[i] ← 0
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΑΓ[j] ← 0
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
                ΑΣ[i] ← ΑΣ[i] + Π[i,j]
                ΑΓ[j] ← ΑΓ[j] + Π[i,j]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Άσκηση 29:
Να συμπληρώσετε την άσκηση 28 ώστε να βρίσκει και να τυπώνει την κάθε πόλη και τον εβδομαδιαίο Μ.Ο. των θερμοκρασιών της και την κάθε ημέρα και τον Μ.Ο. των θερμοκρασιών σε όλες τις πόλεις.

- 🛠 ***Αναζήτηση*** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.

Άσκηση 30:
Να συμπληρώσετε το πρόγραμμα της άσκησης 30 έτσι ώστε να βρίσκει και να τυπώνει την κάθε ημέρα και πόλη όπου η θερμοκρασία είναι κάτω από 10 βαθμούς. Αν δεν υπάρχει να βγάζει αντίστοιχο μήνυμα.

Άσκηση 31:
Αν μια πόλη για όλες τις ημέρες της εβδομάδας έχει θερμοκρασίες κάτω από το 0 τότε λέμε πως έχουμε ολικό παγετό. Να συμπληρώσετε το πρόγραμμα της άσκησης 30 ώστε να βρίσκει, αν υπάρχουν, τις πόλεις που είχαμε ολικό παγετό.
Υπόδειξη: Αρχικά υπέθεσε πως είναι μια η πόλη και ένας μονοδιάστατος πίνακας με τις θερμοκρασίες της. Λύσε το πρόβλημα αν έχουμε ολικό παγετό στη μια αυτή πόλη και μετά επέκτεινέ το όπως το ζητά η άσκηση.

- 🛠 Εύρεση του **μέγιστου** ή του **ελάχιστου** στοιχείου.

Άσκηση 32:
Να συμπληρώσετε το πρόγραμμα της άσκησης 31 έτσι ώστε να βρίσκει την μέγιστη θερμοκρασία καθώς και την ημέρα - πόλη (ή ημέρες - πόλεις) που έχουν αυτή τη μέγιστη θερμοκρασία.

Μερικές φορές χρειάζεται να βρούμε το μέγιστο (ή ελάχιστο) ξεχωριστά κάθε γραμμής ή στήλης. Αυτά τα αποθηκεύουμε σε ξεχωριστούς μονοδιάστατους πίνακες όπως και με τα αθροίσματα:

ΓΡΑΜΜΕΣ i (7)	Π	ΣΤΗΛΕΣ j (5) 1	2	3	4	5	ΜΓ
	1	8	11	6	4	3	11
	2	9	12	15	6	2	15
	3	7	13	12	7	6	13
	4	5	9	11	3	3	11
	5	10	10	8	2	5	10
	6	7	12	11	6	7	12
	7	12	12	10	5	4	12
ΜΣ		12	13	15	7	7	

Τμήμα προγράμματος που βρίσκει τα μέγιστα κατά σειρά και κατά στήλη των στοιχείων ενός δισδιάστατου πίνακα.

```
! προσοχή μην ξεχνάς την αρχική τιμή των πινάκων των μέγιστων
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΜΓ[i] ← Π[i,1]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΜΣ[j] ← Π[1,j]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ Ν
                ΑΝ ΜΓ[i]  > Π[i,j] ΤΟΤΕ
                        ΜΓ[i] ← Π[i,j]
                ΤΕΛΟΣ_ΑΝ
                ΑΝ ΜΣ[j]  > Π[i,j] ΤΟΤΕ
                        ΜΣ[j] ← Π[i,j]
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Άσκηση 33:
Να συμπληρώσετε το πρόγραμμα της άσκησης 32 ώστε να βρίσκει την μέγιστη θερμοκρασία της κάθε πόλης και να την τυπώνει καθώς και τις ημέρες που η πόλη είχε αυτή τη θερμοκρασία. Επίσης θα βρίσκει την μέγιστη θερμοκρασία της κάθε ημέρας και θα την τυπώνει μαζί με τις πόλεις που μετρήθηκε αυτή η μέγιστη θερμοκρασία.

- 🛠 ***Ταξινόμηση*** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

Άσκηση 34:
Να γράψετε προγράμματα τα οποία ταξινομούν έναν δισδιάστατο πίνακα Π μεγέθους ΜxΝ με τους παρακάτω τρόπους:

Οριζοντίως

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

Καθέτως

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

Κατά γραμμή

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

Κατά στήλη

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

9.4. Τυπικές επεξεργασίες πινάκων

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

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

Υπολογισμός αθροισμάτων στοιχείων του πίνακα.
Πολύ συχνά απαιτείται ο υπολογισμός του αθροίσματος στοιχείων του πίνακα που έχουν κοινά χαρακτηριστικά για παράδειγμα βρίσκονται στην ίδια στήλη ή στην ίδια γραμμή.

Εύρεση του μέγιστου ή του ελάχιστου στοιχείου.
Αν ο πίνακας δεν είναι ταξινομημένος, τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, για να βρεθεί το μέγιστο ή το ελάχιστο. Αν ο πίνακας είναι ταξινομημένος, τότε προφανώς το μέγιστο και το ελάχιστο βρίσκονται στα δύο ακριανά στοιχεία του πίνακα.

Ταξινόμηση των στοιχείων του πίνακα.
Στο κεφάλαιο 3 αναφέρθηκε η μέθοδος ταξινόμησης της ευθείας ανταλλαγής. Η μέθοδος αυτή είναι από τις απλούστερες αλλά δεν είναι η πιο αποδοτική. Υπάρχουν πολλές άλλες μέθοδοι ταξινόμησης καθώς και παραλλαγές αυτών. Η επιλογή του καλύτερου αλγόριθμου εξαρτάται κυρίως από το πλήθος των στοιχείων του πίνακα και την αρχική τους διάταξη, αν δηλαδή ο πίνακας είναι τελείως αταξινόμητος ή μερικώς ταξινομημένος.

Αναζήτηση ενός στοιχείου του πίνακα.
Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι αναζήτησης:
_ **Η σειριακή αναζήτηση**
_ **Η δυαδική αναζήτηση**
Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερη αποτελεσματική μέθοδος. Χρησιμοποιείται όμως υποχρεωτικά για πίνακες που δεν είναι ταξινομημένοι. Αντίθετα η **δυαδική αναζήτηση** χρησιμοποιείται μόνο σε **ταξινομημένους** πίνακες και είναι σαφώς **αποδοτικότερη** από τη σειριακή μέθοδο.

Συγχώνευση δύο πινάκων.
Η συγχώνευση είναι μία από τις βασικές λειτουργίες σε πίνακες.
Σκοπός της είναι η δημιουργία από τα στοιχεία δύο (ή περισσότερων) ταξινομημένων πινάκων ενός άλλου, που είναι και αυτός ταξινομημένος.

Δισδιάστατοι πίνακες με περιορισμένη την μια διάσταση

Πολλές φορές χρησιμοποιούμε δισδιάστατους πίνακες με περιορισμένη την μια διάσταση π.χ. Νx2 ή Νx3. Τους πίνακες αυτούς συνήθως τους μεταχειριζόμαστε σαν 2 ή 3 μονοδιάστατους.
Π.χ. σε έναν πίνακα 100x3 καταχωρούμε το Επώνυμο Όνομα και Πατρώνυμο μαθητών. Τον πίνακα αυτόν θα τον διαβάσουμε ως εξής:

```
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ 'Δώσε Επώνυμο, Όνομα και Πατρώνυμο:'
        Διάβασε Π[i,1], Π[i,2], Π[i,3]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Άσκηση 35:
Να ταξινομηθεί ο προηγούμενος πίνακας όπως στον τηλεφωνικό κατάλογο. Όπου υπάρχει συνωνυμία σε επώνυμο και όνομα ταξινομούμε με βάση το πατρώνυμο.

Άσκηση 36 ΘΕΜΑ 4^{Ο} 2002
Άσκηση 37 ΘΕΜΑ 4^{Ο} 2003
Άσκηση 38 ΘΕΜΑ 4^{Ο} 2004
Άσκηση 39 ΘΕΜΑ 4^{Ο} 2005
Άσκηση 40 ΘΕΜΑ 4^{Ο} 2006
Άσκηση 41 ΘΕΜΑ 4^{Ο} 2007
Άσκηση 42 ΘΕΜΑ 4^{Ο} 2008
Άσκηση 43 ΘΕΜΑ 4^{Ο} ΕΠΑΝΑΛ 2001
Άσκηση 44 ΘΕΜΑ 4^{Ο} ΕΣΠΕΡΙΝ 2006
Άσκηση 45 ΘΕΜΑ 4^{Ο} ΕΣΠΕΡΙΝ 2005
Άσκηση 46 ΘΕΜΑ 4^{Ο} ΕΣΠΕΡΙΝ ΕΠΑΝΑΛ 2005

Άσκηση 47: Να γράψετε πρόγραμμα το οποίο:

- Δηλώνει ένα πίνακα ακεραίων 10 θέσεων ο οποίος θα χρησιμοποιηθεί σαν στοίβα.
- Δημιουργεί μενού επιλογών με τις εξής επιλογές:
    1. push (ώθηση)
    2. pop (απώθηση)
    3. Έξοδος

Κάθε φορά που δίνουμε την επιλογή 1 προβαίνει στην αντίστοιχη ενέργεια και τυπώνει το κομμάτι του πίνακα που έχει στοιχεία της στοίβας. Αν έχουμε υπερχείλιση βγάζει μήνυμα, τυπώνει την στοίβα και τερματίζει. Κάθε φορά που δίνουμε την επιλογή 2 προβαίνει στην αντίστοιχη ενέργεια και τυπώνει το στοιχείο που απωθήσαμε και το κομμάτι του πίνακα που έχει στοιχεία της στοίβας. Αν έχουμε υποχείληση βγάζει μήνυμα αλλά δεν τερματίζει. Όταν δίνουμε την επιλογή 3 τυπώνει το κομμάτι του πίνακα που έχει στοιχεία της στοίβας και τερματίζει.

Άσκηση 48: Να γράψετε πρόγραμμα το οποίο:

- Δηλώνει ένα πίνακα ακεραίων 10 θέσεων ο οποίος θα χρησιμοποιηθεί σαν ουρά.
- Δημιουργεί μενού επιλογών με τις εξής επιλογές:
    1. enqueue (εισαγωγή)
    2. dequeue (εξαγωγή)
    3. Έξοδος

Κάθε φορά που δίνουμε την επιλογή 1 προβαίνει στην αντίστοιχη ενέργεια και τυπώνει το κομμάτι του πίνακα που έχει στοιχεία της ουράς. Αν τα στοιχεία της ουράς είναι περισσότερα από 10, βγάζει μήνυμα ότι δεν υπάρχει ελεύθερος χώρος, τυπώνει την ουρά και τερματίζει. Αν τα στοιχεία της ουράς είναι λιγότερα από 10 αλλά βρισκόμαστε στο τέλος του πίνακα, μεταφέρουμε όλα τα στοιχεία στην αρχή του και συνεχίζουμε κανονικά. Κάθε φορά που δίνουμε την επιλογή 2 προβαίνει στην αντίστοιχη ενέργεια και τυπώνει το στοιχείο που εξάγουμε και το κομμάτι του πίνακα που έχει στοιχεία της ουράς. Αν η ουρά είναι κενή βγάζει μήνυμα αλλά δεν τερματίζει. Όταν δίνουμε την επιλογή 3 τυπώνει το κομμάτι του πίνακα που έχει στοιχεία της ουράς και τερματίζει.

Άσκηση 49:

Ένας διαγωνισμός τραγουδιού στην Ευρώπη διεξάγεται ως εξής. Γίνεται μία πρώτη ακρόαση των τραγουδιών κάθε χώρας από την Κριτική Επιτροπή η οποία δίνει κάποιους βαθμούς σε κάθε τραγούδι (από 1- 100). Έστω ότι είναι γνωστοί οι βαθμοί που δόθηκαν στο τραγούδι κάθε χώρας. Να γραφεί ένας αλγόριθμος που θα επιλέγει για τη συνέχεια στη δεύτερη φάση του διαγωνισμού τις χώρες με τη μεγαλύτερη βαθμολογία κάθε φορά ώστε το άθροισμα της βαθμολογίας όλων των τραγουδιών που θα προχωρήσουν στη δεύτερη φάση να είναι μικρότερο από 1000 βαθμούς.

Άσκηση 50:

Να γράψετε τις εντολές που δίνουν τις ακόλουθες τιμές σε ένα πίνακα ακεραίων:

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Άσκηση 51:

Να γραφούν οι εντολές που ανταλλάσσουν τα στοιχεία της τρίτης και της έκτης στήλης σε ένα πίνακα ακεραίων 5Χ6.

Άσκηση 52:

Να γραφεί ένα πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό d και μία βάση μετατροπής b, όπου $2 \leq b \leq 16$ και να μετατρέπει τον αριθμό d σε σύστημα αρίθμησης με βάση b.

Άσκηση 53:
Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα των κυρίων διαγωνίων τετραγωνικού πίνακα ΝΧΝ. Επίσης θα βρίσκει το μέγιστο στοιχείο της κύριας διαγωνίου, το ελάχιστο της δευτερεύουσας διαγωνίου καθώς και τις θέσεις που αυτά βρίσκονται.

Άσκηση 54:
Οι παρακάτω πίνακες ονομάζονται τριγωνικοί. Να γράψετε πρόγραμμα που διαβάζει αυτούς τους πίνακες:

1	6	3	7	5
	2	4	1	7
		8	1	9
			4	8
				5

3				
1	4			
5	8	2		
6	3	7	1	

1	8	2	7
5	4	2	
3	4		
1			

			5
		4	3
	1	4	6

Άσκηση 55:
Δίνονται οι πίνακες Σ1(Κ,Κ) και Π1(Κ,Κ) που περιέχουν τα αποτελέσματα των αγώνων ομίλου του EuroBasket. Ο πίνακας Σ1 περιέχει τα αποτελέσματα των αγώνων (Ν (νίκη) ή Η (ήττα)), ενώ ο πίνακας Π1 τη διαφορά πόντων για κάθε αγώνα. Ο πίνακας Ο1(Κ) περιέχει τα ονόματα των ομάδων.
Να γραφεί πρόγραμμα το οποίο θα βρίσκει και θα εκτυπώνει την τελική βαθμολογία του ομίλου σε φθίνουσα σειρά. Σε περίπτωση ισοβαθμίας προηγείται η ομάδα που έχει την καλύτερη διαφορά πόντων από τις ισόβαθμές της.
Τα στοιχεία της κύριας διαγωνίου δεν περιέχουν καμία πληροφορία (καμία ομάδα δεν παίζει με τον εαυτό της!).
Ο πίνακας περιέχει στοιχεία μόνο κάτω ή πάνω από τη διαγώνιο του, είναι δηλαδή τριγωνικός (κάθε ομάδα παίζει μόνο μία φορά με κάθε αντίπαλο).

Άσκηση 56:
Να γραφεί πρόγραμμα το οποίο να δέχεται δύο τετραγωνικούς δισδιάστατους πίνακες και να υπολογίζει το άθροισμα και το γινόμενο τους.
Υπόδειξη: Αν a και b είναι οι αρχικοί πίνακες και c ο τελικός, τότε ισχύει:

Πρόσθεση: $c_{ij} = a_{ij} + b_{ij}$

Πολ/σμός: $c_{ij} = \sum_{k=1}^{n} a_{ik} . b_{kj}$

Αραιοί πίνακες

Ένας πίνακας λέγεται ***αραιός*** (sparse) αν ένα μεγάλο ποσοστό των στοιχείων του έχουν μηδενική τιμή. Δεν υπάρχει ακριβές ποσοστό σε σχέση με τον αριθμό των μηδενικών στοιχείων, επάνω από το οποίο ένας πίνακας χαρακτηρίζεται ως αραιός. Αρκεί όμως, για παράδειγμα, να πούμε ότι με περισσότερο από 80% μηδενικά ένας πίνακας χαρακτηρίζεται ως αραιός.
Αραιοί πίνακες συναντώνται συχνά σε μεγάλα επιστημονικά προβλήματα (επίλυση εξισώσεων κλπ). Το πρόβλημα με τη διαχείριση των αραιών πινάκων είναι ότι δαπανάται πολύ χώρος για την αποθήκευση μηδενικών. Άρα πρέπει να βρεθεί ένας οικονομικός τρόπος αποθήκευσης των αραιών πινάκων. Στην πράξη έχουν προταθεί αρκετοί τρόποι. Ένας από αυτούς τους τρόπος περιγράφεται στη συνέχεια. Έστω, λοιπόν, ότι δίνεται ο επόμενος πίνακας, που θέλουμε να τον διαχειρισθούμε ως αραιό.

Δ

	1	2	3	4	5	6
1	0	0	6	0	0	0
2	7	0	0	0	1	0
3	0	0	0	0	0	0
4	0	0	5	0	0	2
5	0	0	0	3	0	0

Αντί να αποθηκεύσουμε αυτόν το δισδιάστατο πίνακα 5x6, θα θεωρήσουμε ένα μονοδιάστατο πίνακα όπου θα τοποθετήσουμε μόνο τα μη μηδενικά στοιχεία, για τα οποία όμως χρειαζόμαστε τα στοιχεία των αντίστοιχων γραμμών και στηλών. Έτσι καταλήγουμε κάθε μη μηδενικό στοιχείο να αντιπροσωπεύεται από μία τριάδα στοιχείων, δηλαδή <γραμμή, στήλη, τιμή>. Για το λόγο αυτό δημιουργούμε ένα μονοδιάστατο πίνακα 18 θέσεων για τα 6 μη μηδενικά στοιχεία του αρχικού πίνακα. Ο νέος πίνακας έχει τη μορφή:

Μ

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	3	**6**	*2*	1	**7**	*2*	5	**1**	*4*	3	**5**	*4*	6	**2**	*5*	4	**3**

Άσκηση 57: (Δημιουργία αραιού πίνακα)
Να γράψετε ένα πρόγραμμα το οποίο δεδομένου του πίνακα Δ συμπληρώνει τις πρώτες θέσεις του πίνακα Μ[100] με τα μη μηδενικά στοιχεία. Το πλήθος των μη μηδενικών στοιχείων είναι άγνωστο και απλά ελπίζουμε να μην είναι πάνω από 33.
Επίσης να γράψετε ένα άλλο πρόγραμμα το οποίο διαβάζει απ ευθείας τον πίνακα Μ ζητώντας κάθε φορά από τον χρήστη να εισάγει την τριάδα γραμμή, στήλη και τιμή. Εδώ το πλήθος των πραγματικών στοιχείων είναι γνωστό π.χ. 6.

Άσκηση 58: (Εύρεση αθροισμάτων)
Να γράψετε πρόγραμμα το οποίο από τον πίνακα Μ θα βρίσκει:

1. Το άθροισμα όλων των στοιχείων του πίνακα Δ.
2. Τα αθροίσματα της κάθε μη μηδενικής γραμμής του Δ τα οποία και θα αποθηκεύονται σε ένα πίνακα ΑΓ Κx2 όπου η κάθε θέση ΑΓ[x,1] περιέχει την γραμμή του Δ ενώ η κάθε θέση ΑΓ[x,2] το άθροισμα των στοιχείων της γραμμής.
3. Το αντίστοιχο για τις στήλες του Δ.

Άσκηση 59: (Αναζήτηση)
Να γράψετε πρόγραμμα το οποίο από τον πίνακα Μ θα βρίσκει:

1. Θα διαβάζει γραμμή και στήλη του Δ και από τον Μ θα βρίσκει και θα τυπώνει το στοιχείο του Δ. (Προσοχή θα τυπώνει ακόμα και τα μηδενικά στοιχεία).
2. Θα διαβάζει μια τιμή και από τον Μ θα βρίσκει αν αυτή η τιμή υπάρχει στον Δ και αν ναι πόσες φορές και σε ποιες θέσεις.

Άσκηση 60: (Μέγιστα ελάχιστα)
Να γράψετε πρόγραμμα το οποίο από τον πίνακα Μ θα βρίσκει:

1. Το μέγιστο στοιχείο του Δ πόσες φορές και σε ποιες θέσεις εμφανίζεται.
2. Το ελάχιστο στοιχείο κάθε μη μηδενικής γραμμής.

Δομές Δεδομένων δευτερεύουσας μνήμης

Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα μνήμη, δηλαδή κυρίως στο μαγνητικό δίσκο. Οι ειδικές αυτές δομές ονομάζονται ***αρχεία*** (files). Είναι γνωστό ότι μία σημαντική διαφορά μεταξύ κύριας μνήμης και μαγνητικού δίσκου είναι ότι στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή. Έτσι, τα δεδομένα των αρχείων διατηρούνται ακόμη και μετά τον τερματισμό ενός προγράμματος, κάτι που δεν συμβαίνει στην περίπτωση των δομών της κύριας μνήμης, όπως είναι οι πίνακες, όπου τα δεδομένα χάνονται όταν τελειώσει το πρόγραμμα. Τα στοιχεία ενός αρχείου ονομάζονται ***εγγραφές*** (records), όπου κάθε εγγραφή αποτελείται από ένα ή περισσότερα *πεδία* (fields), που ταυτοποιούν την εγγραφή, και από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής. Για παράδειγμα, έστω η εγγραφή ενός μαθητή με πεδία: Αριθμός Μητρώου, Ονοματεπώνυμο, 'Ετος Γέννησης, Τάξη, Τμήμα. Το πεδίο Αριθμός Μητρώου ταυτοποιεί την εγγραφή και ονομάζεται *πρωτεύον κλειδί* (primary key) ή απλά κλειδί. Το πεδίο Ονοματεπώνυμο επίσης ταυτοποιεί την εγγραφή και γι' αυτό αποκαλείται *δευτερεύον κλειδί* (secondary keys), αν υπάρχει πρωτεύον κλειδί. Το πρόβλημα της *αναζήτησης* (searching) μίας εγγραφής με βάση την τιμή του πρωτεύοντος ή ενός δευτερεύοντος κλειδιού σε αρχεία είναι ιδιαίτερα ενδιαφέρον, αν ληφθεί υπ' όψη η μεγάλη ποικιλία των χαρακτηριστικών τόσο της δομής (για παράδειγμα, στατική ή δυναμική, τρόπος οργάνωσης, μέσο αποθήκευσης κ.λπ.), του τύπου των δεδομένων (για παράδειγμα, ακέραιοι, κείμενο, χαρτογραφικά δεδομένα, χρονοσειρές κ.λπ.), όσο και της αναζήτησης (δηλαδή, με βάση το πρωτεύον ή το δευτερεύον κλειδί κλπ.).

ΚΕΦΑΛΑΙΟ 10

10.1. Τμηματικός προγραμματισμός

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των 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** (subprogram).

10.2. Χαρακτηριστικά των υποπρογραμμάτων

Ο χωρισμός ενός προγράμματος σε **υποπρογράμματα** προϋποθέτει την **ανάλυση** του αρχικού προβλήματος σε μικρότερα **υποπροβλήματα**, τα οποία να μπορούν να αντιμετωπισθούν **ανεξάρτητα** το ένα από το άλλο. Υπάρχουν πάντως τρεις ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα:

1. **Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο.** Στην πραγματικότητα κάθε υποπρόγραμμα ενεργοποιείται με την **είσοδο** σε αυτό που γίνεται πάντοτε από την **αρχή** του, εκτελεί ορισμένες ενέργειες, και απενεργοποιείται με την **έξοδο** από αυτό που γίνεται πάντοτε από το **τέλος** του.
2. **Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα**. Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί **αυτόνομα** χωρίς να επηρεαστούν άλλα υποπρογράμματα. Στην πράξη βέβαια η απόλυτη ανεξαρτησία είναι δύσκολο να επιτευχθεί.
3. **Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο**. Η έννοια του μεγάλου προγράμματος είναι υποκειμενική, αλλά πρέπει κάθε υποπρόγραμμα να είναι τόσο, ώστε να είναι **εύκολα κατανοητό για να μπορεί να ελέγχεται**. Γενικά κάθε υποπρόγραμμα πρέπει να εκτελεί **μόνο μία λειτουργία**. Αν εκτελεί περισσότερες λειτουργίες, τότε συνήθως μπορεί και πρέπει να διασπαστεί σε ακόμη μικρότερα υποπρογράμματα.

10.3. Πλεονεκτήματα του τμηματικού προγραμματισμού

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

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

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

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

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

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

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

Ένα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί πολύ εύκολα και σε άλλα προγράμματα. Από τη στιγμή που έχει δημιουργηθεί, η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων που παρέχει η γλώσσα προγραμματισμού, όπως για τον υπολογισμό του ημίτονου ή του συνημίτονου ή την εντολή με την οποία εκτελεί μία συγκεκριμένη διαδικασία, για παράδειγμα γράφει στην οθόνη (εντολή ΓΡΑΨΕ). Αν λοιπόν χρειάζεται συχνά κάποια λειτουργία που δεν υποστηρίζεται απευθείας από τη γλώσσα, όπως για παράδειγμα η ανταλλαγή τιμών δύο αριθμών, τότε μπορεί να γραφεί το αντίστοιχο υποπρόγραμμα. Η συγγραφή πολλών υποπρογραμμάτων και η δημιουργία βιβλιοθηκών με αυτά, ουσιαστικά επεκτείνουν την ίδια τη γλώσσα προγραμματισμού.

10.4. Παράμετροι

Κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται, όπως λέγεται, από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα, το οποίο ονομάζεται ***κύριο πρόγραμμα***.
Το υποπρόγραμμα είναι αυτόνομο και ανεξάρτητο τμήμα προγράμματος, αλλά συχνά πρέπει να **επικοινωνεί** με το υπόλοιπο πρόγραμμα. Συνήθως δέχεται τιμές από το τμήμα προγράμματος που το καλεί και μετά την εκτέλεση επιστρέφει σε αυτό νέες τιμές, αποτελέσματα.
Οι τιμές αυτές που περνούν από το ένα υποπρόγραμμα στο άλλο λέγονται ***παράμετροι***.
Οι παράμετροι λοιπόν είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μία ουσιώδη διαφορά, χρησιμοποιούνται για να περνούν τιμές στα υποπρογράμματα.
Μία **παράμετρος** είναι μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.
Παραδείγματα παραμέτρων, καθώς και ο τρόπος που περνούν οι τιμές προς και από το υποπρόγραμμα, θα δοθούν στη συνέχεια.

10.5. Διαδικασίες και συναρτήσεις

Υπάρχουν δύο ειδών υποπρογράμματα, οι **διαδικασίες** και οι **συναρτήσεις**. Το είδος κάθε υποπρογράμματος καθορίζεται από το είδος της λειτουργίας που καλείται να επιτελέσει.
Οι **διαδικασίες** μπορούν να εκτελέσουν **οποιαδήποτε λειτουργία** από αυτές που μπορεί να εκτελέσει ένα πρόγραμμα. Να εισάγουν δεδομένα, να εκτελέσουν υπολογισμούς, να μεταβάλλουν τις τιμές των μεταβλητών και να τυπώσουν αποτελέσματα. Με τη χρήση των παραμέτρων αυτές τις τιμές μπορούν να τις μεταφέρουν και στα άλλα υποπρογράμματα.
Αντίθετα η λειτουργία των **συναρτήσεων** είναι πιο **περιορισμένη**. Οι συναρτήσεις υπολογίζουν **μόνο μία τιμή**, αριθμητική, χαρακτήρα ή λογική και μόνο αυτήν επιστρέφουν στο υποπρόγραμμα που την κάλεσε. Οι συναρτήσεις μοιάζουν με τις συναρτήσεις των μαθηματικών και η χρήση τους είναι όμοια με τη χρήση των ενσωματωμένων συναρτήσεων που υποστηρίζει η γλώσσα προγραμματισμού.
Ο **τρόπος κλήσης** καθώς και ο **τρόπος σύνταξης** των δύο αυτών τύπων των υποπρογραμμάτων είναι **διαφορετικός**. Τόσο οι συναρτήσεις όσο και οι διαδικασίες **τοποθετούνται μετά το τέλος του κυρίου προγράμματος**.
Οι **συναρτήσεις εκτελούνται απλά με την εμφάνιση του ονόματος τους σε οποιαδήποτε έκφραση**, ενώ για να εκτελεστούν οι **διαδικασίες** χρησιμοποιείται η **ειδική εντολή ΚΑΛΕΣΕ** και το όνομα της διαδικασίας.
Η **συνάρτηση** είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομά της (όπως οι μαθηματικές συναρτήσεις).
Η **διαδικασία** είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος.

Παράδειγμα 2

Να γραφεί πρόγραμμα, το οποίο υπολογίζει το εμβαδό του κύκλου από την ακτίνα του.

Το πρόγραμμα εκτελεί τρεις συγκεκριμένες απλές λειτουργίες.

α) Διαβάζει τα δεδομένα, την ακτίνα η οποία πρέπει να είναι θετικός αριθμός

β) Υπολογίζει το εμβαδόν ($E=\pi r^2$)

γ) Τυπώνει το αποτέλεσμα, το εμβαδόν, Ε

Αν και το πρόγραμμα είναι πολύ απλό και μπορεί κάλλιστα να γραφεί χωρίς τη χρήση υποπρογραμμάτων, ας το διασπάσουμε σε τρία υποπρογράμματα που εκτελούν τις τρεις παραπάνω λειτουργίες.

Το κύριο πρόγραμμα που καλεί όλα τα υποπρογράμματα έχει ως εξής:

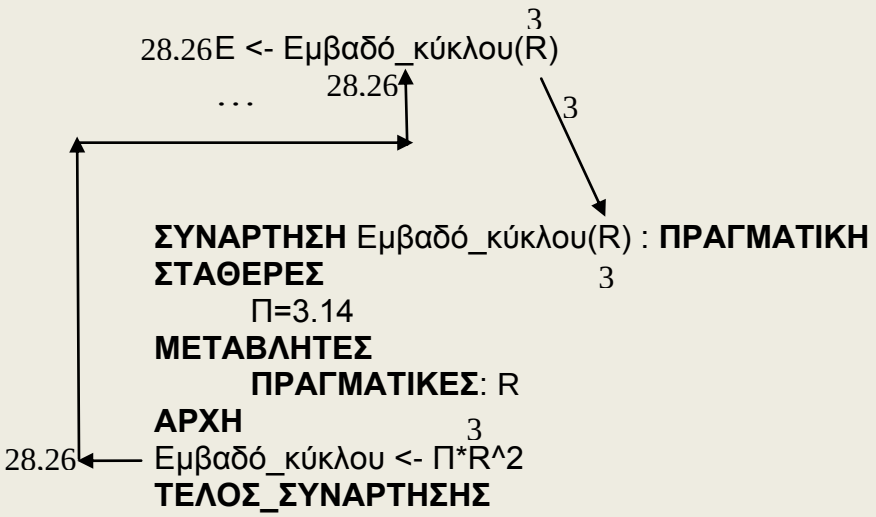
```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα_2
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ : R, E
ΑΡΧΗ
ΚΑΛΕΣΕ Είσοδος_δεδομένων(R)
E <- Εμβαδό_κύκλου(R)
ΚΑΛΕΣΕ Εκτύπωση(E)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ Είσοδος_δεδομένων(Αριθμός)
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ : Αριθμός
ΑΡΧΗ
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'Δώσε την ακτίνα'
ΔΙΑΒΑΣΕ Αριθμός
ΜΕΧΡΙΣ_ΟΤΟΥ Αριθμός>0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΣΥΝΑΡΤΗΣΗ Εμβαδό_κύκλου(R) : ΠΡΑΓΜΑΤΙΚΗ
ΣΤΑΘΕΡΕΣ
        Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: R
ΑΡΧΗ
Εμβαδό_κύκλου <- Π*R^2
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΔΙΑΔΙΚΑΣΙΑ Εκτύπωση(Αποτέλεσμα)
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ : Αποτέλεσμα
ΑΡΧΗ
ΓΡΑΨΕ 'Το εμβαδό του κύκλου είναι :',Αποτέλεσμα
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

10.5.1 Ορισμός και κλήση συναρτήσεων

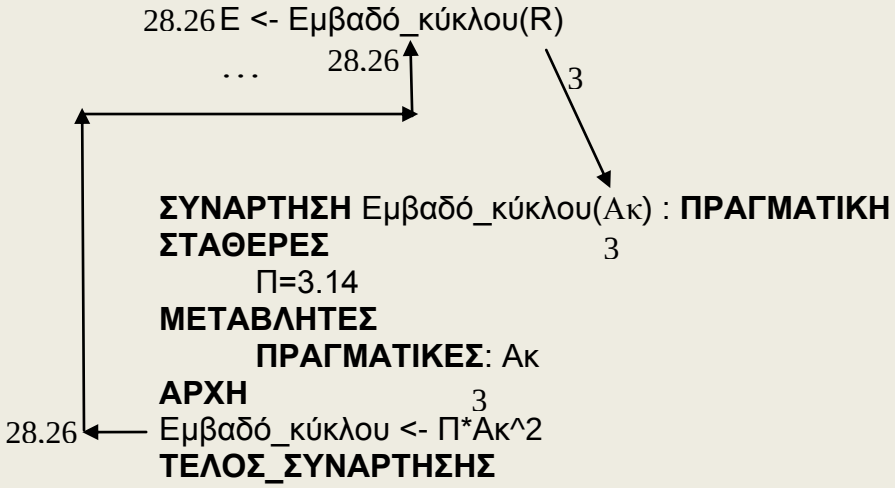
Κάθε συνάρτηση έχει την ακόλουθη δομή.

ΣΥΝΑΡΤΗΣΗ όνομα_συνάρτησης(λίστα παραμέτρων):τύπος συνάρτησης
Τμήμα δηλώσεων
ΑΡΧΗ
....
όνομα_συνάρτησης <- έκφραση
...
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της **ΓΛΩΣΣΑΣ**. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται στη συνάρτηση κατά την κλήση.
Οι συναρτήσεις μπορούν να επιστρέφουν τιμές όλων των τύπων δεδομένων που υποστηρίζει η γλώσσα. Μια συνάρτηση λοιπόν μπορεί να είναι ΠΡΑΓΜΑΤΙΚΗ, ΑΚΕΡΑΙΑ, ΧΑΡΑΚΤΗΡΑΣ, ΛΟΓΙΚΗ
Στις εντολές του σώματος της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μία εντολή εκχώρησης τιμής στο όνομα της συνάρτησης, στο προηγούμενο παράδειγμα Εμβαδό_κύκλου<-Π*R^2.
Κάθε συνάρτηση εκτελείται (καλείται), όπως ακριβώς εκτελούνται οι ενσωματωμένες συναρτήσεις της γλώσσας. Απλώς αναφέρεται το όνομα της σε μια έκφραση ή σε μία εντολή και επιστρέφεται η τιμή της. Στο παράδειγμα η συνάρτηση εκτελείται με την εντολή Ε<-Εμβαδό_κύκλου(R).
Ο μηχανισμός που επιτυγχάνεται αυτό, είναι ο εξής: Το κύριο πρόγραμμα πριν την κλήση της συνάρτησης γνωρίζει την τιμή της μεταβλητής R. Κατά την κλήση μεταβιβάζεται αυτή η τιμή στην αντίστοιχη μεταβλητή της συνάρτησης. Η συνάρτηση υπολογίζει το εμβαδόν του κύκλου και το αποτέλεσμα αυτό εκχωρείται στο όνομα της συνάρτησης. Με το τέλος της συνάρτησης γίνεται επιστροφή στο κύριο πρόγραμμα, όπου η τιμή του εμβαδού εκχωρείται στη μεταβλητή Ε.

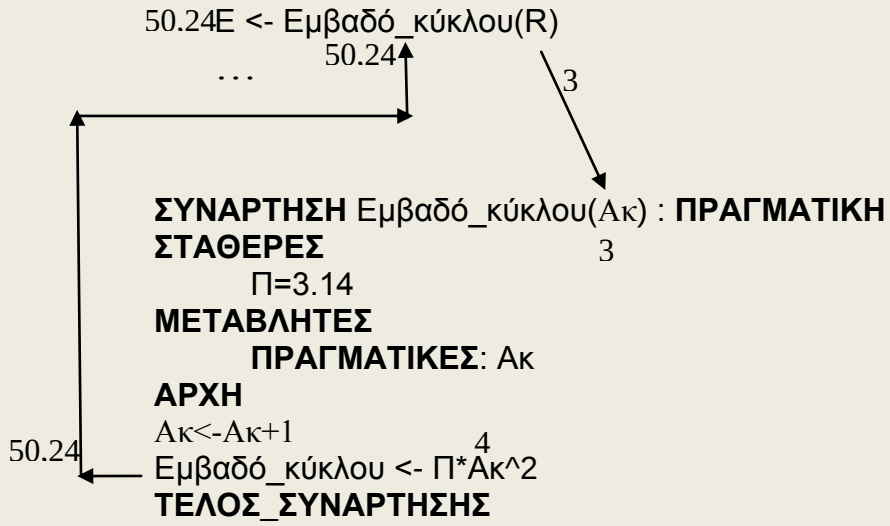


Η R στο κυρίως πρόγραμμα και η R στο σώμα της συνάρτησης είναι στη πραγματικότητα 2 διαφορετικές μεταβλητές, δυο διαφορετικές περιοχές της μνήμης οι οποίες είναι απλώς συνώνυμες, και κάποια στιγμή, με την κλήση της συνάρτησης η πρώτη μεταβιβάζει την τιμή της στη δεύτερη.
Η μεταβλητή (παράμετρος) R της συνάρτησης θα μπορούσε να έχει διαφορετικό όνομα:



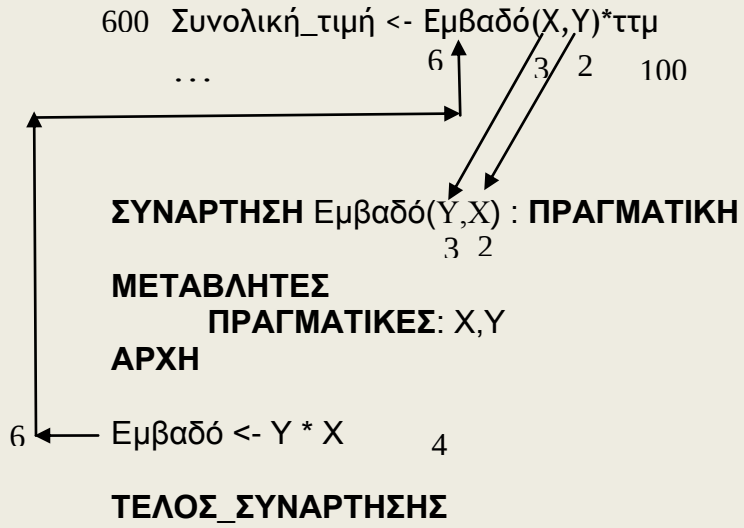
Εφόσον η συνάρτηση δηλώθηκε ΠΡΑΓΜΑΤΙΚΗ τότε και η μεταβλητή Ε στο κυρίως πρόγραμμα πρέπει να δηλωθεί ίδιου τύπου άρα ΠΡΑΓΜΑΤΙΚΗ.
Επίσης και η παράμετρος της συνάρτησης (R ή Ακ) πρέπει να είναι ίδιου τύπου με την μεταβλητή R του κυρίως προγράμματος.

Αν επίτηδες ή από λάθος η τιμή της παραμέτρου αλλάξει μέσα στο σώμα της συνάρτησης η αλλαγή αυτή δεν μεταβιβάζεται πίσω στην αντίστοιχη μεταβλητή.



Το κυρίως πρόγραμμα δεν αναγνωρίζει την Ακ και η συνάρτηση δεν αναγνωρίζει την R. (Αν έχουν το ίδιο όνομα R το καθένα αναγνωρίζει μόνο την δική του R.)

Η παρακάτω συνάρτηση υπολογίζει το εμβαδόν ενός τετραγωνικού οικοπέδου και το κυρίως πρόγραμμα την Συνολική_τιμή του όπου ττμ είναι η τιμή του ενός τετραγωνικού μέτρου:



Όταν έχουμε περισσότερες από μια παραμέτρους το πέρασμα των τιμών γίνεται με βάση την σειρά (η πρώτη στην πρώτη, η δεύτερη στη δεύτερη κ.λπ.) κι όχι με βάση το όνομα. Έτσι η μεταβλητή Χ του προγράμματος περνάει την τιμή της στη παράμετρο Υ της συνάρτησης γιατί και οι δύο είναι οι πρώτες στη σειρά.

Στο παρακάτω παράδειγμα η συνάρτηση άθροισμα(Ν) όπου Ν θετικός ακέραιος βρίσκει και επιστρέφει το άθροισμα 1+2+...+Ν.

```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα_συνάρτησης
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ : Ν,Α
ΑΡΧΗ
ΓΡΑΨΕ "ΔΩΣΕ ΕΝΑ ΘΕΤΙΚΟ ΑΚΕΡΑΙΟ:"
ΔΙΑΒΑΣΕ Ν
ΕΚΤΥΠΩΣΕ "ΤΟ ΑΘΡΟΙΣΜΑ ΕΙΝΑΙ:", Άθροισμα(Ν)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ Άθροισμα(Ν) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ: Α,Ν,Ι
ΑΡΧΗ
Α←0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
        Α←Α+Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
Άθροισμα←Α
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Μέσα σε μια συνάρτηση μπορούμε να χρησιμοποιήσουμε και άλλες μεταβλητές. Ο τύπος τους πρέπει να δηλωθεί στο τμήμα μεταβλητών μαζί με την δήλωση των παραμέτρων. Οι μεταβλητές αυτές είναι άγνωστες στο κυρίως πρόγραμμα καθώς και σε άλλα υποπρογράμματα αν υπάρχουν. Αν π.χ. στο κυρίως πρόγραμμα σαν τελευταία εντολή δίναμε: ΓΡΑΨΕ Α,Ι θα είχαμε μήνυμα λάθους γιατί σ αυτό δεν δηλώσαμε ούτε δώσαμε τιμή σε καμία μεταβλητή Α ή Ι.

Άσκηση 1:
Να γράψετε κυρίως πρόγραμμα και συνάρτηση που κάνουν τα κάτωθι:
Το κυρίως πρόγραμμα διαβάζει 2 πραγματικούς αριθμούς χρησιμοποιεί την συνάρτηση (max(Χ,Υ)) και τυπώνει τον μεγαλύτερο από αυτούς.
Η συνάρτηση max(Χ,Υ) βρίσκει και επιστρέφει τον μεγαλύτερο από τους δύο αριθμούς Χ και Υ.

Άσκηση 2:
Να γράψετε κυρίως πρόγραμμα και συνάρτηση που κάνουν τα κάτωθι:
Το κυρίως πρόγραμμα διαβάζει 1 ακέραιο αριθμό >=0 χρησιμοποιεί την συνάρτηση (παραγοντικό(Χ)) και τυπώνει το παραγοντικό του.
Η συνάρτηση παραγοντικό(Χ) επιστρέφει το παραγοντικό του Χ.

Άσκηση 3:
Να γράψετε συνάρτηση η οποία ελέγχει αν ένας αριθμός είναι άρτιος.

Άσκηση 4:
Τι θα τυπωθεί αν εκτελέσουμε το παρακάτω πρόγραμμα:

```
ΠΡΟΓΡΑΜΜΑ Τι_θα_τυπωθεί
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ : Α,Β,Γ,Δ
ΑΡΧΗ
Α→1
Β→2
Γ→3
Δ→συναρ(Α,Β,Γ)
ΕΚΤΥΠΩΣΕ Α,Β,Γ,Δ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ συναρ(Β,Α,Χ) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ: Α,Β,Γ,Δ,Χ
ΑΡΧΗ
Α←0
Γ←8
Χ←4
Δ←5
ΕΚΤΥΠΩΣΕ Α,Β,Γ,Δ,Χ
συναρ←Α+Β+Χ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

10.5.2 Ορισμός και κλήση διαδικασιών

Κάθε διαδικασία έχει την ακόλουθη δομή.

ΔΙΑΔΙΚΑΣΙΑ Όνομα (λίστα παραμέτρων)
    Τμήμα δηλώσεων
ΑΡΧΗ
εντολές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Το **όνομα** της διαδικασίας είναι οποιοδήποτε έγκυρο όνομα της **ΓΛΩΣΣΑΣ**.
Η **λίστα παραμέτρων** είναι μια λίστα μεταβλητών, των οποίων οι τιμές **μεταβιβάζονται** προς τη διαδικασία κατά την κλήση ή/και **επιστρέφονται** στο κύριο πρόγραμμα μετά το τέλος της διαδικασίας. Στη γενική περίπτωση μπορούν να υπάρχουν καμία, μία ή περισσότερες παράμετροι. Όταν υπάρχουν πολλές παράμετροι, τότε άλλες χρησιμοποιούνται για να μεταβιβάσουν τιμές στη διαδικασία και άλλες για να επιστρέψουν τιμές στο κύριο πρόγραμμα.
Στο σώμα της διαδικασίας μπορούν να υπάρχουν οποιεσδήποτε εντολές της γλώσσας.
Κάθε διαδικασία εκτελείται όταν καλείται από το κύριο πρόγραμμα ή άλλη διαδικασία. Η **κλήση** σε διαδικασία πραγματοποιείται με την εντολή **ΚΑΛΕΣΕ**, που ακολουθείται από το όνομα της διαδικασίας συνοδευόμενο μέσα σε παρενθέσεις με τη λίστα παραμέτρων.
Η γενική μορφή της εντολής **ΚΑΛΕΣΕ** είναι:

Σύνταξη
ΚΑΛΕΣΕ όνομα-διαδικασίας (λίστα-παραμέτρων)

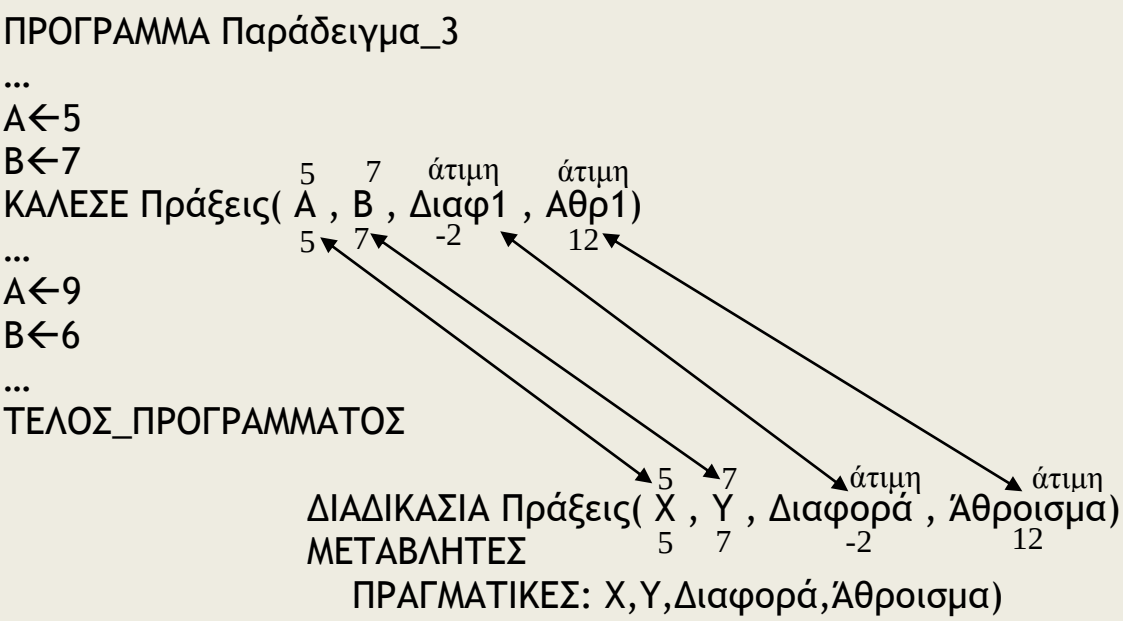
Λειτουργία
Η εκτέλεση του προγράμματος διακόπτεται και εκτελούνται οι εντολές της διαδικασίας που καλείται. Μετά το τέλος της διαδικασίας η εκτέλεση του προγράμματος συνεχίζεται από την εντολή που ακολουθεί. Η λίστα των παραμέτρων ορίζει τις τιμές που περνούν στη διαδικασία και τις τιμές που αυτή επιστρέφει. Η λίστα παραμέτρων δεν είναι υποχρεωτική.

Παράδειγμα
ΚΑΛΕΣΕ Πράξεις (Α, Β, Διαφορά)

Κάθε διαδικασία ή συνάρτηση μπορεί να **καλείται** από το **κύριο πρόγραμμα** ή **άλλη διαδικασία ή συνάρτηση**. Σε κάθε περίπτωση μετά το τέλος της εκτέλεσης της διαδικασίας γίνεται επιστροφή ακριβώς μετά το σημείο απ' όπου κλήθηκε.

10.5.3 Πραγματικές και τυπικές παράμετροι

Η κατανόηση του τρόπου που γίνεται η ανταλλαγή των τιμών ανάμεσα στις παραμέτρους είναι ιδιαίτερα σημαντική και γι αυτό ας παρακολουθήσουμε το επόμενο παράδειγμα.

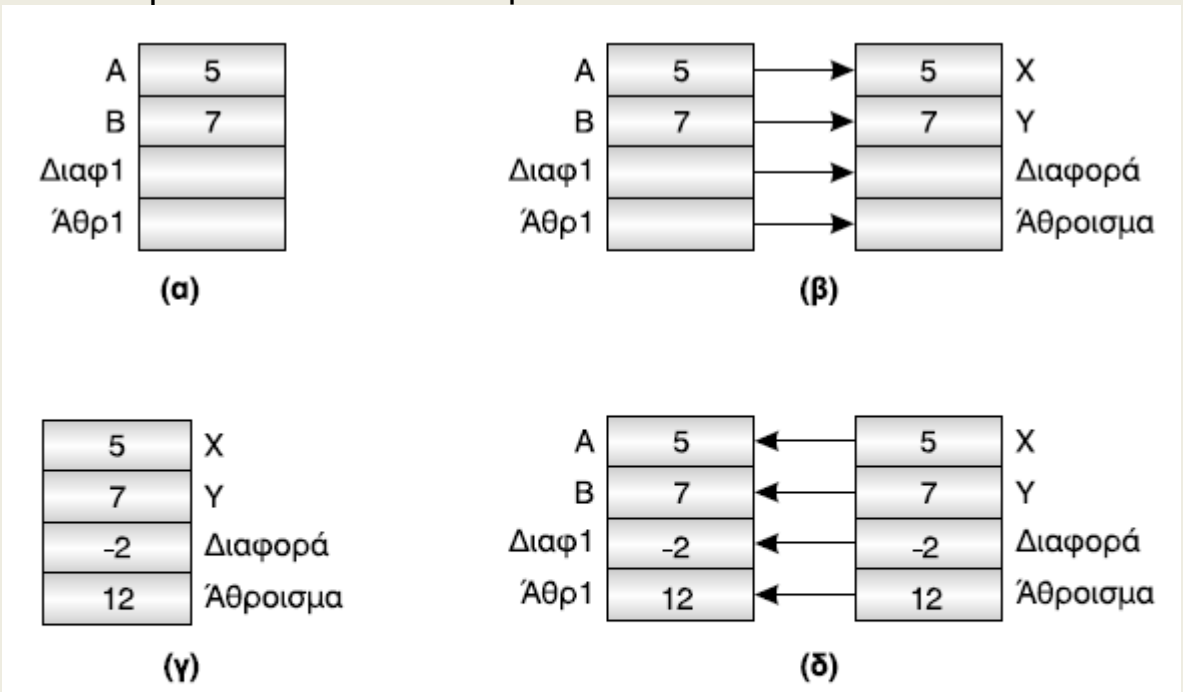


Οι μεταβλητές Α, Β, Διαφ1, Αθρ1, Α,Β, Διαφ2, Αθρ2 είναι μεταβλητές του προγράμματος Παράδειγμα_3 και αποτελούν τις **πραγματικές** παραμέτρους, ενώ οι μεταβλητές Χ,Υ, Διαφορά, Άθροισμα είναι μεταβλητές της διαδικασίας Πράξεις, και ονομάζονται **τυπικές** παράμετροι.

Οι μεταβλητές Χ,Υ, Διαφ1 καθώς και όλες οι μεταβλητές του προγράμματος Παράδειγμα_3 δεν είναι γνωστές στη διαδικασία Πράξεις και αντίστοιχα όλες οι μεταβλητές της διαδικασίας Πράξεις είναι άγνωστες στο πρόγραμμα Παράδειγμα_3. Τα ονόματα των τυπικών και των πραγματικών παραμέτρων μπορούν να είναι οποιαδήποτε. Αφού είναι ονόματα μεταβλητών σε διαφορετικά τμήματα προγράμματος, είναι υποχρεωτικά διαφορετικές μεταβλητές, άσχετα αν έχουν το ίδιο όνομα.

Όλες οι μεταβλητές είναι γνωστές, έχουν ισχύ όπως λέγεται, μόνο για το τμήμα προγράμματος στο οποίο έχουν δηλωθεί, ισχύουν δηλαδή **τοπικά** για το συγκεκριμένο υποπρόγραμμα ή κυρίως πρόγραμμα.

Ας παρακολουθήσουμε πώς γίνεται η επικοινωνία ανάμεσα στο πρόγραμμα Παράδειγμα_3 και τη διαδικασία Πράξεις.
Οι τιμές που υπάρχουν στις μεταβλητές του προγράμματος Α, Β, Διαφ1 και Αθρ1 δίνονται κατά την κλήση στις μεταβλητές της διαδικασίας Χ, Υ, Διαφορά, Άθροισμα.
Έτσι η μεταβλητή Χ παίρνει την τιμή 5 κι η Υ την τιμή 7. Οι μεταβλητές Διαφορά και Άθροισμα δεν παίρνουν καμία τιμή, αφού οι αντίστοιχες μεταβλητές Διαφ1 και Αθρ1 δεν έχουν συγκεκριμένη τιμή.
Μετά την εκτέλεση των εντολών της διαδικασίας, όταν εκτελεστεί η εντολή ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ, οι μεταβλητές της διαδικασίας που αναφέρονται στη δήλωση της διαδικασίας δίνουν τις τιμές που περιέχουν στις αντίστοιχες μεταβλητές που περιλαμβάνονται στην κλήση της διαδικασίας Πράξεις. Έτσι η Α παίρνει την τιμή της Χ (=5), η Β την τιμή της Υ (= 7), η Διαφ1 της Διαφορά (=-2) και η μεταβλητή Άθρ1 της Άθροισμα (=12). Με την επιστροφή στο κύριο πρόγραμμα όλες οι θέσεις μνήμης που είχαν δοθεί στη διαδικασία απελευθερώνονται.



Σχ. 10.2. Πέρασμα παραμέτρων κατά την κλήση διαδικασιών (α) Κατάσταση πριν την κλήση (β) Μεταβίβαση τιμών των μεταβλητών Α και Β στις Χ και Υ αντίστοιχα (γ) Στη διαδικασία εκχωρούνται τιμές στις μεταβλητές Διαφορά και Άθροισμα (δ) Οι τιμές των τελευταίων επιστρέφονται στις Διαφ1 και Αθρ1 μετά το τέλος της διαδικασίας.

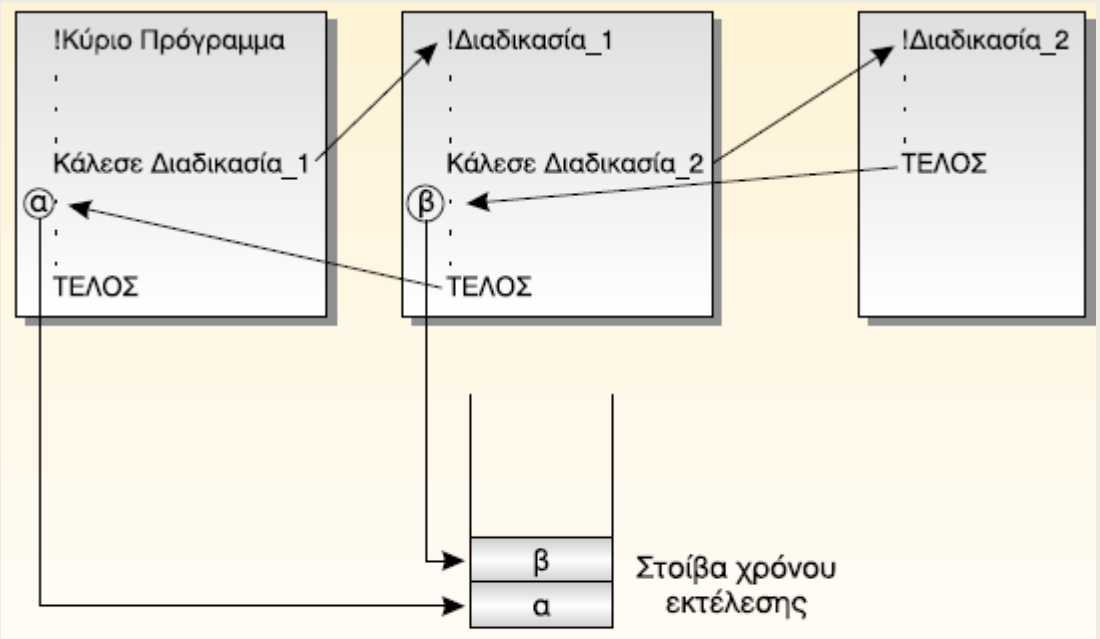
Οι λίστες **των παραμέτρων** πρέπει να ακολουθούν τους εξής **κανόνες**:

1. Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.
2. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση. Για παράδειγμα η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κοκ.
3. Η τυπική παράμετρος και η αντίστοιχη της πραγματική πρέπει να είναι του ιδίου τύπου.

Η χρήση στοίβας στην κλήση διαδικασιών

Η έννοια της στοίβας είναι πολύ χρήσιμη στο ίδιο το λογισμικό των γλωσσών προγραμματισμού. Όταν μία διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η αμέσως επόμενη διεύθυνση του κύριου προγράμματος, που ονομάζεται ***διεύθυνση επιστροφής*** (return address), αποθηκεύεται από το μεταφραστή σε μία στοίβα που ονομάζεται ***στοίβα χρόνου εκτέλεσης*** (execution time stack).
Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής απωθείται από τη στοίβα και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κύριο πρόγραμμα.
Η τεχνική αυτή εφαρμόζεται και γενικότερα, δηλαδή οποτεδήποτε μία διαδικασία ή συνάρτηση καλεί μία διαδικασία ή συνάρτηση.

Για παράδειγμα, έστω ότι μία διαδικασία a καλεί τη διαδικασία b, που με τη σειρά της καλεί τη διαδικασία c κοκ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στη στοίβα με σειρά c, b, a. Μετά την εκτέλεση κάθε διαδικασίας, η διεύθυνση επιστροφής απωθείται από τη στοίβα και ο έλεγχος μεταβιβάζεται στη διεύθυνση αυτή. Το παράδειγμα αυτό δείχνει μία από τις πολλές χρησιμότητες της LIFO ιδιότητας της στοίβας.



Σχ. 10.3. Χρήση στοίβας από το μεταφραστή για το χειρισμό κλήσεων διαδικασιών και επιστροφών από αυτές.

Άσκηση 5:

Τι είδους υποπρόγραμμα, διαδικασία ή συνάρτηση, πρέπει να χρησιμοποιήσεις για τα παρακάτω:

Α) Εισαγωγή τριών δεδομένων.
Β) Εισαγωγή ενός δεδομένου.
Γ) Υπολογισμός του μικρότερου από πέντε ακεραίους.
Δ) Υπολογισμός των δύο μικρότερων από πέντε ακεραίους.
Ε) Έλεγχος αν δύο αριθμοί είναι ίσοι.
Ζ) Να ταξινομεί, και να επιστρέφει ταξινομημένους, πέντε αριθμούς.
Η) Έλεγχος αν ένας χαρακτήρας είναι φωνήεν ή σύμφωνο.

Άσκηση 6:

Τι θα τυπώσουν οι παρακάτω εντολές;

```
……
Α <- 5
Β <- 10
Γ <- 0
ΚΑΛΕΣΕ Διαδ1(Α, Β)
ΓΡΑΨΕ Α,Β,Γ
……
ΔΙΑΔΙΚΑΣΙΑ Διαδ1(Γ,Δ)
……
ΑΡΧΗ
Γ <- Γ-Δ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Διαδ1
```

Α. 5,10,0 Β. 5,10, -5 Γ. -5,10,0 Δ. -5,10,-5

Άσκηση 7:

Τι θα τυπώσουν οι παρακάτω εντολές

```
……
Α <- 5
Β <- 10
ΚΑΛΕΣΕ Διαδ1(Β, Α)
ΓΡΑΨΕ Α,Β
……
ΔΙΑΔΙΚΑΣΙΑ Διαδ1(Α,Β)
……
ΑΡΧΗ
ΓΡΑΨΕ Α,Β
Α <- Α-Β
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Διαδ1
```

Α. 5,10	Β. 10,5	Γ. 5,10	Δ. 10, 5
5,10	5,5	-5,10	5,10

Άσκηση 8:
Να γράψετε κυρίως πρόγραμμα το οποίο θα διαβάζει την ακτίνα ενός κύκλου και θα τυπώνει το εμβαδόν και την περίμετρό του τα οποία θα υπολογίζονται από την διαδικασία με όνομα ***κύκλος*** την οποία και θα καλεί το κυρίως πρόγραμμα.

Άσκηση 9:
Να γράψετε διαδικασία με το όνομα ανταλλαγή, η οποία θα ανταλλάσσει τις τιμές 2 μεταβλητών. Την διαδικασία αυτή να την χρησιμοποιήσετε σε κυρίως πρόγραμμα το οποίο ταξινομεί έναν πίνακα.

Άσκηση 10 ΘΕΜΑ 2ο Δ, Ε 2003
Άσκηση 11 ΘΕΜΑ 2ο 2005
Άσκηση 12 ΘΕΜΑ 1ο Γ 2006
Άσκηση 13 ΘΕΜΑ 1ο Β2 2007
Άσκηση 14 ΘΕΜΑ 2ο 2007
Άσκηση 15 ΘΕΜΑ 3ο 2008
Άσκηση 16 ΘΕΜΑ 1ο Γ3 ΕΠΑΝΑΛ 2007
Άσκηση 17 ΘΕΜΑ 4ο ΕΠΑΝΑΛ 2007

Άσκηση 18:
Να γράψεις πρόγραμμα το οποίο θα διαβάζει δύο αριθμούς, θα καλεί υποπρόγραμμα το οποίο θα υπολογίζει το Μέγιστο Κοινό Διαιρέτη (ΜΚΔ) και άλλο υποπρόγραμμα το οποίο θα υπολογίζει το Ελάχιστο Κοινό Πολλαπλάσιο (ΕΚΠ) και τέλος το κυρίως πρόγραμμα θα τυπώνει τα αποτελέσματα.

Άσκηση 19:
Να γραφεί πρόγραμμα το οποίο να προσθέτει δύο κλάσματα. Το πρόγραμμα δέχεται τέσσερις ακεραίους αριθμούς τους παρανομαστές και τους αριθμητές των δύο κλασμάτων υπολογίζει και εκτυπώνει τον αριθμητή και τον παρανομαστή του αποτελέσματος.
Α/Β + Γ/Δ = Ε/Ζ
Υπόδειξη:
Ενώ το πρόβλημα αρχικά φαίνεται απλό, η υλοποίησή του είναι αρκετά πολύπλοκη. Αρχικά πρέπει να απλοποιηθούν τα κλάσματα, στη συνέχεια να γίνουν ομώνυμα, να προστεθούν οι αριθμητές και τέλος να απλοποιηθεί το αποτέλεσμα.
Οι διαδικασίες αυτές απαιτούν τον υπολογισμό του ΜΚΔ (για την απλοποίηση) και του ΕΚΠ για τη μετατροπή των κλασμάτων σε ομώνυμα. Να χρησιμοποιήσετε τις συναρτήσεις της άσκησης 19.

Υποπρογράμματα και πίνακες

Ένας ολόκληρος πίνακας μπορεί να περάσει σαν παράμετρος σε ένα υποπρόγραμμα όπως ακριβώς μια απλή μεταβλητή. Παρατηρήστε μόνο ότι το μέγεθος του πίνακα δηλώνετε και στο κυρίως πρόγραμμα και στο υποπρόγραμμα αλλά όχι στην κλήση ή στον ορισμό του:

```
! Το πρόγραμμα αυτό διαβάζει και εμφανίζει τα περιεχόμενα
! ενός μονοδιάστατου πίνακα ακεραίων 10 θέσεων,
! χρησιμοποιώντας για το σκοπό αυτό 2 διαδικασίες.
ΠΡΟΓΡΑΜΜΑ Τμηματικός_προγραμματισμός_1
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Π[10]
ΑΡΧΗ
  ΚΑΛΕΣΕ Διάβασε_πίνακα_10_θέσεων(Π)
  ΚΑΛΕΣΕ Εμφάνισε_πίνακα_10_θέσεων(Π)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Τμηματικός_προγραμματισμός_1

ΔΙΑΔΙΚΑΣΙΑ Διάβασε_πίνακα_10_θέσεων(πίνακας)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: πίνακας[10], χ
ΑΡΧΗ
  ΓΡΑΨΕ 'Τώρα εκτελείται η διαδικασία ΔΙΑΒΑΣΜΑΤΟΣ πίνακα'
  ΓΙΑ χ ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΡΑΨΕ 'Δώσε το', χ, 'ο στοιχείο του πίνακα:'
    ΔΙΑΒΑΣΕ πίνακας[χ]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΔΙΑΔΙΚΑΣΙΑ Εμφάνισε_πίνακα_10_θέσεων(πιν)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: πιν[10], κ
ΑΡΧΗ
  ΓΡΑΨΕ 'Τώρα εκτελείται η διαδικασία ΕΜΦΑΝΙΣΗΣ πίνακα'
  ΓΙΑ κ ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΡΑΨΕ πιν[κ]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Άσκηση 20:
Να συμπληρώσετε το παραπάνω παράδειγμα γράφοντας ένα υποπρόγραμμα με το όνομα ***ταξινόμηση*** το οποίο ταξινομεί τον πίνακα. Να το καλέσετε από το κυρίως πρόγραμμα με τέτοιο τρόπο ώστε η διαδικασία της εμφάνισης να εμφανίζει τον πίνακα ταξινομημένο. Επίσης να γράψετε και να καλέσετε υποπρόγραμμα με το όνομα ***άθροισμα***, το οποίο θα υπολογίζει το άθροισμα των στοιχείων του πίνακα. Τι είδους υποπρόγραμμα μπορεί να είναι η **ταξινόμηση** και τι το **άθροισμα**;