ΣΗΜΕΙΩΣΕΙΣ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

ΜΕΡΟΣ Α’

ΘΕΟΔΟΣΙΟΥ ΔΙΟΝΥΣΙΟΣ
2ο ΕΝΙΑΙΟ ΛΥΚΕΙΟ ΑΓΙΟΥ ΔΗΜΗΤΡΙΟΥ

ΣΤΑΘΕΡΕΣ – ΜΕΤΑΒΛΗΤΕΣ – ΤΕΛΕΣΤΕΣ - ΕΚΦΡΑΣΕΙΣ

<u>Σταθερές:</u> τα μεγέθη που δεν μεταβάλλεται η τιμή τους κατά την εκτέλεση

πχ.
2, 6, -8 (**ακέραιες**)
2.5 -4.00 (**πραγματικές**)
'……..' (**χαρακτήρες**)
Αληθής, ψευδής (**λογικές**)

<u>Μεταβλητές:</u> τα μεγέθη που μεταβάλλεται η τιμή τους κατά την εκτέλεση. Τα ονόματα τους λέγονται αναγνωριστικά και ξεκινάνε από γράμμα και μετά μπορούν να έχουν μόνο γράμματα, αριθμούς και την κάτω παύλα (_).
πχ αθρ, ΜΟ, Χ2, Α_2_, ….
Στις μεταβλητές πρέπει να δηλώνουμε υποχρεωτικά τον τύπο τους που είναι ανάλογος με την τιμή που θα βάλουμε στην μεταβλητή και είναι: **Ακέραιες, Πραγματικές, Λογικές ή Χαρακτήρες**.

ΕΚΦΡΑΣΗ = **ΤΕΛΕΣΤΕΟΙ** + **ΤΕΛΕΣΤΕΣ** (σύμβολα πράξεων)

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

Οι πράξεις γίνονται σύμφωνα με την παραπάνω ιεραρχία και για να αλλάξουμε την σειρά χρησιμοποιούμε παρενθέσεις, αφού προηγούνται οι πράξεις μέσα σε αυτές. Αν έχουμε πολλές ισοδύναμες πράξεις εκτελούνται με τη σειρά από τα αριστερά προς τα δεξιά. **Πάντα έχουμε αριθμητικό αποτέλεσμα**.

2. Συγκριτικοί
>, <, =, >=, <=, <>
Πάντα μας δίνουν λογικό αποτέλεσμα (αληθής ή ψευδής)

7 > 3 (αληθής)
'α' > 'β' (ψευδής)
'ΠΑΠΑ**Σ**' > 'ΠΑΠΑ**Δ**ΑΚΗΣ' (αληθής)

3. Λογικοί
1. ΟΧΙ
2. ΚΑΙ
3. Η

Οι λογικές πράξεις εκτελούνται με την παραπάνω σειρά και δίνουν πάντα λογικό αποτέλεσμα.
Η λογική πράξη **ΚΑΙ** δίνει Αληθής όταν είναι και οι δύο Αληθής, ενώ η λογική **Η** όταν είναι τουλάχιστον μία Αληθής. Η πράξη **ΟΧΙ** αντιστρέψει τη λογική τιμή που την ακολουθεί.

DIV μας δίνει το **ακέραιο πηλίκο** της διαίρεσης 2 ακέραιων αριθμών. Πχ 20 div 3 = 6, 2 div 5 = 0

MOD μας δίνει το **ακέραιο υπόλοιπο** της διαίρεσης 2 ακέραιων αριθμών. Πχ 20 mod 3 = 2, 2 mod 5 = 2

ΕΝΤΟΛΗ ΕΙΣΟΔΟΥ

ΔΙΑΒΑΣΕ λίστα μεταβλητών πχ ΔΙΑΒΑΣΕ Χ,Υ και όχι ~~ΔΙΑΒΑΣΕ 7~~

ΕΝΤΟΛΗ ΕΞΟΔΟΥ

ΓΡΑΨΕ λίστα στοιχείων πχ ΓΡΑΨΕ 'Χ=',Χ

ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ

ΜΕΤΑΒΛΗΤΗ ← ΕΚΦΡΑΣΗ

ΚΕΦΑΛΑΙΟ 1ο

ΑΝΑΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

Η έννοια του προβλήματος

Πρόβλημα είναι μία κατάσταση η οποία χρήζει αντιμετώπισης, απαιτείται λύση, η οποία δεν είναι γνωστή, ούτε προφανής.

Μερικά παραδείγματα προβλημάτων είναι η τρύπα του όζοντος, γιατί δημιουργούνται παλίρροιες, γιατί γίνονται σεισμοί κλπ.

Κατανόηση του προβλήματος

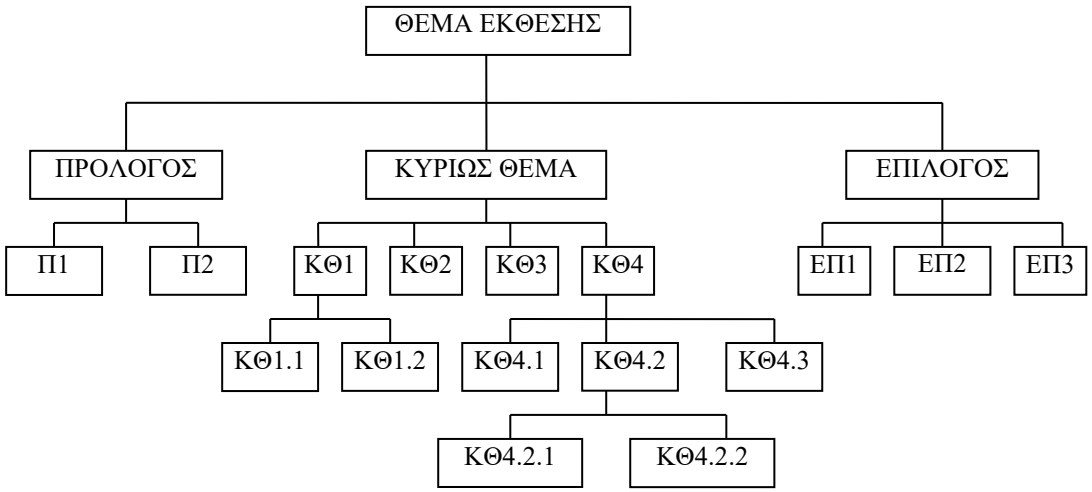
Για να μπορέσουμε να επιλύσουμε ένα πρόβλημα σωστά απαιτείται πρώτα η κατανόηση του. Για να είναι κατανοητό ένα πρόβλημα υπάρχουν δύο παράγοντες:

1. Η ορθή διατύπωση του δημιουργού του προβλήματος
2. Η σωστή ερμηνεία εκείνου που θα το αντιμετωπίσει

Η σαφήνεια διατύπωσης είναι απαραίτητη για την κατανόηση του προβλήματος και κατ' επέκταση και την επίλυση του. 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.

Δομή του προβλήματος

Ακολουθεί παράδειγμα δομής μίας έκθεσης



Η καταγραφή της δομής ενός προβλήματος σημαίνει αυτόματα ότι έχει αρχίσει η διαδικασία ανάλυσης του προβλήματος σε άλλα απλούστερα.

Δομή προβλήματος είναι:

1. τα συστατικά του μέρη
2. τα επιμέρους τμήματα που το αποτελούν
3. καθώς και ο τρόπος που τα μέρη συνδέονται μεταξύ τους

Ο τρόπος να απλοποιήσουμε την επίλυση του προβλήματος είναι η συνεχής ανάλυση του προβλήματος σε μικρότερα προβλήματα. Η ανάλυση σε μικρότερα προβλήματα σταματάει όταν η λύση των υποπροβλημάτων στο τελευταίο επίπεδο είναι εύκολη και προφανής. Η ανάλυση αυτή μας βοηθάει στο να ανακαλύψουμε τη δομή του προβλήματος. Ένας τρόπος παρουσίασης και ανάλυσης της δομής του προβλήματος είναι η διαγραμματική αναπαράσταση.

Καθορισμός απαιτήσεων

Για την επίλυση του προβλήματος απαιτείται η εύρεση των δεδομένων και των ζητούμενων.

Δεδομένα είναι τα στοιχεία που δίνουμε και γίνονται αντιληπτά με μία τουλάχιστον από τις αισθήσεις μας.

Πληροφορία είναι οποιοδήποτε γνωσιακό στοιχείο προκύπτει από την επεξεργασία των δεδομένων που έχουμε εισάγει σε έναν αλγόριθμο.

Επεξεργασία δεδομένων είναι η διαδικασία κατά την οποία ένας μηχανισμός δέχεται δεδομένα τα επεξεργάζεται με ένα προκαθορισμένο τρόπο (εντολές προγράμματος) και μας δίνει τις πληροφορίες.

Τα **στάδια για την επίλυση** ενός προβλήματος είναι:

- κατανόηση, ώστε να βρούμε τα δεδομένα και τα ζητούμενα του προβλήματος
- ανάλυση, ώστε να διαιρεθεί το αρχικό πρόβλημα σε μικρότερα προβλήματα
- επίλυση, επιλύνοντας κάθε υποπρόβλημα επιλύνουμε στο τέλος και το αρχικό πρόβλημα.

ΚΕΦΑΛΑΙΟ 2ο

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

Τι είναι Αλγόριθμος

Οδηγίες για να βρούμε μία διεύθυνση, οδηγίες χρήσης συσκευών, συνταγές μαγειρικής, εύρεση τηλεφώνου.

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

Αλγόριθμος (πληροφορική) *είναι η διαδοχή λογικών βημάτων που απαιτούνται για την επίλυση του προβλήματος με ηλεκτρονικό υπολογιστή σε γλώσσα κατανοητή από αυτόν ή σε γλώσσα που χρησιμοποιεί όμοια δομή (συμβολική γλώσσα)*

Κριτήρια Αλγορίθμου

Είσοδος (ΔΙΑΒΑΣΕ): ένα σύνολο μεταβλητών (μπορεί και καμία πχ εμφάνιση ενός τυχαίου αριθμού, εμφάνιση στην οθόνη της προπαίδειας) που αποτελούν και τα δεδομένα του προβλήματος.

Έξοδος (ΓΡΑΨΕ): τα αποτελέσματα του αλγόριθμου ή τα ζητούμενα του προβλήματος.

Καθοριστικότητα: οι εντολές να είναι αυστηρά καθορισμένες ως προς την εκτέλεση (δεν γίνεται διαίρεση με το 0 και δεν υπάρχει τετραγωνική ρίζα αρνητικού αριθμού).

Περατότητα: λύση του προβλήματος με πεπερασμένο αριθμό εντολών σε πεπερασμένο χρόνο. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία.

Αποτελεσματικότητα: κάθε εντολή να είναι διατυπωμένη απλά και κατανοητά, ώστε να εκτελεστεί επακριβώς και σε πεπερασμένο χρόνο.

Σπουδαιότητα αλγορίθμων

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

Γλωσσών Προγραμματισμού (programming languages). Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλαδή, χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου.

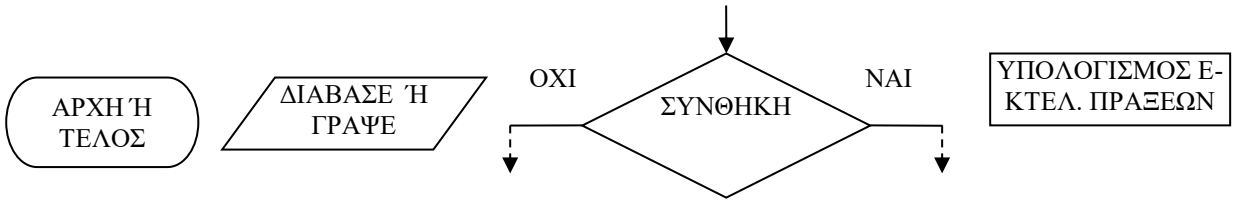
Θεωρητική (theoretical). Το ερώτημα που συχνά τίθεται είναι αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος.

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

Περιγραφή και αναπαράσταση αλγορίθμων

Ελεύθερο κείμενο: απλή ελληνική γλώσσα όπως θα μιλάγαμε σε ένα φίλο τους, χωρίς καμία δομή στην παρουσίαση και πιθανό **πρόβλημα μη εκτέλεσης, άρα της αποτελεσματικότητας**.

Διαγραμματικές τεχνικές: χρησιμοποιούμε ειδικά σύμβολα και σχήματα για να αναπαραστήσουμε γραφικά τα βήματα του αλγόριθμου, όπου το κάθε ένα έχει τη σημασία του. Η πιο παλιά και η πιο γνωστή είναι το διάγραμμα ροής. Ακολουθούν τα σχήματα για τα διαγράμματα ροής.



Φυσική γλώσσα με βήματα: χρήση απλής ελληνικής γλώσσας στην οποία οι προτάσεις έχουν διαχωριστεί σε παραγράφους και έχουν αριθμηθεί, αλλά **χρειάζεται μεγάλη προσοχή στην καθοριστικότητα**.

Κωδικοποίηση: με ένα πρόγραμμα γραμμένο σε μία γλώσσα προγραμματισμού που μπορεί να εκτελεστεί από τον υπολογιστή ή σε κώδικα που χρησιμοποιεί μία αλγοριθμική γλώσσα που είναι κατανοητή σε μας και όταν εκτελεστεί θα δώσει τα ίδια αποτελέσματα. Ακολουθεί η γενική μορφή του αλγορίθμου σε ΓΛΩΣΣΑ

ΠΡΟΓΡΑΜΜΑ όνομα_προγράμματος
ΜΕΤΑΒΛΗΤΕΣ
……...
ΑΡΧΗ
ΕΝΤΟΛΕΣ (προτάσεις της ελληνικής γλώσσας που μπορεί να περιέχουν τύπους και συμβολισμούς)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Οι λέξεις που χρησιμοποιούνται στον κώδικα έχουν αυστηρά καθορισμένη έννοια και τρόπο χρήσης και λέγονται δεσμευμένες λέξεις.

Σύγκριση τρόπων παράστασης αλγορίθμου

Καλύτερος τρόπος είναι ο κωδικοποίηση, μετά το διάγραμμα ροής, η φυσική γλώσσα με βήματα και τέλος η φυσική γλώσσα. Η κωδικοποίηση είναι ο καλύτερος από τους τρόπους παρουσίασης, γιατί πλησιάζει πιο πολύ από τους άλλους την γλώσσα προγραμματισμού (είναι πιο εύκολο να μετατραπεί) που είναι και ο επιθυμητός τρόπος επίλυσης μέσω υπολογιστών.

Σταθερές – μεταβλητές (πεδίο ορισμού)

Σταθερά είναι εκείνα τα μεγέθη που δεν αλλάζει η τιμή τους κατά την εκτέλεση του αλγορίθμου. Επίσης το *3, 7.8* είναι αριθμητικές σταθερές, το *'Γιώργος'* σταθερά χαρακτήρων, ενώ το *Αληθής* ή *Ψευδής* είναι λογικές σταθερές. Μπορούμε και εμείς να δημιουργήσουμε σταθερές πχ π = 3.14, name='John', που δηλώνονται πριν τις μεταβλητές.

Μεταβλητή είναι το αντίθετο, δηλ τα μεγέθη εκείνα που η τιμή τους αλλάζει κατά την εκτέλεση του αλγόριθμου και αντιστοιχούν σε κάποιο όνομα πχ x, y (*Πχ στο y=2x+3 μεταβάλλεται το y και αλλάζει με αυτήν την εντολή ανάλογα με την τιμή της μεταβλητής x, ενώ τα 2, 3 είναι σταθερές).*

Το σύνολο των χαρακτήρων (γράμματα, αριθμοί και η κάτω γραμμή «_») που χρησιμοποιούμε για τα ονόματα προγραμμάτων, υποπρογραμμάτων, μεταβλητών, σταθερών καλούνται **αναγνωριστικά** και δεν πρέπει να είναι τυχαία, αλλά να εξηγούν το ρόλο και τη σημασία των μεγεθών που εκφράζουν. *Πχ Sum(άθροισμα), Max (Μέγιστος)***.** Τα αναγνωριστικά:

1. ξεκινούν πάντα από γράμμα και
2. δεν μπορούν να περιέχουν μέσα κενά και ειδικούς χαρακτήρες (, . ; : ! @ # $ κλπ) εκτός από την κάτω γραμμή «_».

Οι λέξεις (ΠΡΟΓΡΑΜΜΑ, ΜΕΤΑΒΛΗΤΕΣ, ΑΚΕΡΑΙΕΣ, ΔΙΑΒΑΣΕ, ΓΡΑΨΕ, ΑΝ, ΤΟΤΕ κλπ) που χρησιμοποιούνται από την γλώσσα προγραμματισμού καλούνται **δεσμευμένες λέξεις** και δεν μπορούν να χρησιμοποιηθούν σαν **αναγνωριστικά**.

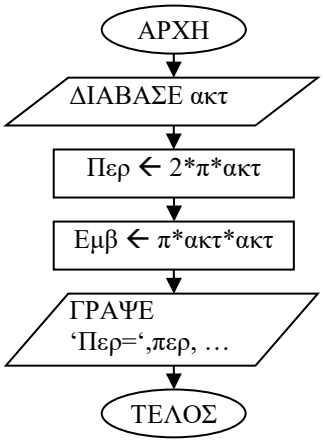
Οι **τύποι** που μπορούμε να χρησιμοποιήσουμε σαν σταθερές ή μεταβλητές είναι:

1. **ΑΚΕΡΑΙΕΣ** ακέραιες αριθμητικές τιμές πχ 7
2. **ΠΡΑΓΜΑΤΙΚΕΣ** πραγματικές αριθμητικές τιμές πχ 12.3
3. **ΧΑΡΑΚΤΗΡΕΣ** τιμές χαρακτήρων μέσα σε ' ' πχ 'Η Εξίσωση δεν έχει λύσεις'
4. **ΛΟΓΙΚΕΣ** λογικές τιμές **αληθής** (**TRUE**) ή **ψευδής** (**FALSE**), όπου δεν μπορούμε να τις διαβάσουμε ή να τις εμφανίσουμε στην οθόνη απ' ευθείας

Χ ← 12	το Χ είναι ακέραια μεταβλητή
Υ ← 13.5	το Υ είναι πραγματική μεταβλητή
SUM ← Χ + Υ	το SUM είναι πραγματική μεταβλητή (ακέραιος και πραγματικός)
ΜΟ ← Χ / 3	ο ΜΟ είναι πραγματική μεταβλητή (αποτέλεσμα διαίρεσης)
ΟΝΟΜΑ ← 'ΓΙΩΡΓΟΣ'	το ΟΝΟΜΑ είναι μεταβλητή χαρακτήρων (το 'ΓΙΩΡΓΟΣ' είναι τιμή χαρακτήρων)
ΠΑΝΤΡΕΜΕΝΟΣ ← αληθής	το ΠΑΝΤΡΕΜΕΝΟΣ είναι λογική μεταβλητή, αφού το αληθής είναι η μία από τις δύο λογικές τιμές που παίρνουν οι λογικές μεταβλητές

Σε ένα πρόγραμμα πρέπει να δηλώνουμε τις μεταβλητές που θα χρησιμοποιήσουμε. Οπότε όταν γράφουμε ένα πρόγραμμα πρέπει να χρησιμοποιούμε την παρακάτω δομή:

ΠΡΟΓΡΑΜΜΑ ΚΥΚΛΟΣ
ΣΤΑΘΕΡΕΣ
π = 3.14
ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: ακτ, περ, εμβ
ΑΡΧΗ
ΔΙΑΒΑΣΕ ακτ
Περ ← 2*π*ακτ
Εμβ ← π*ακτ*ακτ ! ή π*ακτ^2
ΓΡΑΨΕ 'Περ= ', περ, ' Εμβ = ', εμβ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ



ΕΚΦΡΑΣΕΙΣ

Οι εκφράσεις διαμορφώνονται από τους τελεστέους (σταθερές ή μεταβλητές) και από τους τελεστές. Η διεργασία υπολογισμού μίας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και την εκτέλεση των πράξεων. Η τελική τιμή μίας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση των παρενθέσεων. Μία έκφραση μπορεί να αποτελείται από μία σταθερά ή μεταβλητή ή και από μία πολύπλοκη παράσταση.

ΤΕΛΕΣΤΕΣ

Τελεστές είναι τα σύμβολα που χρησιμοποιούμε στις διάφορες πράξεις. Οι τελεστές χωρίζονται σε τρεις κατηγορίες: ***αριθμητικοί, συγκριτικοί και λογικοί τελεστές.***

Στην ιεραρχία των τελεστών προηγούνται οι αριθμητικοί, μετά οι συγκριτικοί και τελευταίοι οι λογικοί.

Συγκριτικοί τελεστές (<, >, =, < >, >=, <=)

Η σύγκριση μεταξύ αριθμών γίνεται με προφανή τρόπο. Η σύγκριση μεμονωμένων χαρακτήρων στηρίζεται στην αλφαβητική σειρά δηλ το 'α' είναι μικρότερο από το 'β'.

Η σύγκριση μεταξύ αλφαριθμητικών δεδομένων στηρίζεται στη σύγκριση χαρακτήρα προς χαρακτήρα μέχρι να βρεθεί κάποιος διαφορετικός, και το αποτέλεσμα των πρώτων διαφορετικών χαρακτήρων είναι και το αποτέλεσμα της σύγκρισης των δύο δεδομένων πχ το 'είναι' είναι μεγαλύτερο από το 'είμαι', αφού το 'ν' είναι μεγαλύτερο του 'μ'.

Αριθμητικοί τελεστές (χρησιμοποιούνται στις πράξεις)

Οι πράξεις είναι ^, +, -, *, / (πραγματική διαίρεση), DIV (ακέραιο πηλίκο), MOD (ακέραιο υπόλοιπο). Προηγούνται οι πολλαπλασιασμοί και οι διαιρέσεις και μετά οι προσθέσεις και οι αφαιρέσεις. Στην περίπτωση που υπάρχουν πολλές ισοδύναμες πράξεις ξεκινάμε από αριστερά. Αν θέλουμε να αλλάξουμε τη σειρά των πράξεων βάζουμε τις πράξεις που θέλουμε να προηγηθούν μέσα σε παρένθεση.

```
        20 | 7          X ← 20 DIV 7 ! (δηλ το 2)
       -14 |
MOD ──► 6  | 2 ◄── DIV   Y ← 20 MOD 7 ! (δηλ το 6)
```

A + B / 2 = $A + \frac{B}{2}$ (προηγείται η διαίρεση)

(A + B) / 2 = $\frac{A+B}{2}$ (προηγείται η πρόσθεση της παρένθεσης)

ΠΡΟΤΕΡΑΙΟΤΗΤΑ
1. ^
2. *, /, DIV, MOD
3. +,-

Λογικοί τελεστές (ΟΧΙ, ΚΑΙ, Ή)

Τις συνθήκες τις χρησιμοποιούμε στις εντολές επιλογής και επανάληψης. Μέχρι τώρα έχουμε συναντήσει στις συνθήκες τους **τελεστές σύγκρισης (> , < , <> , = , >= , <=)** σε απλά παραδείγματα. Στις συγκρίσεις παίρνουμε σαν αποτέλεσμα (αληθής ή ψευδής) ανάλογα αν η συνθήκη ισχύει ή όχι. Υπάρχουν όμως περιπτώσεις που οι συνθήκες είναι πιο πολύπλοκες και για αυτό το λόγο χρησιμοποιούμε τις λογικές πράξεις (άρνηση 'ΟΧΙ', σύζευξη 'ΚΑΙ' και διάζευξη 'Ή'). ***Η σειρά εκτέλεσης των λογικών πράξεων είναι ΟΧΙ, ΚΑΙ, Ή.***

Πίνακας αληθείας λογικών τελεστών ΟΧΙ, ΚΑΙ, Ή

Σ1	Σ2	ΟΧΙ (Σ1) ΑΡΝΗΣΗ	Σ1 ΚΑΙ Σ2 ΣΥΖΕΥΞΗ	Σ1 Ή Σ2 ΔΙΑΖΕΥΞΗ
Α	**Α**	**Ψ**	**Α**	**Α**
Α	**Ψ**	**Ψ**	**Ψ**	**Α**
Ψ	**Α**	**Α**	**Ψ**	**Α**
Ψ	**Ψ**	**Α**	**Ψ**	**Ψ**

Επίσης *χρειάζεται προσοχή στην άρνηση όταν περιέχει λογική πράξη*, όπου πρέπει να αλλάξουμε και τις λογικές πράξεις από ΚΑΙ σε Ή και αντίστροφα. Ακολουθούν δύο παραδείγματα.

ΟΧΙ (Σ1 **ΚΑΙ** Σ2) = ΟΧΙ (Σ1) **Ή** ΟΧΙ (Σ2)
ΟΧΙ (Σ1 **Ή** Σ2) = ΟΧΙ (Σ1) **ΚΑΙ** ΟΧΙ (Σ2)
όπου Α = Αληθής και Ψ = Ψευδής και κάτω από τους τελεστές το αποτέλεσμα και Σ1, Σ2 δύο λογικού τύπου τιμές

Σχόλια Σε κάθε αλγόριθμο καλό θα είναι να υπάρχουν και κάποια σχόλια, ώστε να γίνει πιο κατανοητός και να μας θυμίζει τι κάνει ο αλγόριθμος ή κάποιο κομμάτι του. Ότι ακολουθείται από το «!» αγνοείται από τον μεταγλωττιστή ή τον διερμηνευτή και δεν εκτελείται όπως το υπόλοιπο πρόγραμμα.

Γράφουμε το **σύμβολο «!»** και μετά τα σχόλια.

```
ΔΙΑΒΑΣΕ Β1, Β2        ! Β1 βάση μεγάλη, Β2 βάση μικρή
```

Εντολή απόδοσης τιμής (εκχώρησης)

Το ίσον αντικαθίσταται από το ρήμα τοποθέτησε ή βάλε που αποδίδει μια τιμή σε μία μεταβλητή.

Θα τη συμβολίζουμε με το ‘ ← ‘

Το σύμβολο αυτό δεν είναι το σύμβολο της ισότητας.

Μεταβλητή ← Έκφραση *Παράδειγμα η εντολή a ← a+1*

Πρώτα εκτελούνται οι πράξεις αν υπάρχουν στο δεξί μέρος του συμβόλου εκχώρησης και στη συνέχεια το αποτέλεσμα **εκχωρείται στη μεταβλητή** που βρίσκεται **στο αριστερό μέρος**. Αν το α είχε την τιμή 5 και μετά την εκτέλεση της εντολής θα έχει τιμή 6.

Προσοχή στα ακόλουθα*:*

1. Στο αριστερό μέρος μπορεί να υπάρχει μόνο μεταβλητή και όχι κάποια παράσταση πράξη. Δηλ είναι λάθος η εντολή a +1 ← b+2
2. Οι μεταβλητές στο δεξί μέρος να έχουν τιμή και να μην είναι απροσδιόριστες, γιατί και το αποτέλεσμα θα είναι απροσδιόριστο. Απροσδιόριστη είναι μία μεταβλητή, όταν την χρησιμοποιούμε χωρίς να την έχουμε διαβάσει ή να της έχουμε εκχωρήσει μία τιμή πρώτα.
3. Πρέπει **το αποτέλεσμα της έκφρασης στο δεξί μέρος** να είναι **ίδιου τύπου με τον τύπο της μεταβλητής**. Δεν μπορούμε να αποθηκεύσουμε έναν πραγματικό σε μία ακέραιη μεταβλητή, αλλά **μπορούμε σε μία πραγματική μεταβλητή να αποθηκεύσουμε έναν ακέραιο**.

Εντολή Εισόδου

Εισαγωγή δεδομένων από το πληκτρολόγιο και αποθήκευση τους σε μεταβλητές για επεξεργασία

ΔΙΑΒΑΣΕ λίστα-μεταβλητών

ΔΙΑΒΑΣΕ Α
ΔΙΑΒΑΣΕ Β — Ή ισοδύναμα **ΔΙΑΒΑΣΕ Α, Β**

ΔΙΑΒΑΣΕ ΑΒ διαβάζει μόνο μία μεταβλητή την ΑΒ

~~ΔΙΑΒΑΣΕ Α Β~~ είναι λάθος εντολή το Α Β δεν μπορεί να είναι μεταβλητή με το κενό, ούτε δύο μεταβλητές αφού δεν έχουν μεταξύ τους κόμμα.

Εντολή Εξόδου

Εμφάνιση αποτελεσμάτων στην οθόνη του υπολογιστή

ΓΡΑΨΕ λίστα-στοιχείων (απευθείας τιμές σταθερών, τιμές μεταβλητών, αποτελέσματα πράξεων)

ΕΝΤΟΛΕΣ	ΟΘΟΝΗ
Χ ← 10	
ΟΝ ← ‘ΓΙΑΝΝΗΣ’	
ΓΡΑΨΕ Χ, 7, ‘ον’	10 7 ον
ΓΡΑΨΕ ‘Χ’, ‘20’	Χ 20
ΓΡΑΨΕ ‘Ο ‘, ΟΝ, ‘ ΠΕΡΑΣΕ ΜΕ ‘, Χ	Ο ΓΙΑΝΝΗΣ ΠΕΡΑΣΕ ΜΕ 10
ΓΡΑΨΕ ‘Χ = ‘, Χ*5	Χ = 50

Προσοχή στο παρακάτω συνηθισμένο λάθος:

ΛΑΘΟΣ	ΣΩΣΤΗ	ΣΩΣΤΗ
~~ΓΡΑΨΕ ΑΘΡ ← Χ + Υ~~ Δύο εντολές (εξόδου - εκχώρησης σε μία)	ΑΘΡ ← Χ + Υ ΓΡΑΨΕ ΑΘΡ	ΓΡΑΨΕ Χ + Υ

Παρακολούθηση τιμών μεταβλητών

Για την παρακολούθηση των τιμών των μεταβλητών χρησιμοποιούμε ένα πίνακα τιμών, όπου δίπλα σε κάθε βήμα του αλγόριθμου γράφουμε και τις τιμές των μεταβλητών που αλλάζουν.

Εύρεση λαθών με τους ακόλουθους τρόπους.

1. Ο τρόπος που γίνεται είναι χρησιμοποιώντας ένα πίνακα με πρώτη στήλη τα βήματα (εντολές) και τις επόμενες στήλες με τις μεταβλητές του αλγορίθμου.
2. Δίπλα σε κάθε βήμα του αλγόριθμου ορίζουμε στήλες με τις τιμές των μεταβλητών και έτσι βλέπουμε πως μεταβάλλονται οι τιμές τους πιο εύκολα από ότι ο πίνακας του 1ου τρόπου.

Παράδειγμα Έστω εισάγουμε τις τιμές 5 και 10

	κ	λ	Σ	Βοηθ
Πρόγραμμα παρακολούθηση				
Μεταβλητές				
Ακέραιες : κ, λ, Σ, Βοηθ				
Αρχή	?	?	?	?
Διάβασε κ	5			
Διάβασε λ		10		
Σ ← κ + λ			15	
Γράψε Σ				
Βοηθ ← κ				5
κ ← λ	10			
λ ← Βοηθ		5		
Γράψε κ, λ				
Τέλος_προγράμματος				

ΟΘΟΝΗ

Η μεταβλητή Βοηθ χρειάζεται για την αντιμετάθεση των τιμών των μεταβλητών κ και λ.

ΒΑΣΙΚΕΣ ΑΛΓΟΡΙΘΜΙΚΕΣ ΔΟΜΕΣ

Οι βασικές δομές που χρησιμοποιούμε για την επίλυση οποιουδήποτε αλγόριθμου είναι τρεις: ακολουθία, επιλογή και επανάληψη.

Ακολουθία

Κατά την αντιμετώπιση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών, οι εντολές εκτελούνται η μία μετά την άλλη. Πχ συνταγή μαγειρικής, διαδρομή για το σχολείο, σειρά διαβάσματος ενός κλασσικού βιβλίου, απαντήσεις τεστ στο σχολείο, πρόσθεση δύο αριθμών.

Οι εντολές που χρησιμοποιούνται είναι η εκχώρησης (←), ΔΙΑΒΑΣΕ και ΓΡΑΨΕ. Πχ

x ← 10 / 2	**!** εκχωρεί στο x το αποτέλεσμα της πράξης που είναι το 5
ΔΙΑΒΑΣΕ α, β	! ζητάει από το πληκτρολόγιο δύο τιμές
ΓΡΑΨΕ 'x =', x	! Εμφανίζει στην οθόνη «x=5»

Επιλογή

Πχ όταν σε μία διαδρομή φτάνουμε σε ένα σταυροδρόμι και πρέπει να επιλέξουμε το δρόμο που θα ακολουθήσουμε, η ανατροπή της σειράς διαβάσματος του βιβλίου όταν διαβάζουμε μία παραπομπή, η αλλαγή της ακολουθιακής σειράς απαντήσεων αν απαντάμε πρώτα τις εύκολες ερωτήσεις.

Συνήθως ανάλογα με το αποτέλεσμα μίας συνθήκης (λογικής έκφρασης) που περιέχει κάποιον ή κάποιους από τους **τελεστές σύγκρισης (> , < , <> , = , >= , <=) ή τους λογικούς τελεστές (ΚΑΙ, Ή, ΟΧΙ)**, ακολουθεί διαφορετική διαδρομή ο αλγόριθμος.

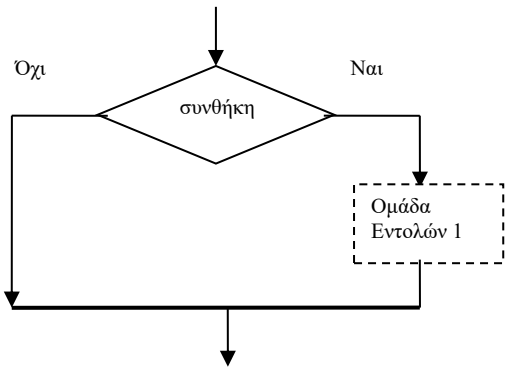
Απλή επιλογή (όταν έχουμε μόνο μία περίπτωση για το συγκεκριμένο έλεγχο)

Χρησιμοποιείται στην περίπτωση που όταν η συνθήκη είναι **αληθής θέλουμε να εκτελεστεί κάποια ομάδα εντολών,** ενώ όταν είναι **ψευδής δεν θέλουμε να εκτελεστεί αυτή η ομάδα εντολών**.

Κώδικας

ΑΝ <*συνθήκη*> **ΤΟΤΕ**
ομάδα εντολών 1
ΤΕΛΟΣ_ΑΝ

Διάγραμμα ροής



Παράδειγμα ο υπολογισμός της απόλυτης τιμής ενός αριθμού χρησιμοποιώντας μόνο μία μεταβλητή.

Σύνθετη επιλογή (όταν έχουμε μόνο δύο περιπτώσεις για το συγκεκριμένο έλεγχο)

Σε κάποια σημεία στο πρόγραμμα δεν είναι εμφανής η επόμενη πράξη, αλλά πρέπει να αποφασιστεί ανάλογα με το αποτέλεσμα μίας συνθήκης (ενός ελέγχου) ποιες εντολές θα εκτελεστούν και ποιες όχι.

Κώδικας

ΑΝ <*συνθήκη*> **ΤΟΤΕ**

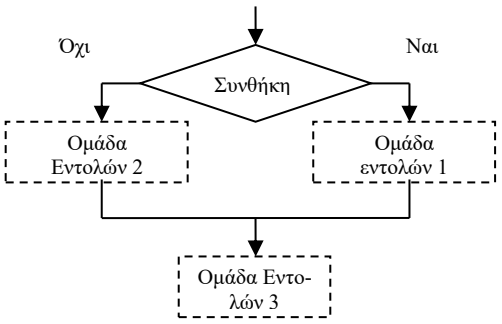
Ομάδα εντολών 1 **! αληθής συνθήκη**

ΑΛΛΙΩΣ

Ομάδα εντολών 2 **! ψευδής συνθήκη**

ΤΕΛΟΣ_ΑΝ

Διάγραμμα ροής



Όταν το πρόγραμμα φτάσει στην συνθήκη, αν είναι αληθής (ναι) θα εκτελεστούν οι εντολές της ομάδας 1 και στη συνέχεια θα εκτελεστούν οι υπόλοιπες εντολές (ομάδα εντολών 3) του προγράμματος από πάνω προς τα κάτω (διαδοχή). Αν η συνθήκη είναι ψευδής (όχι) τότε θα εκτελεστούν οι εντολές της ομάδας 2 και στη συνέχεια θα εκτελεστούν οι υπόλοιπες εντολές της ομάδας 3.

Πάντοτε εκτελείται η μία μόνο από τις δύο ομάδες εντολών 1 και 2 και στη συνέχεια οι εντολές του προγράμματος μετά τον έλεγχο της ομάδας 3.

Παράδειγμα1 να φτιαχτεί πρόγραμμα σε μορφή κώδικα και διάγραμμα ροής που θα διαβάζει τον βαθμό ενός μαθητή και θα μας εμφανίζει μήνυμα στην οθόνη ότι «Περνάει την τάξη» ή «Επαναλαμβάνει την τάξη».

Εμφωλευμένη - Πολλαπλή επιλογή -ΕΠΙΛΕΞΕ *(για παραπάνω από δύο περιπτώσεις)*

Υπάρχει περίπτωση σε ένα σημείο του προγράμματος να υπάρχουν πολλές ομάδες εντολών και να πρέπει με τον έλεγχο μία συνθήκης να εκτελεστεί μόνο μία από αυτές.

ΕΜΦΩΛΕΥΜΕΝΗ ΑΝ

```
ΑΝ <συνθήκη1> ΤΟΤΕ
    Εντολές1
ΑΛΛΙΩΣ
    ΑΝ <συνθήκη2> ΤΟΤΕ
        Εντολές2
    ΑΛΛΙΩΣ
        ΑΝ <συνθήκη3> ΤΟΤΕ
            Εντολές3
        …
        ΑΛΛΙΩΣ
            Εντολέςν
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
```

ΠΟΛΛΑΠΛΗ ΑΝ

```
ΑΝ <συνθήκη1> ΤΟΤΕ
    Εντολές1
ΑΛΛΙΩΣ_ΑΝ <συνθήκη2> ΤΟΤΕ
    Εντολές2
ΑΛΛΙΩΣ_ΑΝ <συνθήκη3> ΤΟΤΕ
    Εντολές3
…
ΑΛΛΙΩΣ
    Εντολέςν
ΤΕΛΟΣ_ΑΝ
```

ΕΠΙΛΕΞΕ

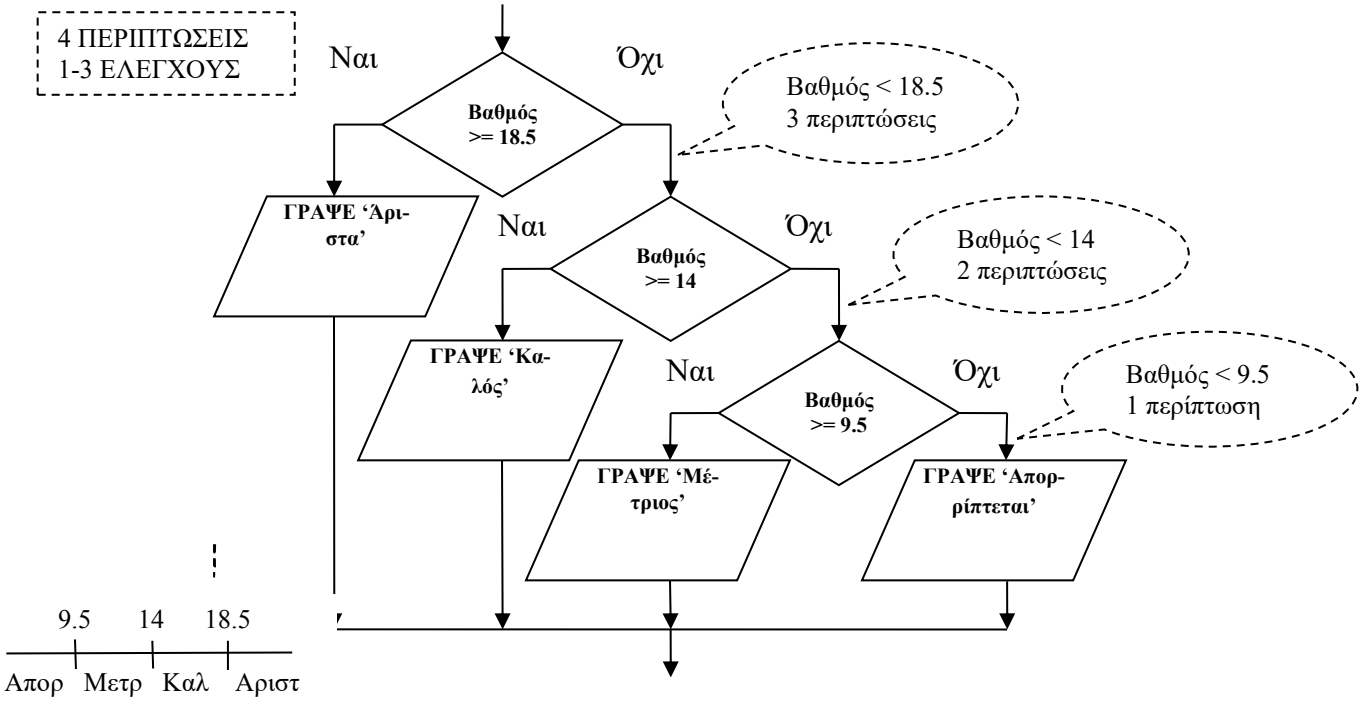
```
ΕΠΙΛΕΞΕ <μεταβλητή ή έκφραση>
    ΠΕΡΙΠΤΩΣΗ <τιμή1 ή όρια1>
        ομάδα εντολών 1
    ΠΕΡΙΠΤΩΣΗ <τιμή2 ή όρια2>
        ομάδα εντολών 2
    …
    ΠΕΡΙΠΤΩΣΗ <τιμήη ή όριαη>
        ομάδα εντολών n
    ΠΕΡΙΠΤΩΣΗ αλλιώς
        ομάδα εντολών
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

Παράδειγμα να φτιαχτεί Πρόγραμμα που ανάλογα με τον αριθμό που δίνουμε από 1-7 εμφανίζει την αντίστοιχη ημέρα ξεκινώντας από την Δευτέρα.

Παράδειγμα με Εμφωλευμένη Επιλογή (Φωλιασμένη Επιλογή)

Είναι δυνατόν μέσα σε μία ομάδα εντολών μίας συνθήκης επιλογής να μην περιέχονται μόνο ακολουθιακές εντολές, αλλά να υπάρχει και άλλη συνθήκη επιλογής. Πολλές φορές μπορούμε να πετύχουμε το ίδιο αποτέλεσμα χωρίς φωλιασμένη επιλογή, αλλά θα είναι πολύ *πιο αργός ο αλγόριθμος*, αφού έχει παραπάνω ελέγχους.

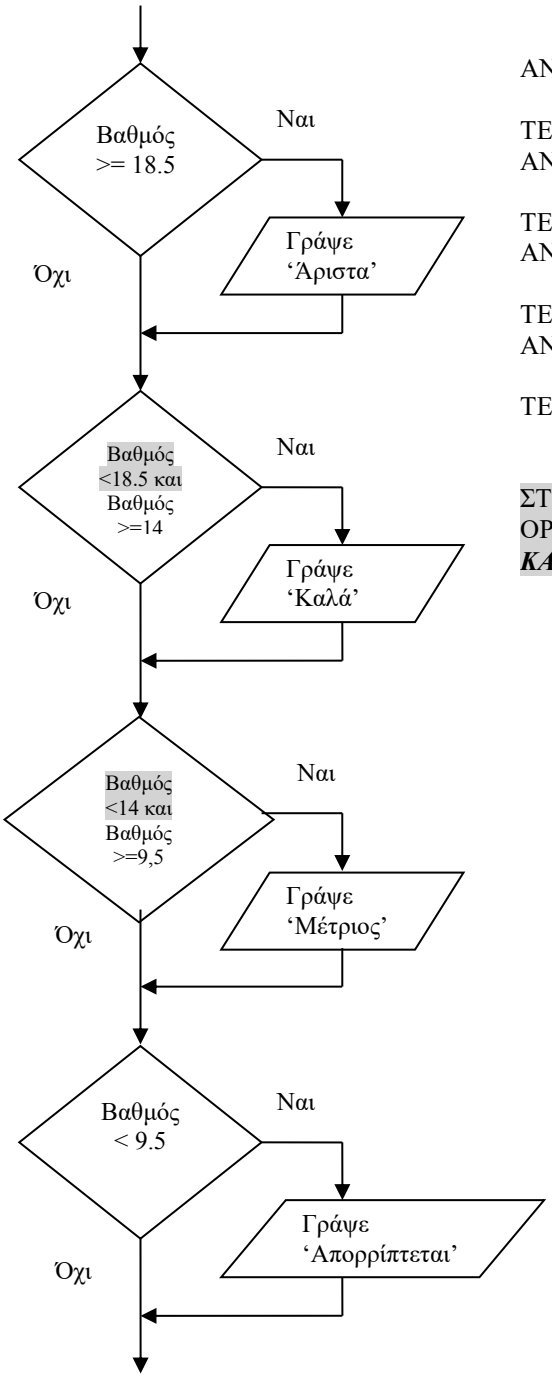
Παράδειγμα να γραφτεί πρόγραμμα που θα διαβάζει έναν βαθμό και θα εμφανίζει στην οθόνη τον χαρακτηρισμό βαθμολογίας («Άριστα», «Καλά», «Μέτρια», «Απορρίπτεται») και με τους δύο τρόπους (απλές επιλογές και φωλιασμένες επιλογές).



ΕΜΦΩΛΕΥΜΕΝΗ ΑΝ	ΠΟΛΛΑΠΛΗ ΑΝ	ΕΠΙΛΕΞΕ
ΑΝ Βαθμός >= 18.5 ΤΟΤΕ ΓΡΑΨΕ 'Άριστα' ΑΛΛΙΩΣ ΑΝ Βαθμός >= 14 ΤΟΤΕ ΓΡΑΨΕ 'Καλός' ΑΛΛΙΩΣ ΑΝ Βαθμός >= 9.5 ΤΟΤΕ ΓΡΑΨΕ 'Μέτριος' ΑΛΛΙΩΣ ΓΡΑΨΕ 'Απορρίπτεται' ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΑΝ ΤΕΛΟΣ_ΑΝ	ΑΝ Βαθμός >= 18.5 ΤΟΤΕ ΓΡΑΨΕ 'Άριστα' ΑΛΛΙΩΣ_ΑΝ Βαθμός >= 14 ΤΟΤΕ ΓΡΑΨΕ 'Καλός' ΑΛΛΙΩΣ_ΑΝ Βαθμός >= 9.5 ΤΟΤΕ ΓΡΑΨΕ 'Μέτριος' ΑΛΛΙΩΣ ΓΡΑΨΕ 'Απορρίπτεται' ΤΕΛΟΣ_ΑΝ	ΕΠΙΛΕΞΕ Βαθμός ΠΕΡΙΠΤΩΣΗ >=18.5 ΓΡΑΨΕ 'Άριστα' ΠΕΡΙΠΤΩΣΗ >=14 ΓΡΑΨΕ 'Καλός' ΠΕΡΙΠΤΩΣΗ >=9.5 ΓΡΑΨΕ 'Μέτριος' ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ ΓΡΑΨΕ 'Απορρίπτεται' ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

4 ΠΕΡΙΠΤΩΣΕΙΣ
4 ΕΛΕΓΧΟΙ ΠΑΝΤΑ

Ισοδύναμο χωρίς φωλιασμένη επιλογή




```
ΑΝ Βαθμός >= 18.5 ΤΟΤΕ
        ΓΡΑΨΕ 'Άριστα'
ΤΕΛΟΣ_ΑΝ
ΑΝ Βαθμός < 18.5 ΚΑΙ Βαθμός >= 14 ΤΟΤΕ
        ΓΡΑΨΕ 'Καλός'
ΤΕΛΟΣ_ΑΝ
ΑΝ Βαθμός < 14 ΚΑΙ Βαθμός >= 9.5 ΤΟΤΕ
        ΓΡΑΨΕ 'Μέτριος'
ΤΕΛΟΣ_ΑΝ
ΑΝ Βαθμός < 9.5 ΤΟΤΕ
        ΓΡΑΨΕ 'Απορρίπτεται'
ΤΕΛΟΣ_ΑΝ
```

ΣΤΙΣ ΑΠΛΕΣ ΑΝ, ΟΤΑΝ ΥΠΑΡΧΟΥΝ ΔΥΟ ΟΡΙΑ ΣΕ ΜΙΑ ΠΕΡΙΟΧΗ ΤΑ ***ΓΡΑΦΟΥΜΕ ΚΑΙ ΤΑ ΔΥΟ ΟΡΙΑ ΥΠΟΧΡΕΩΤΙΚΑ***

Από ότι παρατηρούμε στο παραπάνω παράδειγμα με την εμφωλευμένη επιλογή χρειαζόμαστε 3 ελέγχους από τους οποίους θα εκτελεστούν ή ένας ή δύο ή τρεις έλεγχοι (ανάλογα με το ποια συνθήκη θα ικανοποιηθεί) στην χειρότερη περίπτωση, ενώ με απλές επιλογές χρειαζόμαστε 4 ελέγχους που θα εκτελεστούν όλοι υποχρεωτικά.

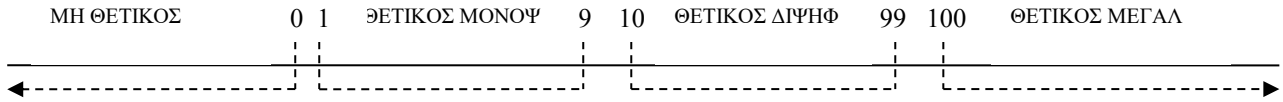
Προσοχή στην μετατροπή απλών ΑΝ ή εμφωλευμένων με περιοχές όταν θέλουμε να τη μετατρέψουμε σε εμφωλευμένες ή πολλαπλή με μία μόνο σύγκριση (**χωρίς λογικές πράξεις**) σε κάθε συνθήκη, **τότε ξεκινάμε από το ένα άκρο και πηγαίνουμε προς το άλλο.**

```
ΑΝ X >= 1 ΚΑΙ X <=9 ΤΟΤΕ
      ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΜΟΝΟΨΗΦΙΟΣ
ΑΛΛΙΩΣ_ΑΝ X>=10 ΚΑΙ X <=99 ΤΟΤΕ
      ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΔΙΨΗΦΙΟΣ'
ΑΛΛΙΩΣ_ΑΝ X>=100 ΤΟΤΕ
      ΓΡΑΨΕ 'ΜΕΓΑΛΟΣ ΘΕΤΙΚΟΣ'
ΑΛΛΙΩΣ
      ΓΡΑΨΕ 'ΜΗ ΘΕΤΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
```

```
ΕΠΙΛΕΞΕ Χ ! ΜΟΝΟ ΟΤΑΝ Χ ΑΚΕΡΑΙΟΣ .. ΣΕ ΠΕΡΙΟΧΕΣ
      ΠΕΡΙΠΤΩΣΗ 1..9
            ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΜΟΝΟΨΗΦΙΟΣ
      ΠΕΡΙΠΤΩΣΗ 10..99
            ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΔΙΨΗΦΙΟΣ'
      ΠΕΡΙΠΤΩΣΗ >=100
            ΓΡΑΨΕ 'ΜΕΓΑΛΟΣ ΘΕΤΙΚΟΣ'
      ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
            ΓΡΑΨΕ 'ΜΗ ΘΕΤΙΚΟΣ'
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

ΛΥΣΗ

Πρώτα φτιάχνουμε τον άξονα με τις περιοχές και τα όρια και στην συνέχεια ξεκινάμε τους ελέγχους από το ένα άκρο και πηγαίνουμε προς το άλλο με τη σειρά.



Α) Στην παρακάτω λύση ξεκινήσαμε τους ελέγχους από το κάτω άκρο και σε κάθε περιοχή παίρνουμε και ελέγχουμε αν η μεταβλητή είναι μικρότερη από το πάνω όριο της περιοχής.

ΠΟΛΛΑΠΛΗ_ΑΝ **ΕΠΙΛΕΞΕ**

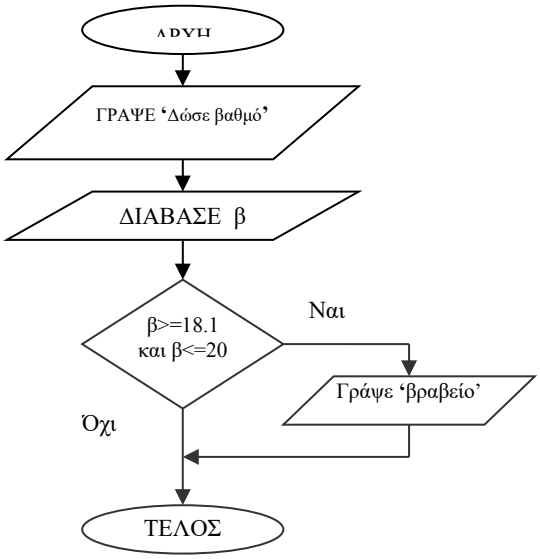
```
ΑΝ X<=0 ΤΟΤΕ
      ΓΡΑΨΕ ' ΜΗ ΘΕΤΙΚΟΣ '
ΑΛΛΙΩΣ_ΑΝ X <=9 ΤΟΤΕ
      ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΜΟΝΟΨΗΦΙΟΣ'
ΑΛΛΙΩΣ_ΑΝ  X <=99 ΤΟΤΕ
      ΓΡΑΨΕ 'ΘΕΤΙΚΟΣ ΔΙΨΗΦΙΟΣ'
ΑΛΛΙΩΣ
      ΓΡΑΨΕ 'ΜΕΓΑΛΟΣ ΘΕΤΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
```

Β) Να ξεκινήσετε από την πάνω περιοχή και να πηγαίνετε με τη σειρά προς την κάτω περιοχή. Να φτιάξετε το αντίστοιχο τμήμα αλγορίθμου.

ΠΟΛΛΑΠΛΗ_ΑΝ **ΕΠΙΛΕΞΕ**

ΠΑΡΑΔΕΙΓΜΑ 1 ΕΠΙΛΟΓΗΣ (απλή επιλογή)

*Να φτιαχτεί Πρόγραμμα που θα διαβάζει τον βαθμό ενός μαθητή και θα μας εμφανίζει το μήνυμα 'βραβείο' αν ο βαθμός είναι μεγαλύτερος ή ίσος του 18,1. (*Στον έλεγχο εκτελούμε κάτι, όταν ισχύει η συνθήκη*)*

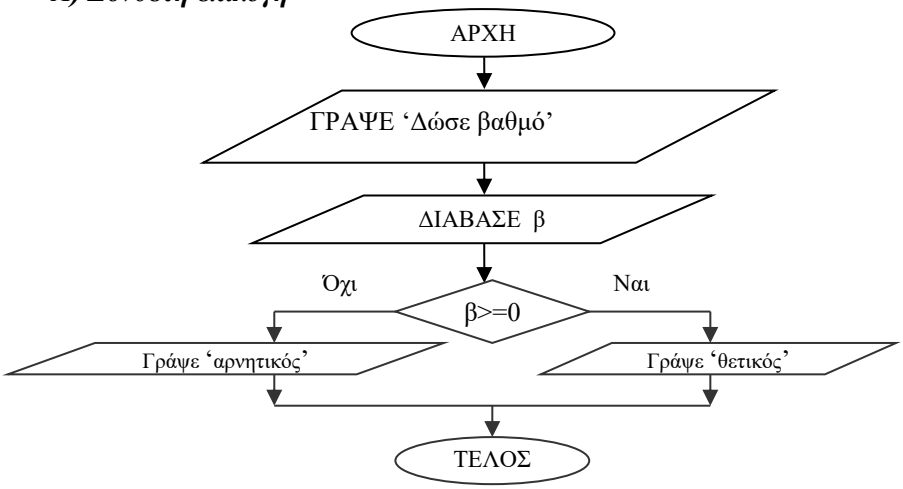



```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ

ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β > = 18.1 ΚΑΙ β<=20 ΤΟΤΕ
        ΓΡΑΨΕ 'βραβείο'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΠΑΡΑΔΕΙΓΜΑ 2 ΕΠΙΛΟΓΩΝ

Να φτιαχτεί Πρόγραμμα που θα διαβάζει έναν αριθμό και μας εμφανίζει αν είναι 'θετικός' ή 'αρνητικός'

Α) Σύνθετη επιλογή

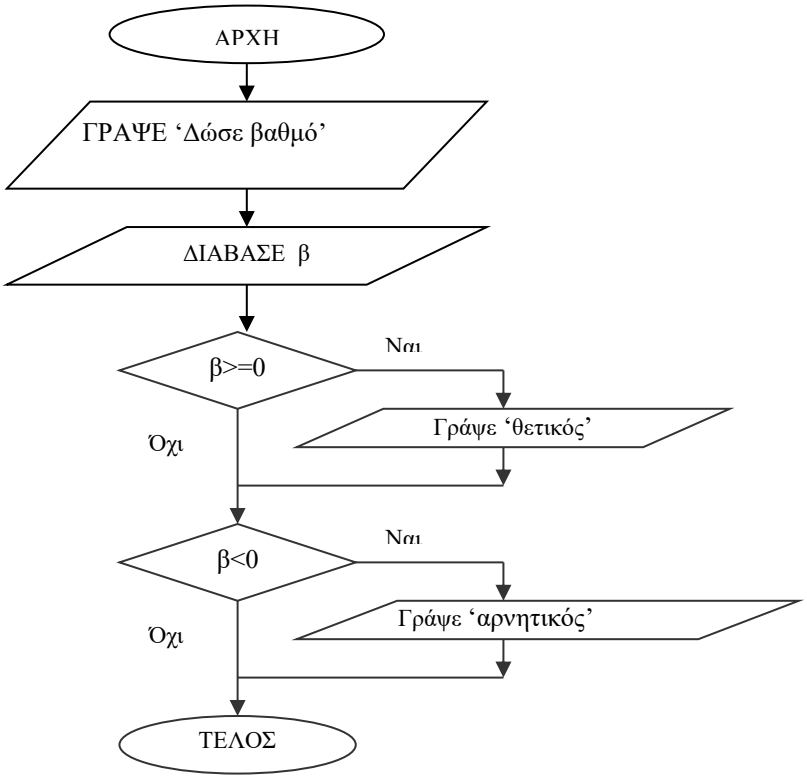



```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β > = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'θετικός'
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'αρνητικός'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Από ότι φαίνεται και από το διάγραμμα ροής της προηγούμενης σελίδας ελέγχουμε ποια από τις δύο περιπτώσεις ισχύει και το διάγραμμα ακολουθεί αυτή τη διαδρομή και εμφανίζει υποχρεωτικά ένα από τα δύο μηνύματα. Ο παραπάνω αλγόριθμος είναι σωστός διότι θεωρήσαμε το 0 ότι είναι θετικός. Αν δεν θεωρούσαμε το 0 θετικό, αλλά άλλη περίπτωση δεν θα γινόταν με σύνθετη ΑΝ αφού όταν δεν ήταν θετικός θα υπήρχαν δύο περιπτώσεις (αρνητικός και μηδέν) και θα χρειαζόμασταν και άλλο έλεγχο στο όχι της συνθήκης (εμφωλευμένη επιλογή), διότι οι περιπτώσεις είναι τρεις τώρα

Β) 2 απλές επιλογές

Το παραπάνω παράδειγμα μπορεί να γραφτεί και με δύο απλές επιλογές όσες δηλαδή και οι περιπτώσεις.




```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β > = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'θετικός'
    ΤΕΛΟΣ_ΑΝ
    ΑΝ β < 0 ΤΟΤΕ ! ΟΧΙ(β>=0)
        ΓΡΑΨΕ 'αρνητικός'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

<u>Προσοχή στην ΕΠΙΛΕΞΕ</u>

<u>ΛΑΘΟΣ</u>

ΠΕΡΙΠΤΩΣΗ <1 Ή >20 (ΤΟ ΣΠΑΜΕ ΣΕ ΔΥΟ ΠΕΡΙΠΤΩΣΕΙΣ)
ΠΕΡΙΠΤΩΣΗ >1 ΚΑΙ <20 (ΟΤΑΝ ΕΧΟΥΜΕ ΠΕΡΙΟΧΕΣ ΞΕΚΙΝΑΜΕ ΑΠΟ ΤΟ ΕΝΑ ΑΚΡΟ)

<u>ΣΩΣΤΑ</u>

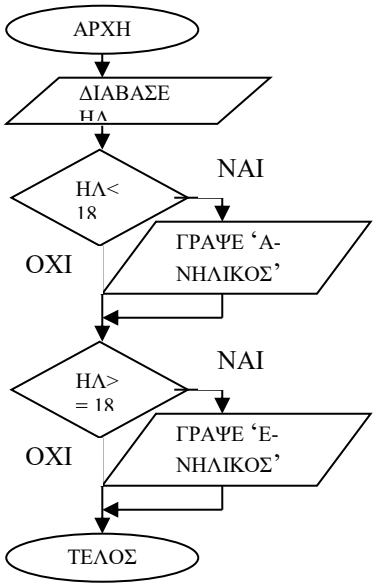
ΠΕΡΙΠΤΩΣΗ 1, 3, 5, 7 ! 1 ή 3 ή 5 ή 7
ΠΕΡΙΠΤΩΣΗ 1..10 ! ΑΠΟ 1-10 (***<u>Η ΕΚΦΡΑΣΗ ΠΟΥ ΕΛΕΓΧΟΥΜΕ ΠΑΙΡΝΕΙ ΜΟΝΟ ΑΚΕΡΑΙΕΣ ΤΙΜΕΣ)</u>***

Η δομή ***ΕΠΙΛΕΞΕ*** λόγω της συμπαγούς δομής προσφέρει ***σημαντικά πλεονεκτήματα*** στον προγραμματισμό.

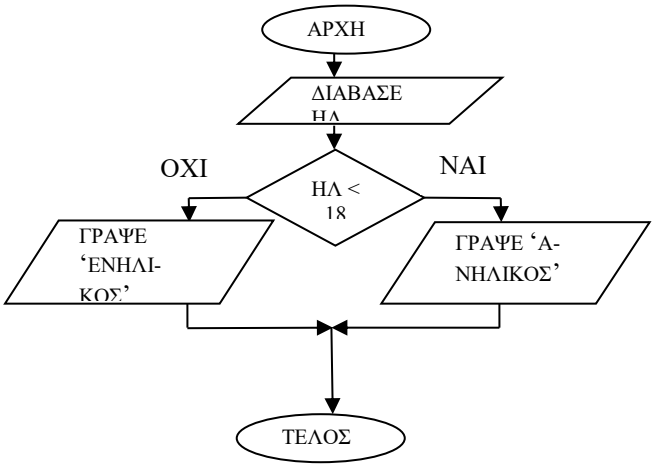
ΑΣΚΗΣΗ ΜΕ ΔΥΟ ΜΟΝΑΔΙΚΕΣ ΠΕΡΙΠΤΩΣΕΙΣ.

Να φτιαχτεί αλγόριθμος που θα διαβάζει την ηλικία ενός ανθρώπου και είναι μικρότερη των 18 ετών να εμφανίζει το μήνυμα «ανήλικος», ενώ αν είναι μεγαλύτερη ή ίση των 18 ετών «ενήλικος».

ΔΙΑΓΡΑΜΜΑ ΜΕ 2 ΑΠΛΕΣ



ΔΙΑΓΡΑΜΜΑ ΜΕ ΜΙΑ ΣΥΝΘΕΤΗ, ΑΦΟΥ ΟΤΑΝ ΔΕΝ ΙΣΧΥΕΙ Η ΜΙΑ ΠΕΡΙΠΤΩΣΗ (<18) ΜΕΝΕΙ ΜΟΝΟ Η ΑΛΛΗ (>=18)




```
ΔΙΑΒΑΣΕ ΗΛ
ΑΝ ΗΛ < 18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΑΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
ΑΝ ΗΛ >=18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΕΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
```

```
ΔΙΑΒΑΣΕ ΗΛ
ΑΝ ΗΛ < 18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΑΝΗΛΙΚΟΣ'
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΕΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
```

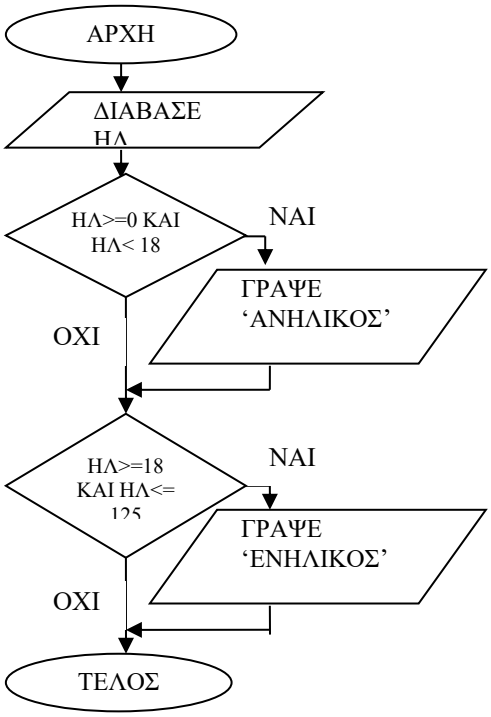
```
ΔΙΑΒΑΣΕ ΗΛ
ΕΠΙΛΕΞΕ ΗΛ
    ΠΕΡΙΠΤΩΣΗ < 18
        ΓΡΑΨΕ 'ΑΝΗΛΙΚΟΣ'
    ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΕΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

Με δύο περιπτώσεις δεν ενδείκνυται η ΕΠΙΛΕΞΕ, ούτε η ΑΛΛΙΩΣ_ΑΝ

ΑΣΚΗΣΗ ΜΕ ΕΛΕΓΧΟ ΤΩΝ 2 ΠΕΡΙΠΤΩΣΕΩΝ ΑΠΌ ΤΙΣ ΤΕΣΣΕΡΙΣ ΠΕΡΙΠΤΩΣΕΙΣ.

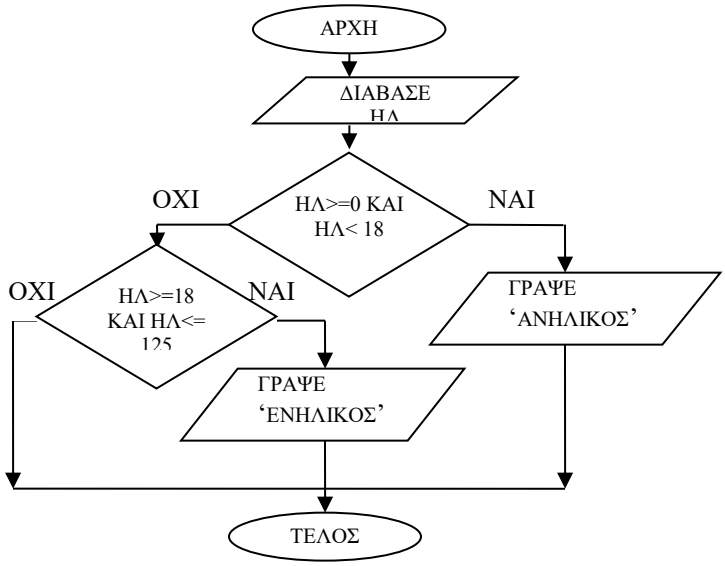
Να φτιαχτεί αλγόριθμος που θα διαβάζει την ηλικία ενός ανθρώπου και είναι μεγαλύτερη των 0 ετών και μικρότερη των 18 ετών να εμφανίζει το μήνυμα «ανήλικος», ενώ αν είναι μεγαλύτερη ή ίση των 18 και μικρότερη των 125 ετών να εμφανίζει «ενήλικος».

ΔΙΑΓΡΑΜΜΑ ΜΕ 2 ΑΠΛΕΣ



ΠΡΟΣΟΧΗ

Η ΣΥΓΚΕΚΡΙΜΕΝΗ ΑΣΚΗΣΗ ΔΕΝ ΓΙΝΕΤΑΙ ΜΕ ΣΥΝΘΕΤΗ ΕΠΙΛΟΓΗ ΓΙΑΤΙ ΕΧΟΥΜΕ ΣΥΝΟΛΙΚΑ 4 ΠΕΡΙΠΤΩΣΕΙΣ ΚΑΙ ΕΜΕΙΣ ΕΛΕΓΧΟΥΜΕ ΜΟΝΟ ΤΙΣ ΔΥΟ. ΑΚΟΛΟΥΘΕΙ ΤΟ ΔΙΑΓΡΑΜΜΑ ΜΕ ΕΜΦΩΛΕΥΜΕΝΗ, ΑΦΟΥ ΟΤΑΝ ΔΕΝ ΙΣΧΥΕΙ Η ΜΙΑ ΠΕΡΙΠΤΩΣΗ [0, 18) ΔΕΝ ΜΕΝΕΙ ΜΟΝΟ Η ΑΛΛΗ [18,125], ΑΛΛΑ ΚΑΙ ΟΙ ΠΕΡΙΟΧΕΣ <0 Ή >125




```
ΔΙΑΒΑΣΕ ΗΛ
ΑΝ ΗΛ >=0 ΚΑΙ ΗΛ < 18  ΤΟΤΕ
    ΓΡΑΨΕ 'ΑΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
ΑΝ ΗΛ >=18  ΚΑΙ ΗΛ <= 125 ΤΟΤΕ
    ΓΡΑΨΕ 'ΕΝΗΛΙΚΟΣ'
ΤΕΛΟΣ_ΑΝ
```

```
ΔΙΑΒΑΣΕ ΗΛ
ΑΝ ΗΛ >= 0 ΚΑΙ ΗΛ < 18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΑΝΗΛΙΚΟΣ'
ΑΛΛΙΩΣ
    ΑΝ ΗΛ >=18 ΚΑΙ ΗΛ <= 125 ΤΟΤΕ
        ΓΡΑΨΕ 'ΕΝΗΛΙΚΟΣ'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
```

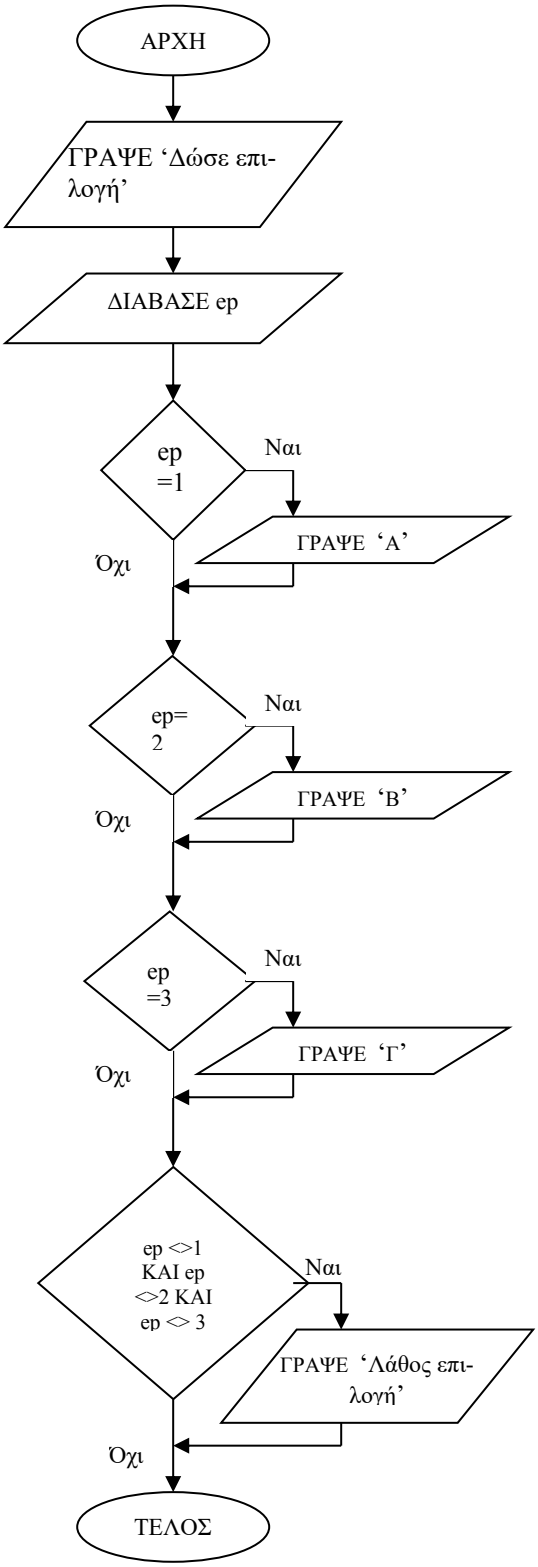
```
ΔΙΑΒΑΣΕ ΗΛ
ΕΠΙΛΕΞΕ ΗΛ
    ΠΕΡΙΠΤΩΣΗ < 0
        ΓΡΑΨΕ ¨ΛΑΘΟΣ¨
    ΠΕΡΙΠΤΩΣΗ <18
        ΓΡΑΨΕ ¨ΑΝΗΛΙΚΟΣ¨
    ΠΕΡΙΠΤΩΣΗ <=125
        ΓΡΑΨΕ ¨ΕΝΗΛΙΚΟΣ¨
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
```

1ο ΠΑΡΑΔΕΙΓΜΑ ΕΠΙΛΟΓΩΝ (3 και πάνω)

Να φτιαχτεί αλγόριθμος που θα διαβάζει έναν αριθμό και θα μας εμφανίζει το αντίστοιχο γράμμα της αλφαβήτου (‘Α’, ή ‘Β’,ή ‘Γ’) αν είναι το 1, 2 ή 3, διαφορετικά το μήνυμα ‘Λάθος επιλογή’

Απλή Επιλογή

Διάγραμμα ροής

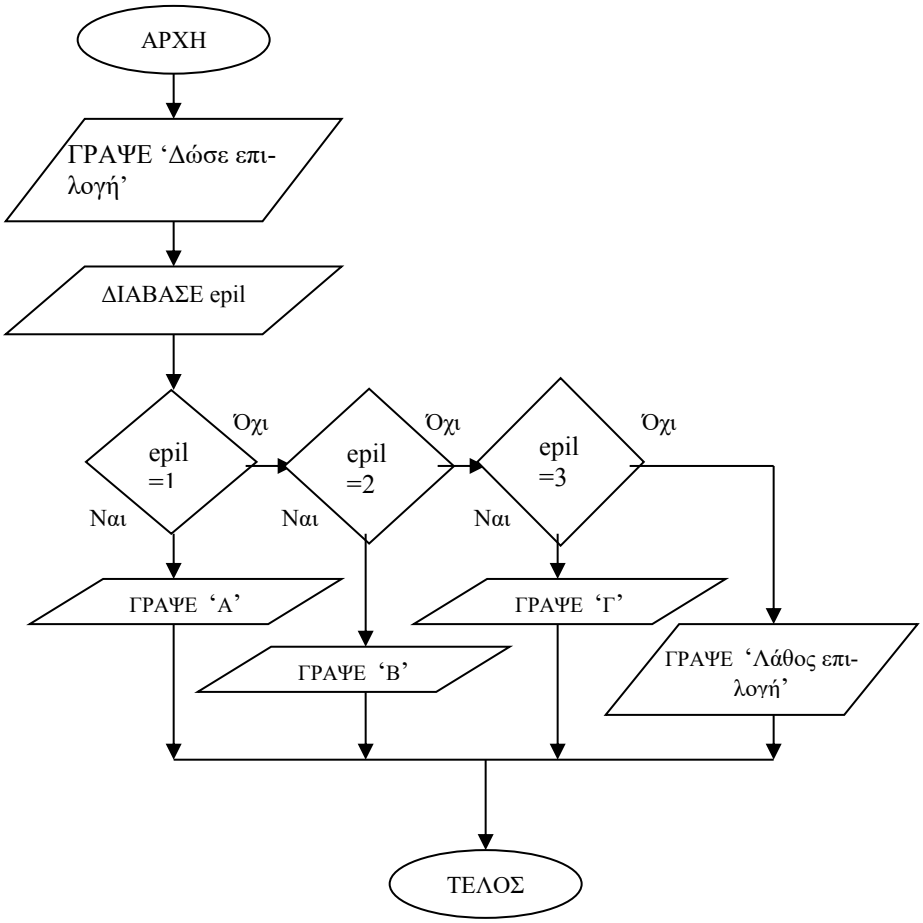


Κώδικας

```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ1

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : ep

ΑΡΧΗ
    ΓΡΑΨΕ ‘Δώσε επιλογή’
    ΔΙΑΒΑΣΕ ep
    ΑΝ ep = 1 ΤΟΤΕ
        ΓΡΑΨΕ ‘Α’
    ΤΕΛΟΣ_ΑΝ
    ΑΝ ep =2 ΤΟΤΕ
        ΓΡΑΨΕ ‘Β’
    ΤΕΛΟΣ_ΑΝ
    ΑΝ ep =3 ΤΟΤΕ
        ΓΡΑΨΕ ‘Γ’
    ΤΕΛΟΣ_ΑΝ
    ΑΝ ep < >1 ΚΑΙ ep < >2 ΚΑΙ ep < >3   ΤΟΤΕ
        ΓΡΑΨΕ ‘Λάθος επιλογή’
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Χρησιμοποιούμε τόσες απλές επιλογές όσες είναι και οι περιπτώσεις δηλ 4 (1, 2, 3 ή κανένα από αυτά).
Προσοχή: Μόνο απλές επιλογές και καμία σύνθετη επιλογή με αυτό τον τρόπο επίλυσης του προβλήματος, γιατί θα έχει λάθος αποτελέσματα. Το πιο συνηθισμένο λάθος είναι να βάλω στον 3ο έλεγχο αλλιώς και να γράψω το μήνυμα ‘Λάθος επιλογή’.

Εμφωλευμένη Επιλογή



<u>Α' τρόπος</u> (Αν … Αλλιώς Αν …..)

Χρησιμοποιούμε τόσα ΤΕΛΟΣ_ΑΝ, όσα και τα ΑΝ και ξεκινάμε πάντα και κλείνουμε τους ελέγχους από τον τελευταίο που ανοίξαμε προς τον πρώτο (από μέσα προς τα έξω)

```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ2
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : epil
ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε επιλογή'
    ΔΙΑΒΑΣΕ epil
    ΑΝ epil = 1 ΤΟΤΕ
        ΓΡΑΨΕ 'Α'
    ΑΛΛΙΩΣ
        ΑΝ epil = 2 ΤΟΤΕ
            ΓΡΑΨΕ 'Β'
        ΑΛΛΙΩΣ
            ΑΝ epil = 3 ΤΟΤΕ
                ΓΡΑΨΕ 'Γ'
            ΑΛΛΙΩΣ
                ΓΡΑΨΕ 'Λάθος επιλογή'
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

<u>Β' τρόπος</u> (Αν … Αλλιώς_αν ……)

```
ΠΡΟΓΡΑΜΜΑ ΕΠΙΛΟΓΩΝ3

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : epil

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε επιλογή'
    ΔΙΑΒΑΣΕ epil
    ΑΝ epil = 1 ΤΟΤΕ
        ΓΡΑΨΕ 'Α'
    ΑΛΛΙΩΣ_ΑΝ epil = 2 ΤΟΤΕ
        ΓΡΑΨΕ 'Β'
    ΑΛΛΙΩΣ_ΑΝ epil = 3 ΤΟΤΕ
        ΓΡΑΨΕ 'Γ'
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Λάθος επιλογή'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Γ' τρόπος (Επίλεξε <μεταβλητή> Περίπτωση …)

Εφαρμόζεται μόνο στην περίπτωση που ελέγχουμε ***μία μεταβλητή ή μία παράσταση*** τι τιμές παίρνει ή τι όρια έχει και ανάλογα εκτελεί διαφορετικές ομάδες εντολών.

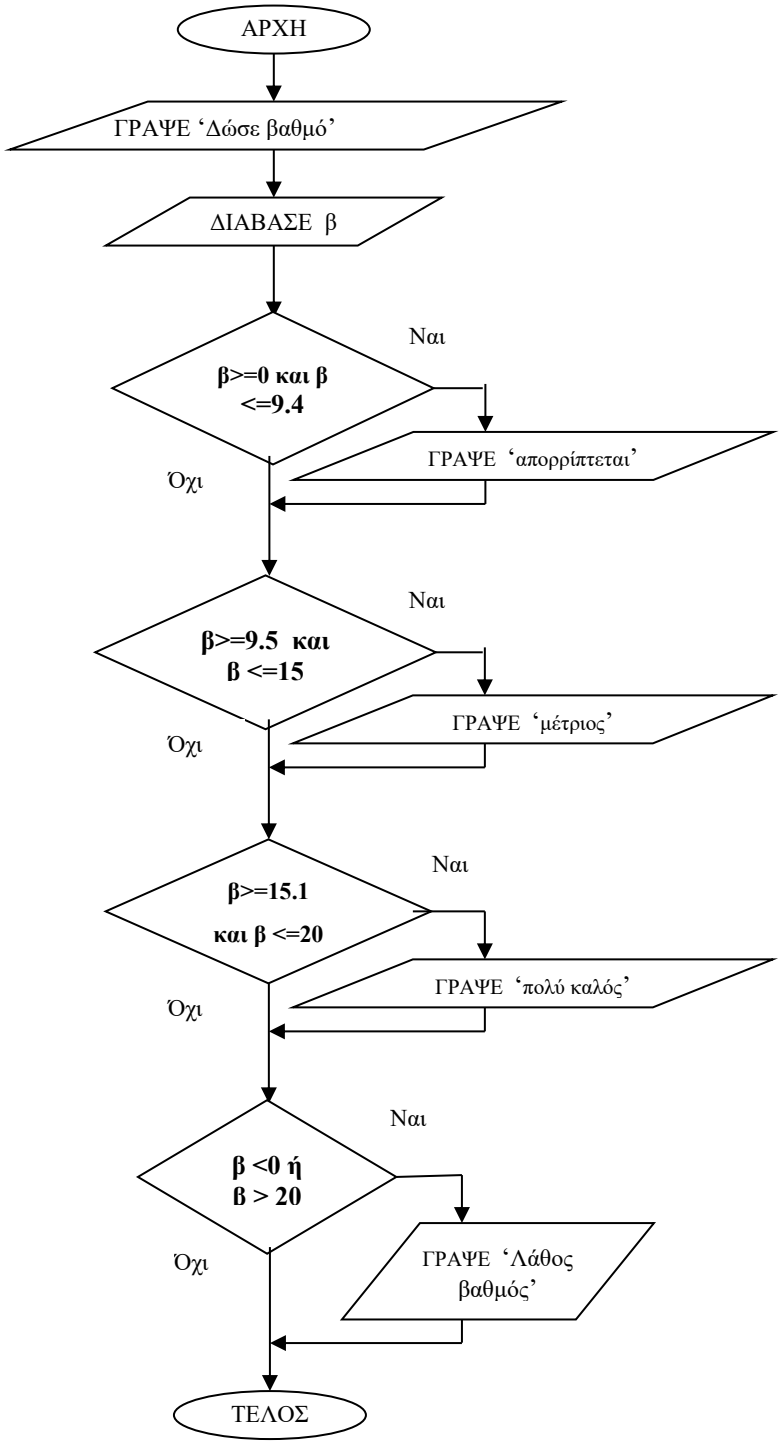
```
ΑΛΓΟΡΙΘΜΟΣ ΕΠΙΛΟΓΩΝ4

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : epil

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε επιλογή'
    ΔΙΑΒΑΣΕ epil
    ΕΠΙΛΕΞΕ epil                    !Μετά την επίλεξε γράφουμε μία μεταβλητή ή έκφραση
        ΠΕΡΙΠΤΩΣΗ 1                 !Μετά την περίπτωση γράφουμε τιμή ή τιμές ή όριο
            ΓΡΑΨΕ 'Α'
        ΠΕΡΙΠΤΩΣΗ 2                 !Δεν μπορούμε να γράψουμε δύο όρια πχ <0  ή  >20,
            ΓΡΑΨΕ 'Β'               !αλλά πρέπει να το γράψουμε σε 2 ξεχωριστές περιπτώσεις
        ΠΕΡΙΠΤΩΣΗ 3
            ΓΡΑΨΕ 'Γ'
        ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
            ΓΡΑΨΕ 'Λάθος επιλογή'
    ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
ΤΕΛΟΣ ΕΠΙΛΟΓΩΝ4
```

2ο ΠΑΡΑΔΕΙΓΜΑ ΕΠΙΛΟΓΩΝ (3 και πάνω)

Να φτιαχτεί αλγόριθμος που θα διαβάζει τον βαθμό ενός μαθητή και θα μας εμφανίζει 'απορρίπτεται' όταν ο βαθμός του είναι 0-9,4 , 'μέτριος' για 9,5-15 , 'πολύ καλός' για 15,1 – 20 και 'λάθος βαθμός' όταν δεν είναι αποδεκτός.

Απλή Επιλογή

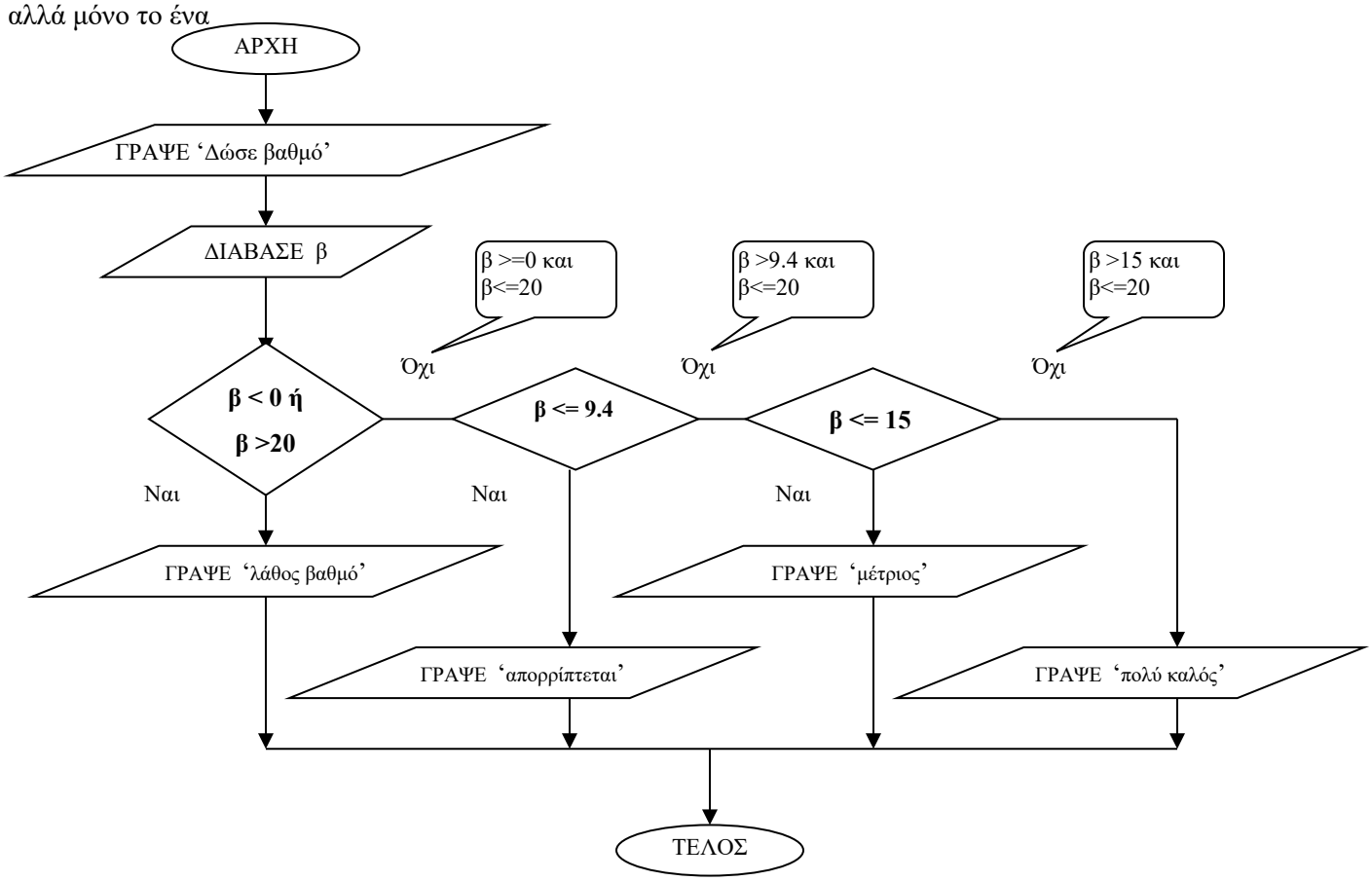
Όταν χρησιμοποιούμαι απλή επιλογή για όρια πρέπει να **γράφουμε και το πάνω αλλά και το κάτω όριο υποχρεωτικά** σε όλες τις περιπτώσεις (τέσσερις).




```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ2

ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β >= 0 ΚΑΙ β<= 9.4 ΤΟΤΕ
        ΓΡΑΨΕ 'απορρίπτεται'
    ΤΕΛΟΣ_ΑΝ
    ΑΝ β>=9.5 ΚΑΙ β <= 15 ΤΟΤΕ
        ΓΡΑΨΕ 'μέτριος'
    ΤΕΛΟΣ_ΑΝ
    ΑΝ β >= 15.1 ΚΑΙ β<= 20 ΤΟΤΕ
        ΓΡΑΨΕ 'πολύ καλός'
    ΤΕΛΟΣ_ΑΝ
    ΑΝ β < 0 Ή β > 20  ΤΟΤΕ
        ΓΡΑΨΕ 'Λάθος βαθμός'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Εμφωλευμένη Επιλογή

Σε τέτοια παραδείγματα με όρια τιμών όπου έχουμε πολλές περιπτώσεις ξεκινάμε από το ένα άκρο και κατευθυνόμαστε προς το άλλο (το μικρότερο όριο και πηγαίνουμε προς τα πάνω ή από το μεγαλύτερο όριο και πηγαίνουμε προς το μικρότερο τον έλεγχο των περιπτώσεων), ώστε να μην ελέγχουμε και τα δύο άκρα της κάθε περίπτωσης., αλλά μόνο το ένα



Α' τρόπος (Αν … Αλλιώς Αν …..)

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ2
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : β
ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β < 0 Ή β > 20 ΤΟΤΕ
        ΓΡΑΨΕ 'λάθος βαθμός'
    ΑΛΛΙΩΣ
        ΑΝ β <= 9,4 ΚΑΙ β >= 0  ΤΟΤΕ
! η αχνή γραφή σημαίνει πλεονασμό (δεν χρειάζεται)
            ΓΡΑΨΕ 'απορρίπτεται'
        ΑΛΛΙΩΣ
            ΑΝ β <= 15 ΚΑΙ β >= 9,5  ΤΟΤΕ
                ΓΡΑΨΕ 'μέτριος'
            ΑΛΛΙΩΣ
                ΓΡΑΨΕ 'πολύ καλός'
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Β' τρόπος (Αν … Αλλιώς_αν ……)

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ2

ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε βαθμό'
    ΔΙΑΒΑΣΕ β
    ΑΝ β < 0 Ή β > 20 ΤΟΤΕ
        ΓΡΑΨΕ 'λάθος βαθμός'
    ΑΛΛΙΩΣ_ΑΝ β <= 9,4 ΚΑΙ β >= 0 ΤΟΤΕ
! η αχνή γραφή σημαίνει πλεονασμό (δεν χρειάζεται)
        ΓΡΑΨΕ 'απορρίπτεται'
    ΑΛΛΙΩΣ_ΑΝ β <= 15 ΚΑΙ β >= 9,5 ΤΟΤΕ
        ΓΡΑΨΕ 'μέτριος'
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'πολύ καλός'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Γ' τρόπος (Επίλεξε <μεταβλητή> Περίπτωση …)

Εφαρμόζεται μόνο στην περίπτωση που ελέγχουμε ***μία μεταβλητή ή μία παράσταση*** σε ποια περίπτωση ανήκει. Η κάθε περίπτωση μπορεί να είναι τιμή/ές, ή περιοχή με ένα όριο μόνο (***οι λογικές πράξεις απαγορεύονται***)

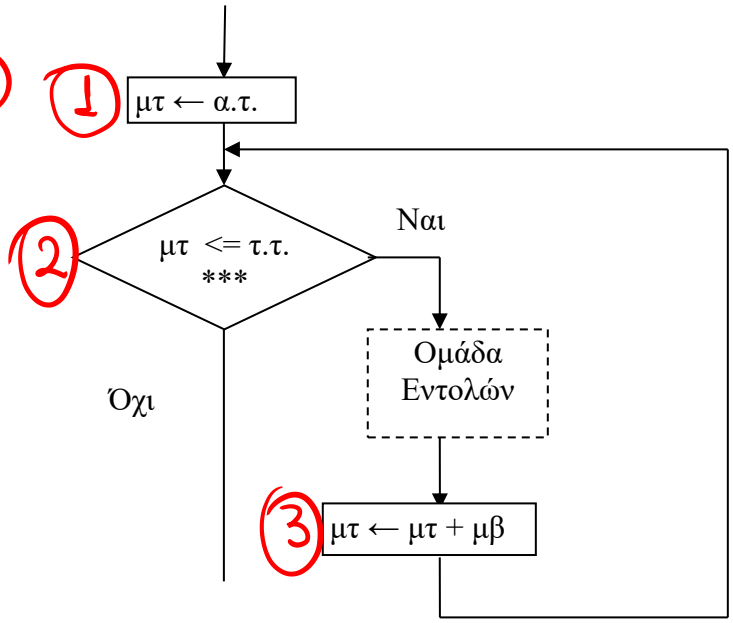
```
ΑΛΓΟΡΙΘΜΟΣ ΠΑΡΑΔΕΙΓΜΑ2

ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : β

ΑΡΧΗ
    ΓΡΑΨΕ ¨Δώσε βαθμό¨
    ΔΙΑΒΑΣΕ β
    ΕΠΙΛΕΞΕ β
        ΠΕΡΙΠΤΩΣΗ <0                    !Δεν μπορούμε να γράψουμε <0 Ή >20 και για αυτό
            ΓΡΑΨΕ 'λάθος βαθμός'        !το γράφουμε σε δύο ξεχωριστές περιπτώσεις
        ΠΕΡΙΠΤΩΣΗ >20
            ΓΡΑΨΕ 'λάθος βαθμός'
        ΠΕΡΙΠΤΩΣΗ <=9,4                 ! ΛΑΘΟΣ 9.4..15,  .. ΜΟΝΟ ΓΙΑ ΑΚΕΡΑΙΟΥΣ
            ΓΡΑΨΕ 'απορρίπτεται'
        ΠΕΡΙΠΤΩΣΗ <=15
            ΓΡΑΨΕ 'μέτριος'
        ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ                ! το διάγραμμα ροής δεν είναι το παραπάνω, αλλά
            ΓΡΑΨΕ 'πολύ καλός'          ! έχει έναν παραπάνω έλεγχο για την ΕΠΙΛΕΞΕ
    ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
ΤΕΛΟΣ ΠΑΡΑΔΕΙΓΜΑ2
```

Επαναληπτικές δομές

Η λογική των επαναληπτικών διαδικασιών εφαρμόζεται όπου μία ακολουθία εντολών εφαρμόζεται σε ένα σύνολο περιπτώσεων που έχουν κάτι κοινό. Όταν ψάχνουμε θέση για να παρκάρουμε κοντά στο σπίτι και κάνουμε κύκλους μέχρι να βρούμε θέση, επανάληψη κεφαλαίου μέχρι να το καταλάβουμε, να διαβάσουμε 100 αριθμούς και να βρούμε το Μέσο Όρο τους.

ΓΙΑ – ΑΠΟ – ΜΕΧΡΙ (ΓΝΩΣΤΟ ΠΛΗΘΟΣ ΕΠΑΝΑΛΗΨΕΩΝ υπάρχει περίπτωση να μην εκτελεστεί καμία φορά, συνθήκη πριν τις εντολές)

ΓΙΑ μτ **ΑΠΟ** ατ **ΜΕΧΡΙ** ττ **ΜΕ ΒΗΜΑ** μβ
    Ομάδα Εντολών
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ



Πχ.
```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ ΒΗΜΑ 3
        ΓΡΑΨΕ Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ Ι ΑΠΟ 20 ΜΕΧΡΙ 1 ΜΕ ΒΗΜΑ -5
        ΓΡΑΨΕ Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        ΔΙΑΒΑΣΕ Ι        ! δεν πρέπει μέσα σε μία ΓΙΑ να μεταβάλλουμε τον μετρητή
        Ι ← …            ! διότι δεν θα ξέρουμε μετά πόσες επαναλήψεις θα γίνουν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Αν το βήμα είναι θετικό ο έλεγχος θα είναι μτ <= τ.τ.

***** Προσοχή. Αν το βήμα είναι αρνητικό ο έλεγχος θα είναι μτ >= τ.τ.**

Αν το βήμα είναι μηδέν εκτελεί πάντοτε άπειρες επαναλήψεις.

Για να γραφεί μία επανάληψη με την εντολή ΓΙΑ πρέπει να είναι γνωστά 3 πράγματα:

1. Αρχική τιμή του μετρητή
2. Τελική τιμή του μετρητή
3. Και το βήμα (σταθερή τιμή) που θα αλλάζει σε κάθε επανάληψη ο μετρητής

Όταν είναι *γνωστός ο αριθμός των επαναλήψεων εκ των προτέρων*, μπορεί να χρησιμοποιηθεί αυτή η δομή επανάληψης που είναι η πιο εύκολη στην σύνταξη και εκτέλεση.

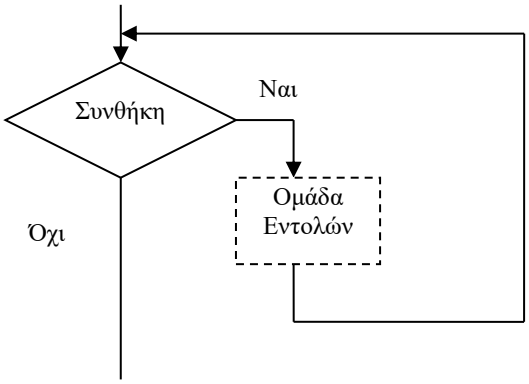
Μόνο όταν το ***βήμα είναι μηδέν*** κάνει ***άπειρες επαναλήψεις.***

Πρέπει να χρησιμοποιηθεί μία μεταβλητή σαν μετρητής, όπως έχουμε αναφέρει για τον έλεγχο του αριθμού των επαναλήψεων και τις μεταβλητές – σταθερές **<ατ> αρχική τιμή μετρητή**, **<ττ> τελική τιμή μετρητή, <μβ> μεταβολή μετρητή** για κάθε επανάληψη που ***μπορεί να είναι και πραγματικές***.

ΟΣΟ – ΕΠΑΝΑΛΑΒΕ (υπάρχει περίπτωση να μην εκτελεστεί καμία φορά, συνθήκη πριν τις εντολές του βρόχου)

ΟΣΟ <συνθήκη συνέχειας> **ΕΠΑΝΑΛΑΒΕ**
    Ομάδα Εντολών
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

```
Πχ
ΔΙΑΒΑΣΕ Χ
ΟΣΟ Χ >=0 ΕΠΑΝΑΛΑΒΕ
        ΔΙΑΒΑΣΕ Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```



Στην αρχή ελέγχεται η συνθήκη και εάν είναι αληθής εκτελείται η ομάδα εντολών. Αυτό γίνεται συνέχεια μέχρι να γίνει η συνθήκη ψευδής.

1. Πολλές φορές είναι γνωστός ο αριθμός επαναλήψεων που θέλουμε να κάνουμε και για αυτό χρησιμοποιούμε μία βοηθητική μεταβλητή η οποία μετρά το πλήθος των επαναλήψεων και καλείται μετρητής. Δεν είναι υποχρεωτικό να ξεκινά από το 1 ο μετρητής. Υπάρχει καλύτερος τρόπος με την **ΓΙΑ …ΑΠΟ … ΜΕΧΡΙ …**
2. Η εντολή ΟΣΟ – ΕΠΑΝΑΛΑΒΕ χρησιμοποιείται κυρίως για επαναλήψεις όπου δεν είναι γνωστός ο αριθμός των επαναλήψεων. Προσοχή στο ότι οι μεταβλητές της συνθήκης πρέπει να μεταβάλλονται μέσα στην ομάδα εντολών, διαφορετικά δεν θα τερματίζει η επανάληψη.
3. Ο έλεγχος της συνθήκης γίνεται στην αρχή οπότε υπάρχει περίπτωση να μην εκτελεστεί ποτέ η ομάδα εντολών.
4. Η ομάδα εντολών εκτελείται όσο η συνθήκη είναι αληθής και όταν γίνει ψευδής τελειώνει και η επανάληψη.

Επίσης για να μετατραπεί μία ΟΣΟ σε ΓΙΑ πρέπει ο έλεγχος να είναι μτ<= ττ (για βήμα θετικό), αλλιώς πρέπει να το μετατρέψουμε (αν γίνεται).

```
Χ ← 13
ΟΣΟ X < 50 ΕΠΑΝΑΛΑΒΕ
        ΓΡΑΨΕ Χ
        Χ ← Χ + 5
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

⇨

```
Χ ← 13
ΟΣΟ X <= 49 ΕΠΑΝΑΛΑΒΕ
        ΓΡΑΨΕ Χ
        Χ ← Χ + 5
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

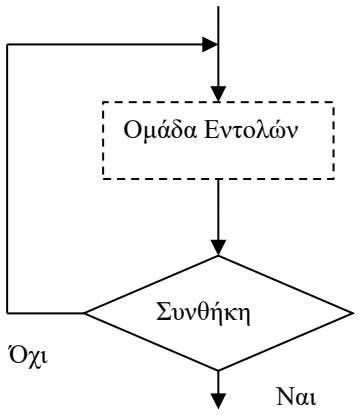
⇨

```
ΓΙΑ Χ ΑΠΟ 13 ΜΕΧΡΙ 49 ΜΕ ΒΗΜΑ 5
        ΓΡΑΨΕ Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ – ΜΕΧΡΙΣ_ΟΤΟΥ (θα εκτελεστεί τουλάχιστον μία φορά, οι εντολές βρόχου βρίσκονται πριν τη συνθήκη)

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
Ομάδα Εντολών
ΜΕΧΡΙΣ_ΟΤΟΥ <συνθήκη τέλους>

Πχ
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
ΔΙΑΒΑΣΕ Χ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ < 0



Σε αντίθεση με την ΟΣΟ – ΕΠΑΝΑΛΑΒΕ, πρώτα εκτελείται μία φορά η ομάδα εντολών και μετά γίνεται ο έλεγχος της συνθήκης. Αν η συνθήκη είναι ψευδής επαναλαμβάνεται η ομάδα εντολών, και σταματάει η επανάληψη όταν η συνθήκη γίνει αληθής.

1. Η εντολή ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ – ΜΕΧΡΙΣ_ΟΤΟΥ …… χρησιμοποιείται και αυτή *κυρίως για επαναλήψεις όπου δεν είναι γνωστός ο αριθμός επαναλήψεων*.
2. Η ομάδα εντολών θα εκτελεστεί τουλάχιστον μία φορά.
3. Η ομάδα εντολών εκτελείται όσο η συνθήκη είναι ψευδής.

Εμφωλευμένοι βρόχοι

Όπως και στην επιλογή έχουμε την εμφωλιασμένη, έτσι και στην επανάληψη υπάρχουν οι εμφωλευμένοι βρόχοι, όπου μέσα στην ομάδα εντολών μίας επανάληψης μπορούμε να έχουμε άλλη επανάληψη. Οι κανόνες που πρέπει να ακολουθούν οι εμφωλευμένοι βρόχοι είναι:

1. Ο εμφωλευμένος βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό. Δηλ η επανάληψη που ξεκινάει τελευταία πρέπει να τελειώνει πρώτη.
2. Η είσοδος κάθε βρόχου γίνεται στην αρχή
3. Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσότερων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.

Εμφωλιασμένες (Φωλιασμένες) δομές

Μπορούμε να «φωλιάσουμε μια δομή μέσα σε μια άλλη (ίδιου ή διαφορετικού τύπου) αρκεί να προσέχουμε τον εξής κανόνα: **αν μια δομή αρχίζει μέσα σε μια άλλη πρέπει να τελειώσει η μέσα πριν την έξω**. Επίσης πρέπει να κοιτάμε όσα ΓΙΑ... (ή ΑΝ... ή ΟΣΟ... κλπ) έχουμε, να έχουμε το ίδιο πλήθος ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ (ή ΤΕΛΟΣ_ΑΝ ή ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ) αντίστοιχα στη σωστή σειρά.

Ομοιότητες-Διαφορές ΟΣΟ, ΓΙΑ, ΜΕΧΡΙΣ_ΟΤΟΥ

Ομοιότητες

1. ΟΣΟ, ΓΙΑ, ΜΕΧΡΙΣ_ΟΤΟΥ: Όλες εργασίες μπορούν να γίνουν με την ΟΣΟ, ενώ το αντίστροφο δεν ισχύει πάντοτε και απαιτεί μεγάλη προσοχή.

2. ΟΣΟ, ΓΙΑ: α) Ο έλεγχος της συνθήκης για την εκτέλεση των εντολών βρόχου γίνεται στην αρχή. Άρα μπορεί να μην εκτελεστούν καμία φορά. Β) Το διάγραμμα ροής της ΓΙΑ είναι ίδιο με της ΟΣΟ, αλλά όχι και ο κώδικας. Γ) Η ομάδα εντολών εκτελείται όσο η συνθήκη είναι αληθής και στις δύο περιπτώσεις.

3. ΟΣΟ, ΜΕΧΡΙΣ_ΟΤΟΥ: Μπορεί να μην τελειώσουν ποτέ (πράγμα λάθος προγραμματιστικά και αλγοριθμικά)

Διαφορές

1. ΟΣΟ, ΓΙΑ με ΜΕΧΡΙΣ_ΟΤΟΥ: Ο έλεγχος στην ΜΕΧΡΙΣ_ΟΤΟΥ γίνεται στο τέλος (οπότε, ότι υπάρχει μέσα στην επανάληψη εκτελείται τουλάχιστον μια φορά) σε αντίθεση με τις ΟΣΟ, ΓΙΑ όπου ο έλεγχος γίνεται στην αρχή.

2. ΟΣΟ και ΓΙΑ με ΜΕΧΡΙΣ_ΟΤΟΥ: Στην ΟΣΟ και την ΓΙΑ οι εντολές (που υπάρχουν μέσα τους) εκτελούνται όσο η συνθήκη είναι αλήθεια, ενώ στην ΜΕΧΡΙΣ_ΟΤΟΥ οι εντολές εκτελούνται μέχρι η συνθήκη να γίνει αλήθεια. Δηλαδή οι λογικές της συνθήκης τους είναι αντίστροφες.

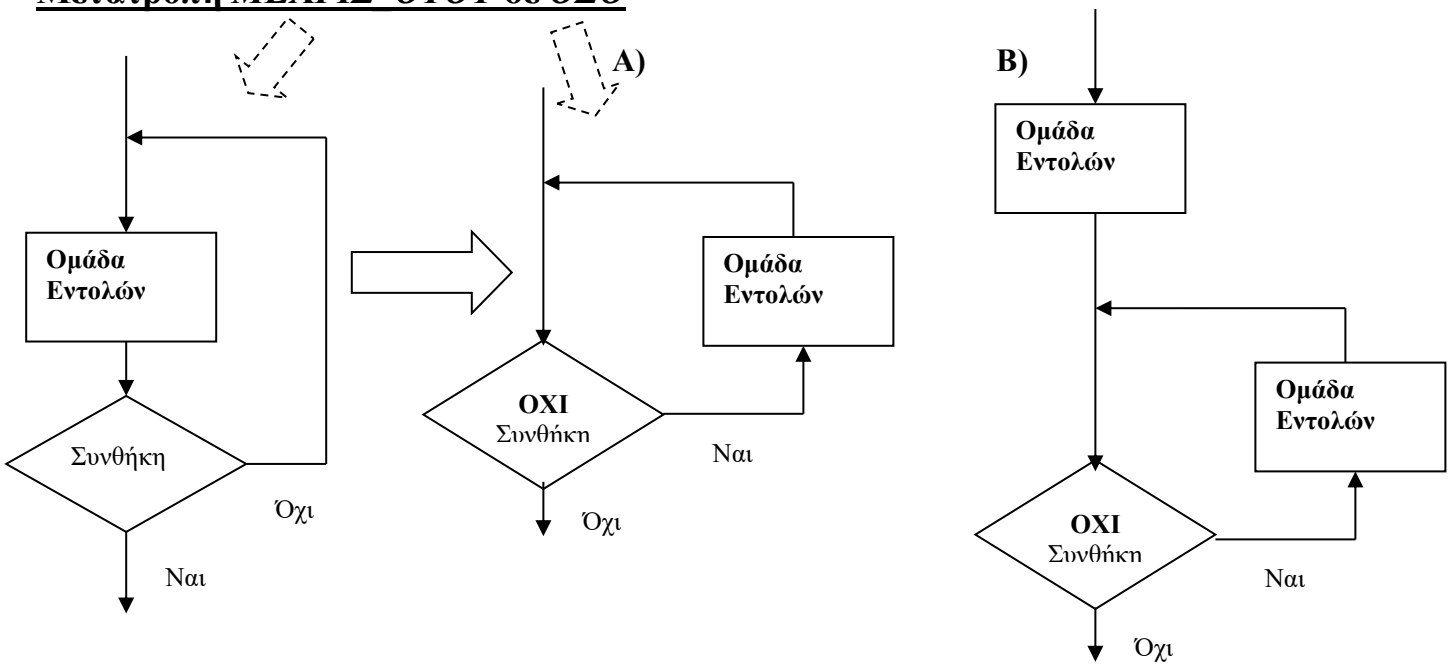
3. ΟΣΟ, ΜΕΧΡΙΣ_ΟΤΟΥ με ΓΙΑ: Στην ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ έχουμε λογικές συνθήκες και τις χρησιμοποιούμε συνήθως όταν δεν γνωρίζουμε τον αριθμό επαναλήψεων, ενώ στην ΓΙΑ την χρησιμοποιούμε όταν είναι γνωστός ο αριθμός των επαναλήψεων και ελέγχουμε έναν μετρητή.

ΟΛΙΣΘΗΣΗ – ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΑΛΑ ΡΩΣΙΚΑ

Ολίσθηση προς τα αριστερά ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η ολίσθηση προς τα δεξιά ισοδυναμεί με την διαίρεση με το δύο στο δυαδικό σύστημα αρίθμησης.

Ο πολλαπλασιασμός αλά ρωσικά επιτυγχάνεται με διαδοχικές ολισθήσεις, που είναι πολύ απλές πράξεις για τον επεξεργαστή, με αποτέλεσμα να γίνεται πιο γρήγορα από ότι με τον κλασσικό τρόπο και ας φαίνεται πιο πολύπλοκος.

Μετατροπή *ΜΕΧΡΙΣ ΟΤΟΥ* σε *ΟΣΟ*



Η συνθήκη στην ΟΣΟ είναι **αντίστροφη** από ότι στην ΜΕΧΡΙΣ_ΟΤΟΥ, αφού στην μία οι εντολές επαναλαμβάνονται όταν η συνθήκη είναι ψευδής, ενώ στην άλλη όταν η συνθήκη είναι αληθής. Προσοχή όταν η ΜΕΧΡΙΣ_ΟΤΟΥ εκτελείται μόνο μία φορά, διότι η ΟΣΟ δεν θα εκτελεστεί καμία, άρα υλοποιούμε την μετατροπή με τον Β) τρόπο σε αυτή την περίπτωση.

Α) Αν η ΜΕΧΡΙΣ ΟΤΟΥ εκτελείται πάνω από μία φορά

Χ ← 10	Χ ← 10
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ	ΟΣΟ **Χ <= 20** ΕΠΑΝΑΛΑΒΕ
ΓΡΑΨΕ Χ	ΓΡΑΨΕ Χ
Χ ← Χ+2	Χ ← Χ+2
ΜΕΧΡΙΣ_ΟΤΟΥ **Χ > 20**	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Β) Αν η ΜΕΧΡΙΣ ΟΤΟΥ εκτελείται μία φορά ή υπάρχει η περίπτωση να εκτελεστούν μία φορά

Επειδή στην ΟΣΟ δεν θα μπει καμία φορά, την ομάδα εντολών την γράφουμε και μία φόρα πριν την επανάληψη, ώστε να εκτελεστεί μία φορά όπως στην ΜΕΧΡΙΣ_ΟΤΟΥ ***(ΠΑΝΤΑ ΣΩΣΤΟ)***

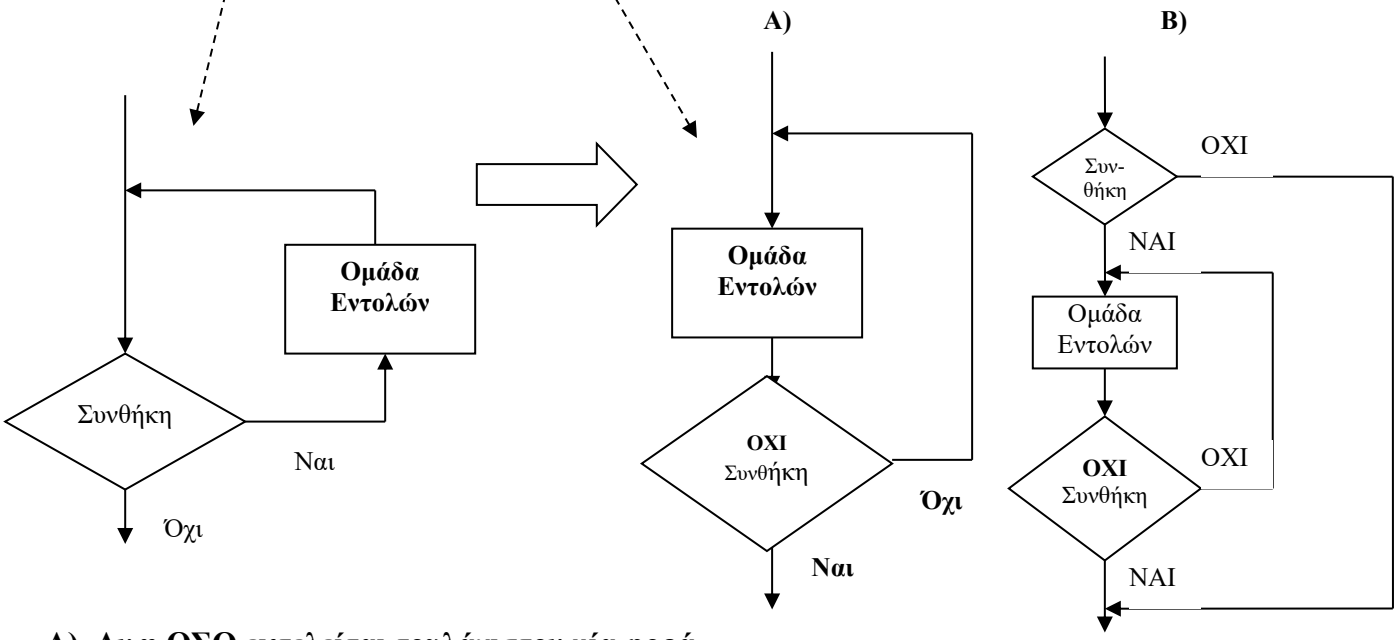
Χ ← 100	Χ ← 100
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ	***ΓΡΑΨΕ Χ*** ! μία φορά πριν την ΟΣΟ
ΓΡΑΨΕ Χ	***Χ ←Χ+2*** ! ώστε να εκτελεστούν 1 φορά
Χ ← Χ+2	ΟΣΟ Χ <= 20 ΕΠΑΝΑΛΑΒΕ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ > 20	ΓΡΑΨΕ Χ
	Χ ← Χ+2
	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Μετατροπή *ΟΣΟ* σε *ΜΕΧΡΙΣ ΟΤΟΥ*

Αυτή η μετατροπή πραγματοποιείται-γίνεται με δύο τρόπους:

Α) **μόνο όταν η επανάληψη στην ΟΣΟ πραγματοποιείται τουλάχιστον μία φορά**, αλλιώς δεν γίνεται με αυτό τον τρόπο, αφού η ΜΕΧΡΙΣ_ΟΤΟΥ θα εκτελείται τουλάχιστον μία φορά.

Β) πάντοτε, αφού πρώτα χρησιμοποιήσουμε μία συνθήκη ΑΝ με την Συνθήκη της ΟΣΟ (έξω από την ΜΕΧΡΙΣ_ΟΤΟΥ), οπότε αν δεν ισχύει δεν εκτελείται η ΟΣΟ όπως και η ΜΕΧΡΙΣ_ΟΤΟΥ καμία φορά.



Α) Αν η ΟΣΟ εκτελείται τουλάχιστον μία φορά

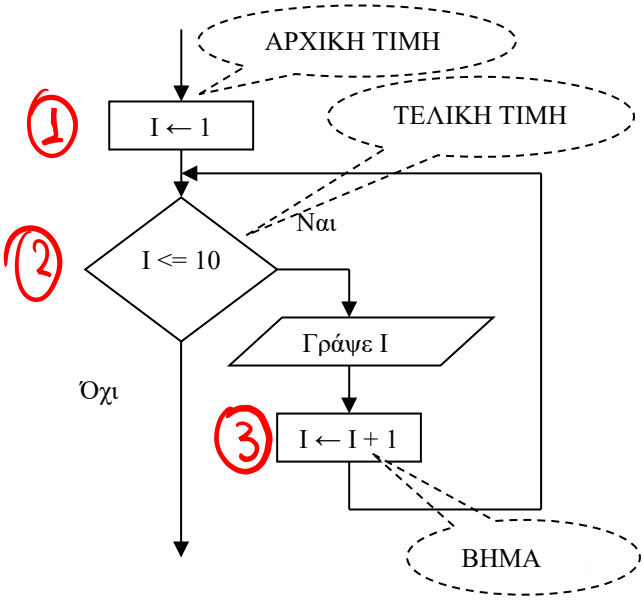
Χ ← 10	Χ ← 10
ΟΣΟ **Χ <= 20** ΕΠΑΝΑΛΑΒΕ	ΑΡΧΗ_ΕΠΑΝΑΛΑΛΗΨΗΣ
ΓΡΑΨΕ Χ	ΓΡΑΨΕ Χ
Χ ← Χ + 3	Χ ← Χ + 3
ΤΕΛΟΣ_ΕΠΑΝΑΛΑΛΗΨΗΣ	ΜΕΧΡΙΣ_ΟΤΟΥ **Χ > 20**

Β) Αν η ΟΣΟ δεν εκτελείται καμία φορά ή υπάρχει η περίπτωση να μην εκτελεστούν

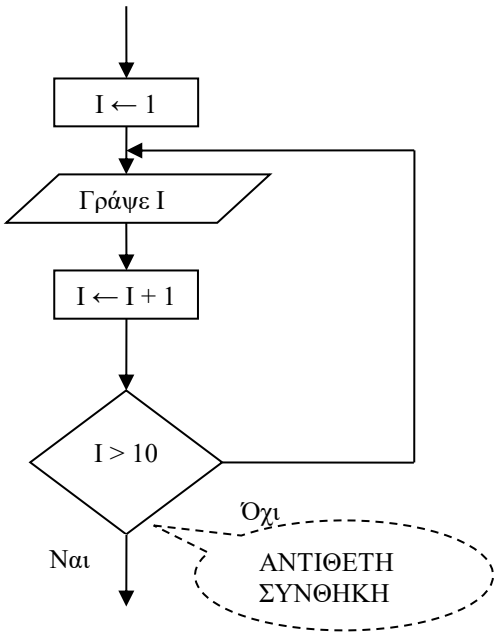
Σ ← 0	Σ ← 0 !***(ΠΑΝΤΑ ΣΩΣΤΟ)***
Χ ← 100	Χ ← 100
ΟΣΟ **Χ <= 20** ΕΠΑΝΑΛΑΒΕ	ΑΝ **Χ <= 20** ΤΟΤΕ ! συνθήκη ΟΣΟ
Σ ← Σ + Χ	ΑΡΧΗ_ΕΠΑΝΑΛΑΛΗΨΗΣ
Χ ← Χ + 3	Σ ← Σ + Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΑΛΗΨΗΣ	Χ ← Χ + 3
ΓΡΑΨΕ Σ	ΜΕΧΡΙΣ_ΟΤΟΥ Χ > 20
	ΤΕΛΟΣ_ΑΝ
	ΓΡΑΨΕ Σ

Μετατροπή ΓΙΑ σε ΜΕΧΡΙΣ_ΟΤΟΥ

Διάγραμμα **ΓΙΑ** ίδιο με **ΟΣΟ**



Διάγραμμα σε **ΜΕΧΡΙΣ_ΟΤΟΥ**



Αν μας ζητηθεί να μετατρέψουμε μία ΓΙΑ σε ΜΕΧΡΙΣ_ΟΤΟΥ την μετατρέπουμε πρώτα σε ΟΣΟ και στη συνέχεια σε ΜΕΧΡΙΣ-ΟΤΟΥ.

Το διάγραμμα της ΓΙΑ είναι ίδιο με της ΟΣΟ, ενώ για να το μετατρέψουμε σε ΜΕΧΡΙΣ_ΟΤΟΥ **πρέπει να εκτελούνται τουλάχιστον μία φορά οι εντολές της ΟΣΟ**. Την ομάδα εντολών την μεταφέρουμε πριν από τον έλεγχο της επανάληψης στην ΜΕΧΡΙΣ_ΟΤΟΥ, και τέλος αντιστρέφουμε την συνθήκη. Στην περίπτωση που δεν εκτελούνται καμία φορά οι εντολές της ΟΣΟ τότε δεν μετατρέπονται σε ΜΕΧΡΙΣ_ΟΤΟΥ (εκτός και βάλουμε την ΜΕΧΡΙΣ_ΟΤΟΥ μέσα σε μία ΑΝ που έχει τη συνθήκη της ΟΣΟ).

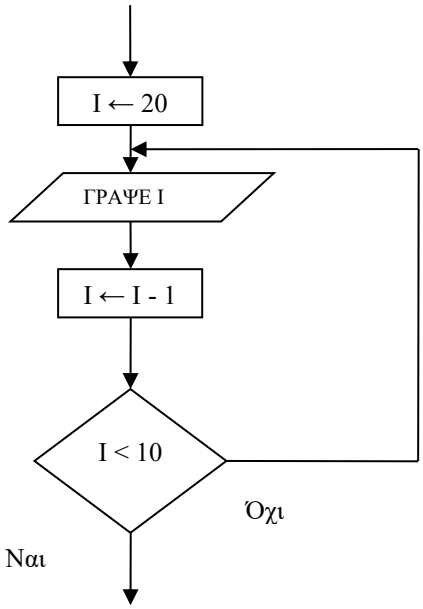
ΓΙΑ	**ΟΣΟ**	**ΜΕΧΡΙΣ_ΟΤΟΥ**
	I ← 1	***I ← 1***
ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ 10	ΟΣΟ I <= 10 ΕΠΑΝΑΛΑΒΕ	ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ I	ΓΡΑΨΕ I	ΓΡΑΨΕ I
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	***I ← I + 1***	***I ← I + 1***
! I = 11	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	ΜΕΧΡΙΣ_ΟΤΟΥ I > 10

Όταν μετατρέπουμε μία ΓΙΑ σε ΟΣΟ έχει δύο εντολές παραπάνω, αφού πρέπει να γράψουμε έξω από την επανάληψη την αρχική τιμή του μετρητή και μέσα στην επανάληψη τελευταία εντολή την αύξηση του μετρητή.

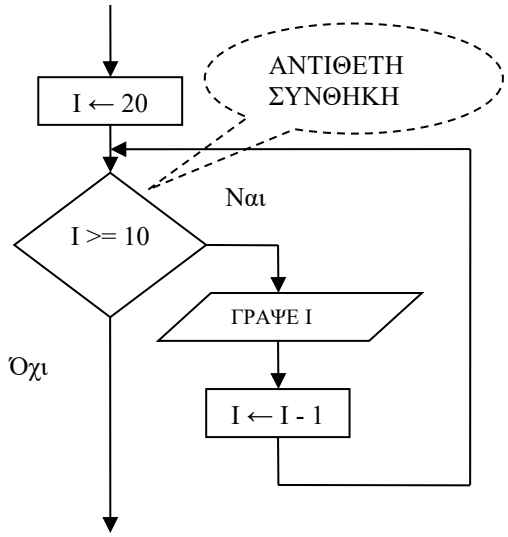
Μετατροπή ΜΕΧΡΙΣ ΟΤΟΥ σε ΓΙΑ

Για μεγαλύτερη ευκολία την μετατρέπουμε την ΜΕΧΡΙΣ_ΟΤΟΥ σε ΟΣΟ πρώτα, γιατί η ΟΣΟ μοιάζει πιο πολύ με την ΓΙΑ, αφού όλες οι ΓΙΑ γίνονται σε ΟΣΟ, αλλά όχι και το αντίστροφο.

Διάγραμμα σε **ΜΕΧΡΙΣ_ΟΤΟΥ**



Διάγραμμα **ΟΣΟ** ίδιο με **ΓΙΑ**



ΜΕΧΡΙΣ_ΟΤΟΥ	**ΟΣΟ**	**ΓΙΑ**
I ← 20	**I ← 20**	
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ	ΟΣΟ **I >= 10** ΕΠΑΝΑΛΑΒΕ	ΓΙΑ I ΑΠΟ **20** ΜΕΧΡΙ **10** ΜΕ ΒΗΜΑ **-1**
ΓΡΑΨΕ I	ΓΡΑΨΕ I	ΓΡΑΨΕ I
I ← I - 1	**I ← I - 1**	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΜΕΧΡΙΣ_ΟΤΟΥ I < 10	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	

Για να μετατραπεί μία ΜΕΧΡΙΣ ΟΤΟΥ ή ΟΣΟ σε ΓΙΑ πρέπει να είναι γνωστά 3 πράγματα:

1. **αρχική τιμή μετρητή έξω από την επανάληψη**
2. **τελική τιμή (ττ) του μετρητή (μτ)** (στην ΟΣΟ συνθήκη **μτ<=ττ** για βήμα θετικό, **μτ>=ττ** για βήμα αρνητικό ενώ στην ΜΕΧΡΙΣ_ΟΤΟΥ **μτ>ττ** για βήμα θετικό, **μτ <ττ** για βήμα αρνητικό, ενώ σε περίπτωση που δεν είναι αλλάζουμε τον αριθμό της σύγκρισης, ώστε να αλλάξει και ο τελεστής σύγκρισης και να γίνει όπως χρειάζεται στη ΓΙΑ)
3. **και το βήμα (σταθερός αριθμός ή μεταβλητή που δεν μεταβάλλεται μέσα στην επανάληψη) τελευταία εντολή μέσα στην επανάληψη** (διαφορετικά ελέγχουμε αν μπορούμε το μεταφέρουμε σαν τελευταία εντολή κάνοντας τις απαραίτητες τροποποιήσεις στις επόμενες εντολές)

ΤΥΠΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ

1. Αντιμετάθεση Μεταβλητών

Temp ←x
x ← ψ
ψ ← Temp

2. Αν το πρόβλημα μας λέει να ελέγχουμε αν ο αριθμός είναι άρτιος ή περιττός, ή αν διαιρείται ακριβώς με κάποιο αριθμό, πρέπει να χρησιμοποιήσουμε την MOD στη συνθήκη του ελέγχου, πχ η X MOD 3 = 0 ελέγχει αν το X είναι πολλαπλάσιο του 3.

   Πχ να διαβάζουμε έναν αριθμό και να εμφανίζουμε στην οθόνη 'άρτιος' ή 'περιττός'

```
ΔΙΑΒΑΣΕ Χ
ΑΝ X MOD 2 = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'ΑΡΤΙΟΣ'
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΠΕΡΙΤΤΟΣ'
ΤΕΛΟΣ_ΑΝ
```

3. Αν είναι **<u>γνωστός ο αριθμός των επαναλήψεων</u>** μίας ομάδας εντολών χρησιμοποιούμε την εντολή ΓΙΑ. Πχ πρόγραμμα που θα διαβάζει 30 αριθμούς.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
   ΔΙΑΒΑΣΕ Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ! Ι =31    και το Χ θα έχει την τελευταία που έδωσε (την 30η τιμή)
```

4. **<u>Υπολογισμός Αθροίσματος σε επανάληψη</u>**

Αν θέλουμε ***σε μία επανάληψη να βρίσκουμε ένα άθροισμα***, πρέπει πριν την επανάληψη να μηδενίσουμε τον αθροιστή, ώστε την πρώτη φορά που θα μπει στην επανάληψη και θα κάνει στην πράξη να έχει ο αθροιστής την τιμή 0 και μέσα στην επανάληψη να προσθέτουμε τους αριθμούς. Πχ πρόγραμμα που θα διαβάζει 30 αριθμούς και θα μας εμφανίζει στην οθόνη το άθροισμα τους.

```
Σ ← 0                  ! αρχικοποιώ πριν την επανάληψη
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΔΙΑΒΑΣΕ Χ
        Σ ← Σ + Χ      ! υπολογίζω μέσα στην επανάληψη
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Σ                ! εμφανίζω έξω από την επανάληψη 1 φορά
```

Υπολογισμός πλήθος (μετρητή) μέσα σε επανάληψη

```
πληθ ← 0                        !μηδενισμός μετρητή πριν την επανάληψη
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΔΙΑΒΑΣΕ Χ
        ΑΝ Χ > 0 ΤΟΤΕ
            πληθ ← πληθ +1      ! αύξηση μετρητή κατά 1 για κάθε θετικό αριθμό μέσα στην επανάληψη
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ πληθ                      ! εμφανίζει το πλήθος των θετικών αριθμών που δώσαμε στο τέλος 1 φορά
```

Υπολογισμός Γινόμενου σε επανάληψη

Γ ← 1 ! στο γινόμενο βάζουμε αρχική τιμή 1
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
Γ ← Γ * συντ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Αν θέλουμε να υπολογίσουμε δύναμη $α^ν$ βάζουμε Γ ← Γ * α, αν θέλουμε να βρούμε το ν παραγοντικό βάζουμε Γ ← Γ * Ι.

5. **ΔΙΑΒΑΖΟΥΜΕ ΣΥΝΕΧΩΣ ΜΕΧΡΙ ΝΑ ΔΩΣΟΥΜΕ ΜΙΑ ΣΩΣΤΗ ΤΙΜΗ (ΕΛΕΓΧΟΣ ΟΡΘΗΣ ΕΚΧΩΡΗΣΗΣ Ή ΕΛΕΓΧΟΣ ΕΓΚΥΡΟΤΗΤΑΣ)**. Αν σε κάποιο πρόβλημα μας ζητηθεί να διαβάζουμε ***μία τιμή*** και να ελέγχουμε, ώστε να βρίσκεται ***μέσα σε κάποια όρια,*** μπορούμε να το πετύχουμε με τις δύο επαναλήψεις ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ. Για παράδειγμα να διαβάζουμε μία μεταβλητή που να έχει τιμή 1-20.

α) Με την ΜΕΧΡΙΣ_ΟΤΟΥ, που είναι καλύτερος τρόπος, αφού διαβάζουμε μόνο μέσα στην επανάληψη και στη συνθήκη γράφουμε αυτό που μας ζητάνε πχ 1-20

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΔΙΑΒΑΣΕ Χ          !διαβάζουμε μέχρι να δώσουμε τιμή μέσα στα όρια
ΜΕΧΡΙΣ_ΟΤΟΥ Χ>=1 ΚΑΙ Χ <= 20
!επεξεργασία του Χ έξω από την επανάληψη του διαβάσματος
```

β) Με την ΟΣΟ διαβάζουμε 1η φορά απ' έξω και τις υπόλοιπες μέσα στην επανάληψη

```
ΔΙΑΒΑΣΕ Χ          !διαβάζουμε την 1η τιμή
ΟΣΟ Χ < 1 Ή Χ > 20 ΕΠΑΝΑΛΑΒΕ
    ΔΙΑΒΑΣΕ Χ     !διαβάζουμε τη 2η, 3η, 4η, … μέχρι να δώσουμε μία σωστή
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
!επεξεργασία του Χ έξω από την επανάληψη του διαβάσματος
```

6. Αν μας ζητηθεί να διαβάζουμε 30 θετικούς μονοψήφιους αριθμούς (1 - 9) θα πρέπει μέσα στην επανάληψη ΓΙΑ που θα διαβάζουμε τους 30 αριθμούς να βάλουμε και μία ΜΕΧΡΙΣ_ΟΤΟΥ για το διάβασμα του αριθμού, ώστε να είναι 1-9.

Χωρίς έλεγχο για 30 αριθμούς

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
    ΔΙΑΒΑΣΕ Χ
    !Επεξεργασία Χ…..
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Με έλεγχο για Χ μεταξύ 1-9

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ Χ
    ΜΕΧΡΙΣ_ΟΤΟΥ Χ>=1 ΚΑΙ Χ<=9
    !Επεξεργασία Χ…..
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

ΠΡΟΣΟΧΗ ΛΑΘΟΣ

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΔΙΑΒΑΣΕ Χ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ>=1 ΚΑΙ Χ<=9
```

ΔΙΑΒΑΖΟΥΜΕ ΣΥΝΕΧΩΣ ΤΙΜΕΣ ΜΕΧΡΙ ΝΑ ΔΩΣΟΥΜΕ ΜΙΑ ΛΑΘΟΣ ΤΙΜΗ. Αν μας ζητηθεί πρόγραμμα που ***θα διαβάζει τιμές μέχρι να δώσουμε κάποια λάθος τιμή,*** όπου θα επεξεργάζεται και τους προηγούμενους αριθμούς που δίνουμε (εκτός της τιμής εξόδου) μπορούμε να το πετύχουμε με 2 τρόπους:

α) με την ΟΣΟ. Πρέπει να διαβάζουμε την πρώτη τιμή έξω από την επανάληψη και τις επόμενες μέσα στην επανάληψη σαν τελευταία εντολή στην ομάδα εντολών. Αν θέλουμε να υπολογίσουμε τον Μ.Ο. θετικών, πρέπει μέσα στην επανάληψη να βρίσκουμε το άθροισμα και το πλήθος των αριθμών και μετά την επανάληψη να κάνουμε την διαίρεση για τον υπολογισμό του Μ.Ο.. Δεν πρέπει να ξεχνάμε να μηδενίζουμε τον μετρητή των αριθμών και τον αθροιστή έξω από την επανάληψη.

Πχ να διαβάζει και να προσθέτει αριθμούς μέχρι να δώσουμε μία αρνητική τιμή και να εμφανίζει στο τέλος το Μ.Ο των μη αρνητικών αριθμών.

```
Σ ← 0
ΠΛΗΘΟΣ ← 0
ΔΙΑΒΑΣΕ Χ                    !διαβάζουμε την 1η τιμή
ΟΣΟ Χ >= 0 ΕΠΑΝΑΛΑΒΕ
        ΠΛΗΘΟΣ ← ΠΛΗΘΟΣ + 1
        Σ ← Σ + Χ
        ΔΙΑΒΑΣΕ Χ            !διαβάζουμε τη 2η, 3η, 4η, … μέχρι να δώσουμε μία λάθος
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ ΠΛΗΘΟΣ >0 ΤΟΤΕ     ! Πριν από διαίρεση πρέπει να ελέγχουμε αν ο παρονομαστής είναι < > 0.
        ΜΟ ← Σ / ΠΛΗΘΟΣ
        ΓΡΑΨΕ ΜΟ
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΔΕΝ ΕΔΩΣΕΣ ΚΑΝΕΝΑ ΘΕΤΙΚΟ ΑΡΙΘΜΟ'
ΤΕΛΟΣ_ΑΝ
```

Η ΟΣΟ μας βολεύει καλύτερα σε τέτοια προβλήματα, γιατί για το αν θα εκτελέσουμε τις υπόλοιπες εντολές εκτός από την ΔΙΑΒΑΣΕ έχει γίνει ο έλεγχος στην συνθήκη της επανάληψης, ενώ στην ΜΕΧΡΙΣ_ΟΤΟΥ πρέπει να γίνει και άλλος έλεγχος ΑΝ μέσα, για τον αν η τιμή που διαβάσαμε είναι μέσα στις επιθυμητές.

Α) με την ΜΕΧΡΙΣ_ΟΤΟΥ. Εδώ ***δεν μπορούμε να διαβάζουμε*** την πρώτη τιμή έξω από την επανάληψη και τις επόμενες μέσα στην επανάληψη σαν τελευταία εντολή στην ομάδα εντολών. Πρέπει να ***διαβάζουμε μέσα στην επανάληψη σαν πρώτη εντολή*** και στη συνέχεια, αφού ***κάνουμε έλεγχο*** (αντίθετη συνθήκη με της ΜΕΧΡΙΣ_ΟΤΟΥ) αν η τιμή της είναι αποδεκτή, εκτελούμε τις υπόλοιπες εντολές της ομάδας εντολών της επανάληψης. Αν θέλουμε να λύσουμε το ίδιο πρόβλημα με το παραπάνω της ΟΣΟ ο κώδικας είναι ο παρακάτω δεξιά:

```
Σ ← 0
ΠΛΗΘΟΣ ← 0
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ Χ  ! μέχρι να δώσουμε μία λάθος τιμή
        ΑΝ Χ >= 0 ΤΟΤΕ
                ΠΛΗΘΟΣ ←ΠΛΗΘΟΣ +1
                Σ ← Σ + Χ
        ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ<0
ΑΝ ΠΛΗΘΟΣ >0 ΤΟΤΕ
        ΜΟ ← Σ / ΠΛΗΘΟΣ
        ΓΡΑΨΕ ΜΟ
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ ΑΡΙΘΜΟ'
ΤΕΛΟΣ_ΑΝ
```

2ος τρόπος (μετατροπή της ΟΣΟ της προηγούμενης σελίδας σε ΜΕΧΡΙΣ_ΟΤΟΥ)

```
Σ ← 0
ΠΛΗΘΟΣ ← 0
ΔΙΑΒΑΣΕ Χ
ΑΝ Χ >= 0 ΤΟΤΕ
        ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
                ΠΛΗΘΟΣ ← ΠΛΗΘΟΣ + 1
                Σ ← Σ + Χ
                ΔΙΑΒΑΣΕ Χ
        ΜΕΧΡΙΣ_ΟΤΟΥ Χ<0
ΤΕΛΟΣ_ΑΝ
….
```

7. **Αν μας ζητηθεί να διαβάζουμε τιμές μέχρι να συμβεί κάτι που δεν έχει σχέση με την μεταβλητή που χρησιμοποιούμε για το διάβασμα των τιμών, τότε διαβάζουμε μόνο μέσα στην επανάληψη τις τιμές**, αφού η συνθήκη στην ΟΣΟ δεν ελέγχει την μεταβλητή που διαβάζουμε, αλλά κάτι άλλο. Πχ να διαβάζουμε αριθμούς μέχρι να δώσουμε 10 θετικούς αριθμούς.

(δεν ελέγχω αυτό που διαβάζω (πχ το Χ), αλλά κάτι άλλο)

```
ΠΘ ← 0
ΟΣΟ ΠΘ < 10 ΕΠΑΝΑΛΑΒΕ
   ΔΙΑΒΑΣΕ Χ
   ΑΝ Χ > 0 ΤΟΤΕ
         ΠΘ ← ΠΘ + 1
   ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΠΘ ← 0
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ Χ
      ΑΝ Χ > 0 ΤΟΤΕ
            ΠΘ ← ΠΘ + 1
      ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ ΠΘ = 10
```

8. Αν μας ζητηθεί να διαβάζουμε το πολύ ένα πλήθος αριθμών (έστω 10) εκτός και δώσουμε νωρίτερα μία τιμή (πχ το 100), γίνεται πιο εύκολα με την **ΜΕΧΡΙΣ_ΟΤΟΥ**.

(ελέγχω αυτό που διαβάζω (πχ το Χ) + κάτι ακόμη)

```
ΠΛ ← 0
Σ ← 0
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ Χ   ! επεξεργαζόμαστε και το 100, αν δεν θέλαμε θα βάζαμε το Σ ← Σ + Χ
      Σ ← Σ + Χ   ! μέσα σε μία ΑΝ Χ < > 100 ΤΟΤΕ … ΤΕΛΟΣ_ΑΝ
      ΠΛ ← ΠΛ + 1
ΜΕΧΡΙΣ_ΟΤΟΥ ΠΛ = 10 Ή Χ = 100
```

9. Αν μας ζητηθεί σε ένα πρόβλημα ***με επανάληψη*** να βρούμε πχ ***τον μέγιστο*** από κάποιους αριθμούς, πρέπει να ορίσουμε αρχική τιμή έξω από την επανάληψη. Ο καλύτερος τρόπος είναι να διαβάσουμε την πρώτη τιμή έξω από την επανάληψη και να την ορίσουμε σαν την μεγαλύτερη και στη συνέχεια μέσα στην επανάληψη να διαβάζουμε τις υπόλοιπες τιμές και να γίνεται ο έλεγχος αν είναι μεγαλύτερη από τη μέγιστη που έχουμε βρει μέχρι εκείνη τη στιγμή.

Μέγιστο – ελάχιστο για γνωστό πλήθος επαναλήψεων

Να εισάγουμε 30 αριθμούς και να βρούμε τον ελάχιστο και τον μέγιστο. Στο 1ο τρόπο βάζουμε έναν πολύ μεγάλο θετικό στο ΜΙΝ και έναν πολύ μεγάλο αρνητικό αριθμό στο ΜΑΧ, ώστε με τον πρώτο αριθμό που θα δώσω να αλλάξει και ο ΜΙΝ και ο ΜΑΧ και να μπει σαν τιμή ο πρώτος αριθμός που θα διαβάσουμε. Στο δεύτερο διαβάζουμε τον πρώτο αριθμό και τον θέτουμε σαν ΜΙΝ και ΜΑΧ και διαβάζουμε τους υπόλοιπους.

(α' τρόπος)	**(β' τρόπος)**	(Γ' τρόπος)
ΜΑΧ ← -10^18	ΔΙΑΒΑΣΕ Χ !1η τιμή	ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
ΜΙΝ ← 10^18	**ΜΑΧ ← Χ**	ΔΙΑΒΑΣΕ Χ
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30	**ΜΙΝ ← Χ**	ΑΝ Ι = 1 ΤΟΤΕ
ΔΙΑΒΑΣΕ Χ	ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 30	**ΜΑΧ ← Χ**
ΑΝ Χ > ΜΑΧ ΤΟΤΕ	ΔΙΑΒΑΣΕ Χ	**ΜΙΝ ← Χ**
ΜΑΧ ← Χ	ΑΝ Χ > ΜΑΧ ΤΟΤΕ	ΑΛΛΙΩΣ
ΤΕΛΟΣ_ΑΝ	**ΜΑΧ ← Χ**	ΑΝ Χ > ΜΑΧ ΤΟΤΕ
ΑΝ Χ < ΜΙΝ ΤΟΤΕ	ΤΕΛΟΣ_ΑΝ	**ΜΑΧ ← Χ**
ΜΙΝ ← Χ	ΑΝ Χ < ΜΙΝ ΤΟΤΕ	ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ	**ΜΙΝ ← Χ**	ΑΝ Χ < ΜΙΝ ΤΟΤΕ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	ΤΕΛΟΣ_ΑΝ	**ΜΙΝ ← Χ**
ΓΡΑΨΕ ΜΑΧ, ΜΙΝ	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	ΤΕΛΟΣ_ΑΝ
	ΓΡΑΨΕ ΜΑΧ, ΜΙΝ	ΤΕΛΟΣ_ΑΝ
		ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
		ΓΡΑΨΕ ΜΑΧ, ΜΙΝ

Στο 1ο τρόπο οι έλεγχοι για τον ΜΙΝ και ΜΑΧ γίνονται υποχρεωτικά με **απλές ΑΝ,** ώστε ο πρώτος αριθμός να μπει και στον ΜΙΝ και στον ΜΑΧ, ενώ αν τον κάναμε τον έλεγχο με σύνθετη ΑΝ, θα άλλαζε μόνο ο ΜΑΧ για τον πρώτο αριθμό, ενώ τον δεύτερο τρόπο μπορούμε να τον κάνουμε και με σύνθετη ΑΝ.

Αν μας έλεγε η άσκηση να βρούμε το μεγαλύτερο και το μικρότερο **βαθμό ενός μαθητή** θα ορίζαμε σαν αρχική τιμή στο **ΜΑΧ ← -1 και στο ΜΙΝ ← 21**, ώστε με τον πρώτο βαθμό να αλλάξουν και να μπει μία σωστή βαθμολογία. Αν ορίζαμε σαν αρχικές στο ~~**ΜΑΧ←20**~~ **και στο** ~~**ΜΙΝ←0**~~ **είναι φυσικά λάθος**, αφού δεν θα άλλαζαν μέσα στην επανάληψη το ΜΑΧ και το ΜΙΝ και θα μας εμφάνιζε τις αρχικές τιμές και στο τέλος.

Μέγιστο – ελάχιστο για άγνωστο πλήθος επαναλήψεων

Καλύτερος τρόπος είναι με την ΟΣΟ (β΄ τρόπος), ενώ με την ΜΕΧΡΙΣ-ΟΤΟΥ (β' τρόπος) πρέπει να γίνει έλεγχος μήπως με την πρώτη τιμή θέλουμε να σταματήσουμε την επανάληψη.

Ας δούμε για παράδειγμα ένα πρόγραμμα που θα διαβάζει αριθμούς μέχρι να δώσουμε έναν αρνητικό και στο τέλος να μας εμφανίζει τον μέγιστο θετικό.

```
(α' τρόπος ΟΣΟ ευκολότερος)
MAX ← -10 ^ 18
ΔΙΑΒΑΣΕ Χ
ΟΣΟ Χ >= 0 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Χ > MAX ΤΟΤΕ
        MAX ← Χ
    ΤΕΛΟΣ_ΑΝ
    ΔΙΑΒΑΣΕ Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ MAX < > -10 ^ 18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

```
(β' τρόπος ΟΣΟ)
ΔΙΑΒΑΣΕ Χ
ΑΝ Χ >= 0 ΤΟΤΕ
    MAX ← Χ
    ΟΣΟ Χ >= 0 ΕΠΑΝΑΛΑΒΕ
        ΑΝ Χ > MAX ΤΟΤΕ
            MAX ← Χ
        ΤΕΛΟΣ_ΑΝ
        ΔΙΑΒΑΣΕ Χ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

```
(γ' τρόπος ΟΣΟ)
ΔΙΑΒΑΣΕ Χ
ΠΛ ← 0
ΟΣΟ Χ >= 0 ΕΠΑΝΑΛΑΒΕ
    ΠΛ ← ΠΛ + 1
    ΑΝ ΠΛ=1 ΤΟΤΕ
        MAX ← Χ
    ΑΛΛΙΩΣ
        ΑΝ Χ > MAX ΤΟΤΕ
                MAX ← Χ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
    ΔΙΑΒΑΣΕ Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ ΠΛ > 0 ΤΟΤΕ
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

```
(α' τρόπος ΜΕΧΡΙΣ_ΟΤΟΥ ευκολότερος)
MAX ← -10 ^ 18
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΔΙΑΒΑΣΕ Χ
    ΑΝ Χ >= 0 ΤΟΤΕ
        ΑΝ Χ > MAX ΤΟΤΕ
            MAX ← Χ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ < 0
ΑΝ MAX < > -10 ^ 18 ΤΟΤΕ
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

```
(β' τρόπος ΜΕΧΡΙΣ_ΟΤΟΥ)
ΔΙΑΒΑΣΕ Χ
ΑΝ Χ >= 0 ΤΟΤΕ
    MAX ← Χ
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ Χ > MAX ΤΟΤΕ
            MAX ← Χ
        ΤΕΛΟΣ_ΑΝ
        ΔΙΑΒΑΣΕ Χ
    ΜΕΧΡΙΣ_ΟΤΟΥ Χ < 0
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

```
(Γ' τρόπος ΜΕΧΡΙΣ_ΟΤΟΥ)
ΠΛ ← 0
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΔΙΑΒΑΣΕ Χ
    ΑΝ Χ >= 0 ΤΟΤΕ
        ΠΛ ← ΠΛ + 1
        ΑΝ ΠΛ = 1 ΤΟΤΕ
            MAX ← Χ
        ΑΛΛΙΩΣ
            ΑΝ Χ > MAX ΤΟΤΕ
                MAX ← Χ
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ Χ < 0
ΑΝ ΠΛ >0 ΤΟΤΕ
    ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ:', MAX
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΚΑΝΕΝΑ ΘΕΤΙΚΟ'
ΤΕΛΟΣ_ΑΝ
```

Ο έλεγχος πριν από την επανάληψη στην ΜΕΧΡΙΣ_ΟΤΟΥ β΄ τρόπος γίνεται γιατί αν δεν τον κάνουμε και δώσουμε αρνητική πρώτη τιμή θα μας ζητήσει και δεύτερη τιμή, ενώ θα έπρεπε να σταματάει, και να μην μας εμφανίζει MAX αν δώσουμε μόνο μία αρνητική.

Στον γ τρόπο χρησιμοποιούμε ένα πλήθος και όταν είναι η πρώτη τιμή που δώσαμε την βάζουμε στον MAX, ενώ για τις επόμενες κάνουμε έλεγχο αν είναι μεγαλύτερες του MAX. Τον MAX τον εμφανίζουμε μόνο στην περίπτωση που δώσουμε σωστές τιμές (ΠΛ > 0).

10. Αν ζητηθεί να βρίσκουμε εκτός από **τον ελάχιστο και τη θέση του** πρέπει εκτός από το ΜΙΝ να ορίσουμε και μία μεταβλητή πχ ΘΕΣΗ που φυλάμε τη θέση του μεγαλύτερου. Πχ αν μας πούνε να διαβάσουμε τις ηλικίες 30 ανθρώπων και να βρούμε ποιος (σειρά εισαγωγής) είναι ο μικρότερος.

```
ΔΙΑΒΑΣΕ ΗΛ
ΜΙΝ ← ΗΛ                    ! ΟΡΙΖΟΥΜΕ ΤΗΝ 1η ΗΛΙΚΙΑ ΠΟΥ ΔΙΑΒΑΣΑΜΕ ΣΑΝ ΜΙΝ
ΘΕΣΗ ← 1                    ! Ο 1ος ΕΧΕΙ ΤΗΝ ΜΙΚΡΟΤΕΡΗ ΗΛΙΚΙΑ
ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 30        ! Ι ΕΙΝΑΙ Η ΣΕΙΡΑ ΕΙΣΑΓΩΓΗΣ ΤΩΝ ΗΛΙΚΙΩΝ ΤΩΝ ΑΝΘΡΩΠΩΝ
    ΔΙΑΒΑΣΕ ΗΛ
    ΑΝ ΗΛ < ΜΙΝ ΤΟΤΕ        ! ΑΝ ΚΑΠΟΙΟΣ ΕΧΕΙ ΜΙΚΡΟΤΕΡΗ ΗΛΙΚΙΑ
        ΜΙΝ ← ΗΛ            ! ΟΡΙΖΟΥΜΕ ΑΥΤΗΝ ΣΑΝ ΜΙΚΡΟΤΕΡΗ
        ΘΕΣΗ ← Ι            ! ΦΥΛΑΜΕ ΤΗ ΣΕΙΡΑ ΕΙΣΑΓΩΓΗΣ ΤΟΥ Ι (2-30) ΑΝΘΡΩΠΟΥ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΘΕΣΗ
```

Αν μας ζητάει το **όνομα του μαθητή με την μικρότερη ηλικία** εκτός το ΜΙΝ χρειαζόμαστε και μία ακόμα μεταβλητή πχ ΟΝΜΙΝ που θα φυλάμε μαζί με την ηλικία του μικρότερο και το όνομα του.

```
ΔΙΑΒΑΣΕ ΗΛ, ΟΝ
ΜΙΝ ← ΗΛ                    ! ΟΡΙΖΟΥΜΕ ΤΗΝ 1η ΗΛΙΚΙΑ ΠΟΥ ΔΙΑΒΑΣΑΜΕ ΣΑΝ ΜΙΝ
ΟΝΜΙΝ ← ΟΝ                  ! Ο 1ος ΕΧΕΙ ΤΗΝ ΜΙΚΡΟΤΕΡΗ ΗΛΙΚΙΑ ΚΑΙ ΦΥΛΑΜΕ ΤΟ ΟΝΟΜΑ ΤΟΥ
ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 30        ! Ι ΕΙΝΑΙ Η ΣΕΙΡΑ ΕΙΣΑΓΩΓΗΣ ΤΩΝ ΗΛΙΚΙΩΝ ΤΩΝ ΑΝΘΡΩΠΩΝ
    ΔΙΑΒΑΣΕ ΗΛ,ΟΝ
    ΑΝ ΗΛ < ΜΙΝ ΤΟΤΕ        ! ΑΝ ΚΑΠΟΙΟΣ ΕΧΕΙ ΜΙΚΡΟΤΕΡΗ ΗΛΙΚΙΑ
        ΜΙΝ ← ΗΛ            ! ΟΡΙΖΟΥΜΕ ΑΥΤΗΝ ΣΑΝ ΜΙΚΡΟΤΕΡΗ
        ΟΝΜΙΝ ← ΟΝ          ! ΦΥΛΑΜΕ ΤΟ ΟΝΟΜΑ ΤΟΥ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΟΝΜΙΝ
```

11. Παράδειγμα πίνακα τιμών και τι θα εμφανιστεί με επαναλήψεις:

```
Λ ← 0
Κ ← 10
Μ ← 20
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ ΒΗΜΑ 3
    Λ ← Λ + Ι
    Κ ← Κ + Λ
    Μ ← Μ + Κ + Λ
    ΓΡΑΨΕ Ι, Κ, Λ, Μ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Ι
```

```
1  11 1 32
4  16  5  53
7  28  12  93
10  50  22  165
13
```

	Λ	Κ	Μ	Ι
	0	10	20	?
1η Επαν	1	11	32	1
2η Επαν	5	16	53	4
3η Επαν	12	28	93	7
4η Επαν	22	50	165	10
				13

ΟΛΙΣΘΗΣΗ είναι μία στοιχειώδη λειτουργία για τον επεξεργαστή που εκτελείται ταχύτατα και μετακινεί τα δυαδικά ψηφία ενός αριθμού. Η ολίσθηση προς τα δεξιά ισοδυναμεί στο δυαδικό σύστημα με την διαίρεση με το 2, ενώ η ολίσθηση προς τα αριστερά με τον πολλαπλασιασμό με το 2

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΑΛΑ ΡΩΣΙΚΑ (υπολογίζει το γινόμενο 2 αριθμών). Αν ο δεύτερος αριθμός είναι μονός, τότε προσθέτουμε τον πρώτο σε ένα άθροισμα και στη συνέχεια πολλαπλασιάζουμε τον πρώτο με το 2 και διαιρούμε (ακέραια διαίρεση) τον δεύτερο με το 2. Η παραπάνω διαδικασία επαναλαμβάνεται ΟΣΟ ο δεύτερος αριθμός είναι μεγαλύτερος του 0. Σαν αποτέλεσμα έχουμε η πράξη του πολλαπλασιασμού να εκτελείται πιο γρήγορα σε σχέση με τον κλασσικό τρόπο.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΣΚΗΣΕΩΝ

1. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει ένα αριθμό.
   ΔΙΑΒΑΣΕ Χ

2. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει 30 αριθμούς
   ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
   ΔΙΑΒΑΣΕ Χ
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

3. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει έναν αριθμό και να γίνεται έλεγχος, ώστε να είναι μεταξύ του -50 και του 50 *(ΕΛΕΓΧΟΣ ΜΕΧΡΙ ΝΑ ΔΩΣΟΥΜΕ ΜΙΑ ΣΩΣΤΗ ΤΙΜΗ)*.
   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
   ΔΙΑΒΑΣΕ Χ
   ΜΕΧΡΙΣ_ΟΤΟΥ Χ>=-50 ΚΑΙ Χ <=50
   !Σωστή τιμή εκτός επανάληψης και εδώ θα γίνει επεξεργασία αυτής της τιμής

4. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει έναν αριθμό και αν είναι μεταξύ του -50 και του 50 να εμφανίζει ‘αποδεκτός’, αλλιώς ‘μη αποδεκτός’.

   ΔΙΑΒΑΣΕ Χ
   ΑΝ Χ>=-50 ΚΑΙ Χ<=50 ΤΟΤΕ
   ΓΡΑΨΕ ‘αποδεκτός’
   ΑΛΛΙΩΣ
   ΓΡΑΨΕ ‘μη αποδεκτός’
   ΤΕΛΟΣ_ΑΝ

5. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει 30 αριθμούς και να γίνεται έλεγχος κατά την εισαγωγή, ώστε να είναι μεταξύ το 0 και του 20

   ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 **!30 φορές**
   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
   ΔΙΑΒΑΣΕ Χ
   ΜΕΧΡΙΣ_ΟΤΟΥ Χ>=0 ΚΑΙ Χ<=20 **! έλεγχος ορθής εκχώρησης**
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

6. Να φτιαχτεί τμήμα αλγόριθμου που θα διαβάζει συνεχώς βαθμούς μέχρι να δώσουμε έναν βαθμό εκτός ορίων και να υπολογίζει …… . Οι σωστοί βαθμοί είναι μεταξύ 0-20 (ΕΛΕΓΧΟΣ ΩΣΤΕ ΝΑ ΣΤΑΜΑΤΑΕΙ ΟΤΑΝ ΔΩΣΟΥΜΕ ΜΙΑ ΛΑΘΟΣ ΤΙΜΗ ΚΑΙ ΝΑ ΕΠΕΞΕΡΓΑΖΕΤΑΙ ΤΙΣ ΣΩΣΤΕΣ ΤΙΜΕΣ).

ΔΙΑΒΑΣΕ Χ ΟΣΟ Χ>=0 ΚΑΙ Χ<=20 ΕΠΑΝΑΛΑΒΕ … ***! Σωστές τιμές μέσα*** … ***! Επεξεργασία εντός*** ΔΙΑΒΑΣΕ Χ ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ***! Λάθος η τιμή του Χ εκτός της επανάληψης***	ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ ΔΙΑΒΑΣΕ Χ ΑΝ Χ>=0 ΚΑΙ Χ<=20 ΤΟΤΕ … ***! Σωστές τιμές μέσα*** … ***! Επεξεργασία εντός*** ΤΕΛΟΣ_ΑΝ ΜΕΧΡΙΣ_ΟΤΟΥ Χ<0 Ή Χ>20 ***! Λάθος η τιμή του Χ εκτός της επανάληψης***

ΚΕΦΑΛΑΙΟ 3ο

Δεδομένα

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

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).

Γλωσσών προγραμματισμού. Οι γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων τύπων (types) μεταβλητών (variables) για να περιγράψουν ένα δεδομένο.

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

ΑΛΓΟΡΙΘΜΟΙ + ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ = ΠΡΟΓΡΑΜΜΑΤΑ

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

Οι **βασικές λειτουργίες**-πράξεις επί των δομών δεδομένων είναι οι ακόλουθες:

1. **Προσπέλαση** που σημαίνει επεξεργασία ή προβολή της δομής
2. **Εισαγωγή** ενός νέου κόμβου μέσα στη δομή με την ίδια μορφή με τους υπόλοιπους κόμβους της δομής
3. **Διαγραφή** ενός κόμβου από μία δομή
4. **Αναζήτηση** όπου εντοπίζουμε κάποιο ή κάποιους κόμβους με συγκεκριμένες ιδιότητες
5. **Ταξινόμηση** κατά αύξουσα ή φθίνουσα σειρά σύμφωνα με κάποιο πεδίο
6. **Αντιγραφή** κόμβων από μία δομή δεδομένων σε μία άλλη
7. **Συγχώνευση** δύο η περισσοτέρων δομών σε μία
8. **Διαχωρισμός** το αντίστροφο της συγχώνευσης

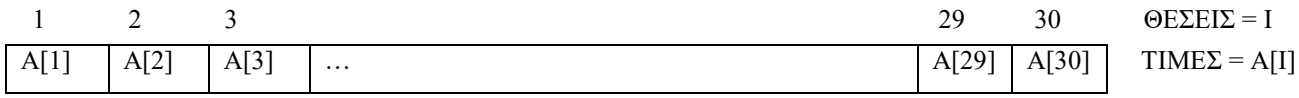
Οι δομές διακρίνονται σε δύο κατηγορίες:

1. **Δυναμικές δομές** που δεν αποθηκεύονται σε συνεχόμενες θέσεις και δεν έχουν σταθερό μέγεθος αφού μπορεί και μεταβάλλεται (προσθέτουμε, διαγράφουμε) κατά την εκτέλεση του προγράμματος. Οι δυναμικές δομές στηρίζονται στην τεχνική της δυναμικής παραχώρησης της μνήμης.
2. **Στατικές δομές (πχ πίνακες)** που αποθηκεύονται σε συνεχόμενες θέσεις και έχουν σταθερό μέγεθος κατά την εκτέλεση του προγράμματος.

ΠΙΝΑΚΕΣ ΜΙΑΣ ΔΙΑΣΤΑΣΗΣ

Πίνακες χρησιμοποιούμε όταν έχουμε μεγάλο πλήθος δεδομένων του ιδίου τύπου τα οποία θέλουμε να επεξεργαστούμε παραπάνω από μία φορά και γνωρίζουμε το μέγεθος του πριν την εκτέλεση.

Πίνακας Α 30 στοιχείων



Οι πίνακες είναι μια στατική δομή δεδομένων όπου το μέγεθος που καταλαμβάνουν ορίζεται κατά την μετάφραση του προγράμματος και το μέγεθος της μνήμης που καταλαμβάνουν είναι σταθερό μέχρι την περάτωση του προγράμματος και βρίσκονται σε συνεχόμενες θέσεις.
Σε κάθε πίνακα πρέπει να δώσουμε ένα όνομα και να ορίσουμε το μέγεθός του. Ο παραπάνω πίνακας έχει το όνομα Α και περιέχει 30 τιμές. Εμείς μπορούμε να επεξεργαζόμαστε μόνο μία τιμή του πίνακα κάθε φορά χρησιμοποιώντας εκτός από το όνομα της και ένα **δείκτη** που μας προσδιορίζει την θέση του πίνακα που θέλουμε να επεξεργαστούμε. Κάτω από ένα όνομα μπορούμε να επεξεργαστούμε πολλές **τιμές του ιδίου τύπου** με πολύ εύκολο τρόπο χρησιμοποιώντας την επανάληψη ΓΙΑ .. ΑΠΟ … ΜΕΧΡΙ και τον **μετρητή** της ταυτόχρονα και **σαν δείκτη** για την θέση του πίνακα.

Αν θέλουμε να εκχωρήσουμε στην θέση 5 του πίνακα την τιμή 123 γράφουμε την εντολή:
Α[5] ← 123

Αν θέλουμε να εμφανίσουμε στην οθόνη την τιμή που έχει το 10 στοιχείο του πίνακα γράφουμε την εντολή:
Γράψε Α[10]

<u>Δήλωση πίνακα</u> με όνομα Α που θα περιέχει 30 ακέραιους αριθμούς

```
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ : Α[30]   ! ΘΕΣΕΙΣ 1-30
```

<u>Διάβασμα των στοιχείων του πίνακα</u> Α που θα περιέχει 30 ακέραιους αριθμούς.

Αν θέλουμε οι 30 τιμές να είναι μεταξύ του -100 και το 100

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΔΙΑΒΑΣΕ Α[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
!ΓΡΑΨΕ Α[Ι]
```

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
                ΔΙΑΒΑΣΕ Α[Ι]
        ΜΕΧΡΙΣ_ΟΤΟΥ Α[Ι] >= -100 ΚΑΙ Α[Ι] <= 100
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εμφάνιση των στοιχείων του πίνακα Α που περιέχει 30 ακέραιους αριθμούς

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΓΡΑΨΕ Α[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός αθροίσματος και ΜΟ των στοιχείων ενός πίνακα Α με 30 ακέραιους αριθμούς

```
Σ ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        Σ ← Σ + Α[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΜΟ ← Σ / 30
ΓΡΑΨΕ Σ, ΜΟ
```

Εύρεση του μέγιστου μέσα σε ένα πίνακα Α με 30 ακέραιους αριθμούς

```
ΜΑΧ ← Α[1]          !αρχικοποίηση μεταβλητής
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ Α[Ι] > ΜΑΧ  ΤΟΤΕ
                ΜΑΧ ← Α[Ι]
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΜΑΧ
```

Εύρεση του μέγιστου και της θέσης του μέσα σε ένα πίνακα Α με 30 ακέραιους αριθμούς

```
ΜΑΧ ← Α[1]          !αρχικοποίηση μεταβλητής
ΘΕΣΗ ← 1
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ Α[Ι] > ΜΑΧ  ΤΟΤΕ
                ΜΑΧ ← Α[Ι]
                ΘΕΣΗ ← Ι
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΜΑΧ, ΘΕΣΗ          ! ΕΠ[ΘΕΣΗ] ΑΝ ΘΕΛΑΜΕ ΤΟ ΕΠΩΝΥΜΟ ΤΟΥ
```

ΠΟΙΟΣ ΕΙΝΑΙ Ο ΜΕΓΑΛΥΤΕΡΟΣ
ΕΝΑΣ ΜΕ ΘΕΣΗ

Σημείωση: σε περίπτωση που υπάρχει πολλές φορές ο ΜΑΧ τότε θα μας εμφανίσει την θέση που τον βρήκε πρώτη φορά. Αν στη σύγκριση αντί για > γράψουμε >= δηλ **ΑΝ Α[Ι] >= ΜΑΧ ΤΟΤΕ** θα μας εμφανίσει την θέση που τον βρήκε τελευταία φορά.

<u>Εύρεση και εμφάνιση όλων των θέσεων όπου υπάρχει η μέγιστη τιμή (ΠΟΙΟΙ – ΠΟΛΛΟΙ)</u>

Για να εμφανίσουμε τις θέσεις του μέγιστου δεν μπορούμε να το κάνουμε πριν ελέγξουμε όλους τους αριθμούς, αφού μπορεί στους επόμενους αριθμούς να υπάρχει μεγαλύτερος. ***Άρα πρώτα βρίσκουμε τον μέγιστο και μετά ελέγχουμε σε ποια στοιχεία του πίνακα υπάρχει η μέγιστη τιμή.***

```
MAX ← A[1]
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ A[I] > MAX  ΤΟΤΕ      ! 1η επανάληψη
                MAX ← A[I]       ! εύρεση του μεγαλύτερου σε αυτή την επανάληψη
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ A[I] = MAX  ΤΟΤΕ      ! 2η επανάληψη
                ΓΡΑΨΕ I          ! Εμφάνιση της θέσης όσων έχουν την μέγιστη τιμή
        ΤΕΛΟΣ_ΑΝ                 ! σε αυτή την επανάληψη
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

<u>Εύρεση αριθμού μέσα σε πίνακα Α με 30 ακέραιους αριθμούς και πόσες φορές (α' τρόπος)</u>

Όταν μας ζητηθεί σε μία άσκηση να δούμε κάτι αν υπάρχει μέσα σε έναν πίνακα ή όχι, το μήνυμα του ότι δεν βρέθηκε δεν το εμφανίζουμε μέσα στην επανάληψη που ελέγχουμε το κάθε στοιχείο του πίνακα, αλλά μετά την επανάληψη. ***Για να το πετύχουμε αυτό χρησιμοποιούμε μία μεταβλητή που της βάζουμε μία αρχική τιμή πριν την επανάληψη και αν βρούμε αυτό που ψάχνουμε μέσα στην επανάληψη της αλλάζουμε τιμή.*** Τέλος μετά την επανάληψη ελέγχουμε αν έχει την αρχική τιμή (δεν βρέθηκε) ή αν άλλαξε.

```
ΔΙΑΒΑΣΕ ΑΡ                  ! ΑΡ είναι ο αριθμός που θα ψάξουμε
ΜΕΤΡ ← 0                    ! τον χρησιμοποιούμε για να δούμε πόσες φορές βρήκαμε τον αριθμό
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ A[I] = ΑΡ ΤΟΤΕ
                ΜΕΤΡ ← ΜΕΤΡ + 1
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30          !ΛΑΘΟΣ ΤΡΟΠΟΣ
        ΑΝ A[I] = ΑΡ ΤΟΤΕ
                ΓΡΑΨΕ 'ΒΡΕΘΗΚΕ'
        ΑΛΛΙΩΣ                  !ΕΜΦΑΝΙΖΕΙ 30 ΦΟΡΕΣ
                ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΑΝ ΜΕΤΡ > 0 ΤΟΤΕ            ! ΜΕΤΑ ΤΟ ΤΕΛΟΣ ΕΛΕΓΧΟΥ ΟΛΟΥ ΤΟΥ ΠΙΝΑΚΑ
        ΓΡΑΨΕ 'ΒΡΕΘΗΚΕ', ΜΕΤΡ, 'ΦΟΡΕΣ'  ! ΕΛΕΓΧΟΥΜΕ ΠΟΣΕΣ ΦΟΡΕΣ ΒΡΕΘΗΚΕ
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

<u>Εύρεση και σε ποια θέση αν βρέθηκε (β' τρόπος)</u>

```
ΔΙΑΒΑΣΕ ΑΡ                ! ΑΡ είναι ο αριθμός που θα ψάξουμε
ΘΕΣΗ ← 0                  ! φυλάμε τη θέση που τη βρήκαμε
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ Α[Ι] = ΑΡ ΤΟΤΕ        ! ΚΑΙ ΘΕΣΗ =0   αν θέλουμε την πρώτη θέση που το βρήκε
                ΘΕΣΗ ← Ι
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΑΝ ΘΕΣΗ > 0 ΤΟΤΕ          ! ΜΕΤΑ ΤΟ ΤΕΛΟΣ ΤΟΥ ΕΛΕΓΧΟΥ ΟΛΟΥ ΤΟΥ ΠΙΝΑΚΑ
        ΓΡΑΨΕ 'ΒΡΕΘΗΚΕ ',ΘΕΣΗ    ! ΕΜΦΑΝΙΖΟΥΜΕ ΤΗΝ ΤΕΛΕΥΤΑΙΑ ΘΕΣΗ
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

<u>Αν θέλει να εμφανίζουμε όλες τις θέσεις που υπάρχει στον πίνακα (γ' τρόπος)</u>

```
ΔΙΑΒΑΣΕ ΑΡ                ! ΑΡ είναι ο αριθμός που θα ψάξουμε
ΜΕΤΡ ← 0                  ! τον χρησιμοποιούμε για να δούμε αν βρήκαμε τον αριθμό
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ Α[Ι] = ΑΡ ΤΟΤΕ
                ΜΕΤΡ ←  ΜΕΤΡ + 1
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΑΝ ΜΕΤΡ > 0 ΤΟΤΕ          ! ΜΕΤΑ ΤΟ ΤΕΛΟΣ ΕΛΕΓΧΟΥ ΟΛΟΥ ΤΟΥ ΠΙΝΑΚΑ
        ΓΡΑΨΕ 'ΒΡΕΘΗΚΕ'       ! ΕΛΕΓΧΟΥΜΕ ΣΕ ΠΟΙΕΣ ΘΕΣΕΙΣ ΒΡΕΘΗΚΕ
        ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
                ΑΝ Α[Ι] = ΑΡ ΤΟΤΕ
                        ΓΡΑΨΕ Ι
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

```
Μ ← 0                              !Β ΤΡΟΠΟΣ
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
    ΑΝ Α[Ι] = ΑΡ ΤΟΤΕ
        ΓΡΑΨΕ Ι          !ΕΜΦΑΝΙΖΟΥΜΕ ΜΕΣΑ
        Μ ← Μ+1          ! ΚΑΙ ΤΑΥΤΟΧΡΟΝΑ ΜΕΤΡΑΜΕ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ Μ = 0 ΤΟΤΕ              !ΕΛΕΓΧΟΥΜΕ ΑΝ ΒΡΕΘΗΚΕ
    ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

<u>ΔΙΑΒΑΣΜΑ ΠΙΝΑΚΑ ΩΣΠΟΥ ΝΑ ΓΕΜΙΣΕΙ (20 ΣΤΟΙΧΕΙΑ) Ή ΝΑ ΔΩΣΟΥΜΕ ΜΙΑ ΤΙΜΗ (-1)</u>

```
ΠΛ ← 0                     !ΜΕΤΡΗΤΗΣ ΘΕΣΗΣ
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ Χ
        ΑΝ Χ < > -1 ΤΟΤΕ
                ΠΛ ← ΠΛ + 1        !ΤΟΝ ΑΥΞΑΝΟΥΜΕ ΣΕ ΚΑΘΕ ΕΠΑΝΑΛΗΨΗ
                ΠΙΝ[ΠΛ] ← Χ
        ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ ΠΛ=20 Ή Χ = -1
```

ΠΙΝΑΚΕΣ ΔΥΟ ΔΙΑΣΤΑΣΕΩΝ

Πίνακας **B**	**1**	**2**	**3**	**4**	**5**
1	B[1, 1]				B[1, 5]
2					
3					
4					
5			B[5, 3]		
6					
7					
8					
9					
10	B[10, 1]				B[10, 5]

ΓΡΑΜΜΕΣ

ΣΤΗΛΕΣ

Δείκτης Ι για τις γραμμές (οριζόντια) από 1-10

Δείκτης Κ για τις στήλες (κάθετα) από 1-5

Όλα τα παρακάτω παραδείγματα θα αναφέρονται σε ένα πίνακα Β που έχει ***10 γραμμές και 5 στήλες***. Για να ορίσουμε ένα στοιχείο αυτού το πίνακα χρειαζόμαστε 2 δείκτες, έναν για την γραμμή και έναν για την στήλη. Πχ Β[5, 3] είναι το στοιχείο της 5ης γραμμής και της 3ης στήλης.

Για να επεξεργαστούμε τα στοιχεία ενός πίνακα δύο διαστάσεων χρησιμοποιούμε δύο εμφωλευμένες επαναλήψεις (μία για τις γραμμές και μία για τις στήλες), ώστε να δημιουργήσουμε τους δύο δείκτες που χρειαζόμαστε, αφού ο πίνακας είναι δύο διαστάσεων.

Δήλωση πίνακα με όνομα Β που θα περιέχει 50 ακέραιους αριθμούς 10Χ5

```
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Β[10, 5]  ! 10 ΓΡΑΜΜΕΣ (1-10),  5 ΣΤΗΛΕΣ (1-5)
```

Διάβασμα των στοιχείων του πίνακα Β (γεμίζοντας κατά γραμμή)

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10          ! ΕΞΩΤΕΡΙΚΗ ΕΠΑΝΑΛΗΨΗ Η ΓΡΑΜΜΗ
   ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5       ! ΕΣΩΤΕΡΙΚΗ ΕΠΑΝΑΛΗΨΗ Η ΣΤΗΛΗ
      ΔΙΑΒΑΣΕ Β[Ι, Κ]
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
I = 1    K=1
         K=2
   ….    K=5
I = 2    K=1
         K=2
         …..
```

Πίνακας Β
10 γραμμών (μετρητής Ι)
5 στηλών (μετρητής Κ)

Διάβασμα των στοιχείων του πίνακα Β (γεμίζοντας κατά στήλη)

```
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5        ! εξωτερική επανάληψη η στήλη
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10  ! εσωτερική επανάληψη η γραμμή
        ΔΙΑΒΑΣΕ Β[Ι, Κ]     ! Δεν αλλάζουμε τους δείκτες του πίνακα για αποφυγή λαθών
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Κ = 1	Ι=1
	Ι=2
….	Ι=10
Κ = 2	Ι=1
	Ι=2
	…..

Εμφάνιση των στοιχείων του πίνακα Β κατά γραμμή

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10       ! εξωτερική επανάληψη η γραμμή
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΓΡΑΨΕ Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός αθροίσματος όλων των στοιχείων του πίνακα Β

```
SUM ← 0                    ! μηδενίζουμε τον αθροιστή έξω από τις δύο επαναλήψεις
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + B[I, K]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ SUM          ! εμφανίζουμε το συνολικό άθροισμα έξω από τις δύο επαναλ.
```

Εύρεση ελάχιστου όλου του πίνακα Β

```
MIN ← B[1, 1]              ! ορίζουμε αρχική τιμή έξω και από τις δύο επαναλήψεις
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΑΝ B[I, K] < MIN ΤΟΤΕ
            MIN ← B[I, K]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ MIN          ! εμφανίζουμε μετά τις δύο επαναλήψεις τον ελάχιστο
```

Εύρεση ελάχιστου σε όλο τον πίνακα καθώς και της θέσης του στον πίνακα Β

```
ΜΙΝ ← Β[1,1]
ΓΡΑΜΜΗ ← 1
ΣΤΗΛΗ ← 1
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΑΝ Β[Ι, Κ] < ΜΙΝ ΤΟΤΕ
            ΜΙΝ ← Β[Ι, Κ]
            ΓΡΑΜΜΗ ← Ι
            ΣΤΗΛΗ ← Κ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΜΙΝ, ΓΡΑΜΜΗ, ΣΤΗΛΗ
```

Σημείωση: σε περίπτωση που υπάρχει πολλές φορές ο Μιν τότε θα μας εμφανίσει την θέση (Γραμμή και Στήλη) που τον βρήκε πρώτη φορά. Αν στη σύγκριση αντί για > γράψουμε >= δηλ **Αν Μιν >= Β[ι,κ] τότε** θα μας εμφανίσει την θέση που τον βρήκε τελευταία φορά.

Εύρεση ελάχιστου κάθε γραμμής του πίνακα καθώς και της θέσης του στον πίνακα Β

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΜΙΝ ← Β[Ι, 1]      ή ΜΙΝ ← 10^18
    ΣΤΗΛΗ ← 1
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΑΝ Β[Ι, Κ] < ΜΙΝ ΤΟΤΕ
            ΜΙΝ ← Β[Ι, Κ]
            ΣΤΗΛΗ ← Κ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ ΜΙΝ, ΣΤΗΛΗ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εύρεση ελάχιστου κάθε στήλης του πίνακα και της θέσης του στον πίνακα Β

```
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5        ! Σε σχέση με το προηγούμενο παράδειγμα αντιστρέφουμε τις
    MIN ← B[1, K]           ! δύο επαναλήψεις και κάνουμε εξωτερική των στηλών
    ΓΡΑΜΜΗ ← 1        ή MIN ← 10^18
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        ΑΝ B[I,K] < MIN ΤΟΤΕ
            MIN ← B[I, K]
            ΓΡΑΜΜΗ ← I
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ MIN, ΓΡΑΜΜΗ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εύρεση όλων των θέσεων με τη μέγιστη τιμή σε κάθε γραμμής του πίνακα Β

Επειδή δεν μπορούμε σε μία επανάληψη να βρούμε την μέγιστή τιμή και ποιοι την έχουν πρέπει να κάνουμε 2 επαναλήψεις. Στην πρώτη βρίσκουμε τον μέγιστο (πρέπει να ελέγξουμε όλο τον πίνακα για να τον βρούμε, αφού μπορεί MAX να είναι και ο τελευταίος αριθμός) και στη δεύτερη επανάληψη εμφανίζουμε ποιες στήλες έχουν αυτή την τιμή.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10          ! για κάθε γραμμή (10 φορές)
    MAX ← B[I, 1]
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΑΝ B[I, K] > MAX ΤΟΤΕ
            MAX ← B[I, K]      ! στην 1η επανάληψη της στήλης βρίσκουμε τον MAX
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ MAX, 'ΚΑΙ ΟΙ ΜΑΘΗΤΕΣ ΠΟΥ ΕΧΟΥΝ ΑΥΤΟ ΤΟ ΒΑΘΜΟ ΕΙΝΑΙ ΟΙ :'
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        ΑΝ MAX = B[I, K] ΤΟΤΕ
            ΓΡΑΨΕ Κ            ! στην 2η επανάληψη εμφανίζουμε τις στήλες με τιμή MAX
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ               ! για την επανάληψη Ι
```

ΜΕΣΑ ΣΤΗΝ ΕΞΩΤΕΡΙΚΗ ΓΙΑ Ι, ΒΑΖΟΥΜΕ ΔΥΟ ΦΟΡΕΣ ΓΙΑ Κ

Υπολογισμός αθροίσματος των στοιχείων της κάθε γραμμής του πίνακα Β

Η ***εξωτερική επανάληψη είναι της γραμμής,*** αφού θέλουμε το άθροισμα κάθε γραμμής και ***η εμφωλευμένη επανάληψη είναι της στήλης,*** ώστε να επεξεργαστούμε για κάθε γραμμή.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    SUM ← 0                 ! τον μηδενίζουμε για κάθε γραμμή μέσα στην επανάλ. της γραμμής
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + B[I, K]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ SUM       ! εμφανίζουμε το άθροισμα κάθε γραμμής
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός αθροίσματος των στοιχείων της κάθε στήλης του πίνακα Β

Τώρα η ***εξωτερική επανάληψη είναι της στήλης*** αφού θέλουμε το άθροισμα κάθε στήλης και ***η εμφωλευμένη επανάληψη είναι της γραμμής.***

```
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    SUM ← 0                 ! τον μηδενίζουμε για κάθε στήλη
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        SUM ← SUM + B[I, K]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ SUM       ! εμφανίζουμε το άθροισμα κάθε στήλης
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός μέγιστου αθροίσματος ανά γραμμή του πίνακα Β

Η ***εξωτερική επανάληψη είναι της γραμμής,*** αφού θέλουμε το άθροισμα κάθε γραμμής και ***η εμφωλευμένη επανάληψη είναι της στήλης,*** ώστε να επεξεργαστούμε για κάθε γραμμή.

```
MAX ← -10^18
THESI ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    SUM ← 0                 ! τον μηδενίζουμε το άθροισμα σε κάθε γραμμή
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + B[I, K]     ! ΚΑΛΥΤΕΡΑ ΜΕ ΤΟΝ ΕΠΟΜΕΝΟ ΤΡΟΠΟ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ SUM       ! εμφανίζουμε το άθροισμα κάθε γραμμής αν μας το ζητάει
```

```
    ΑΝ SUM > MAX ΤΟΤΕ
        MAX ← SUM          ! συγκρίνουμε αν το άθροισμα αυτής της γραμμής είναι
        THESI ← I          ! μεγαλύτερο από το μεγαλύτερο μέχρι τώρα
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ THESI, MAX
```

Υπολογισμός μέγιστου αθροίσματος ανά στήλη του πίνακα Β (πίνακα αθροισμάτων)

Η ***εξωτερική επανάληψη είναι της στήλης,*** αφού θέλουμε το άθροισμα κάθε στήλης και ***η εμφωλευμένη επανάληψη είναι της γραμμής,*** ώστε να επεξεργαστούμε για κάθε στήλη, ο δείκτης των γραμμών να παίρνει όλες τις τιμές (1-10). Σε αυτό το παράδειγμα βρίσκουμε και φυλάμε το άθροισμα κάθε στήλης σε έναν πίνακα SUM και μετά βρίσκουμε το MAX μέσα σε αυτόν τον πίνακα.

```
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    ΑΘΡ ← 0
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        ΑΘΡ ← ΑΘΡ + Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    SUM [K] ← ΑΘΡ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
MAX ← SUM[1]
THESI ← 1
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    ΑΝ SUM[K] > MAX ΤΟΤΕ
        MAX ← SUM[K]       ! συγκρίνουμε αν το άθροισμα αυτής της στήλης είναι
        THESI ← K          ! μεγαλύτερο από το μεγαλύτερο μέχρι τώρα
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ THESI, MAX
```

ΙΣΟΔΥΝΑΜΟ ΜΕ ΤΟ ΔΙΠΛΑ

```
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    SUM [K] ← 0
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        SUM [K] ← SUM [K] + B[I, K]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός του συνολικού αθροίσματος, του αθροίσματος κάθε γραμμής και κάθε στήλης του πίνακα Β. (βιβλίο μαθητή σελ 58)

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΡ[Ι] ← 0          ! φυλάμε το άθροισμα κάθε γραμμής και το μηδενίζουμε αρχικά
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    ΣΤ[Κ] ← 0          ! φυλάμε το άθροισμα κάθε στήλης και το μηδενίζουμε αρχικά
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
SUM ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + Β[Ι, Κ]
        ΓΡ[Ι] ← ΓΡ[Ι] + Β[Ι, Κ]
        ΣΤ[Κ] ← ΣΤ[Κ] + Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ SUM
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΡΑΨΕ 'ΤΟ ΑΘΡΟΙΣΜΑ ΤΗΣ ',Ι,' ΓΡΑΜΜΗΣ ΕΙΝΑΙ ',ΓΡ[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
    ΓΡΑΨΕ 'ΤΟ ΑΘΡΟΙΣΜΑ ΤΗΣ ',Κ,' ΣΤΗΛΗΣ ΕΙΝΑΙ ',ΣΤ[Κ]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Υπολογισμός Μέσου Όρου κάθε γραμμής του πίνακα Β και φύλαξη τους σε ένα πίνακα ΜΟ[10] για περαιτέρω επεξεργασία.

Οι λόγοι που φυλάμε σε πίνακες κάποιες τιμές είναι για να τις επεξεργαστούμε πολλές φορές. Στο παράδειγμα αυτό μετά την εύρεση των ΜΟ βρίσκουμε τον μεγαλύτερο ΜΟ και στη συνέχεια ποιοι είχαν ΜΟ > 18,5

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    SUM ← 0                 ! τον μηδενίζουμε το άθροισμα σε κάθε στήλη
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΟ[Ι] ← SUM / 5          ! φυλάμε το ΜΟ κάθε γραμμής σε ένα πίνακα
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΜΑΧ ← ΜΟ[1]                 ! περαιτέρω επεξεργασία των ΜΟ
ΘΕΣΗΜΑΧ ← 1
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΑΝ ΜΟ[Ι] > ΜΑΧ ΤΟΤΕ      ! εύρεση του μεγαλύτερου ΜΟ και της γραμμής που είναι
        ΜΑΧ ← ΜΟ[Ι]
        ΘΕΣΗΜΑΧ ← Ι
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΓΡΑΨΕ 'Ο ΜΑΘΗΤΗΣ ',ΘΕΣΗΜΑΧ, ' ΕΒΓΑΛΕ ΤΟΝ ΜΕΓΑΛΥΤΕΡΟ ΒΑΘΜΟ', ΜΑΧ

ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10              ! ΕΥΡΕΣΗ ΤΩΝ ΓΡΑΜΜΩΝ ΜΕ ΜΟ > 18,5
    ΑΝ ΜΟ[Ι] > 18,5 ΤΟΤΕ
        ΓΡΑΨΕ 'Ο ΜΑΘΗΤΗΣ ', Ι ,' ΑΡΙΣΤΕΥΣΕ'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εύρεση του μεγαλύτερου ΜΟ κατά γραμμή χωρίς την χρήση πίνακα για την φύλαξη των ΜΟ

Αυτή η άσκηση είναι συνδυασμός των ασκήσεων άθροισμα γραμμής και Μέγιστος 10 αριθμών. Γι' αυτό και ορίζουμε στο ΜΑΧ αρχική τιμή έξω από την πρώτη επανάληψη, ενώ ο έλεγχος για νέο Max γίνεται μέσα στην πρώτη επανάληψη.

```
ΜΑΧ ← -10^20                ! αρχικοποίηση ΜΑΧ και ΘΕΣΗΜΑΧ
ΘΕΣΗΜΑΧ ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    SUM ← 0                 ! τον μηδενίζουμε το άθροισμα σε κάθε στήλη
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        SUM ← SUM + B[I, K]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΟ ← SUM / 5
    ΑΝ ΜΟ > ΜΑΧ ΤΟΤΕ        ! εύρεση του μεγαλύτερου ΜΟ και της γραμμής που είναι
        ΜΑΧ ← ΜΟ
        ΘΕΣΗΜΑΧ ← Ι
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'Ο ΜΑΘΗΤΗΣ ',ΘΕΣΗΜΑΧ, ' ΕΒΓΑΛΕ ΤΟΝ ΜΕΓΑΛΥΤΕΡΟ ΒΑΘΜΟ', ΜΑΧ
```

ΑΝΑΖΗΤΗΣΗ ΣΤΟΙΧΕΙΟΥ ΣΕ ΠΙΝΑΚΑ

Η παρακάτω **αναζήτηση δεν σταματάει όταν το βρει την πρώτη φορά**, αλλά συνεχίζει και ψάχνει μέχρι να τελειώσει ο πίνακας.

```
FOUND ← ΨΕΥΔΗΣ! είναι μία λογική μεταβλητή που φυλάμε αν τον βρήκαμε ή όχι
THESI ← 0              ! αν τον βρούμε φυλάμε την θέση που τον βρήκαμε
ΔΙΑΒΑΣΕ Χ
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 20    ·
    ΑΝ Α[Ι] = Χ ΤΟΤΕ
        FOUND ← ΑΛΗΘΗΣ
        THESI ← I
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ FOUND = ΑΛΗΘΗΣ ΤΟΤΕ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ ', Χ ,' ΒΡΕΘΗΚΕ ΣΤΗ ΘΕΣΗ ', THESI
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

Σημείωση: αν ο αριθμός υπάρχει πολλές φορές μας εμφανίζει την τελευταία θέση που βρέθηκε μέσα στον πίνακα.

Είναι μία εργασία που την συναντάμε πολύ συχνά στους πίνακες. Υπάρχουν πολλών ειδών αναζητήσεις με κυριότερες την **σειριακή (σε μη ταξινομημένους πίνακες)** που είναι πολύ αργή και την **δυαδική (σε ταξινομημένους πίνακες)** που είναι γρηγορότερη.

Σειριακή Αναζήτηση

Τον παραπάνω αλγόριθμο μπορούμε να τον τροποποιήσουμε, ώστε να ***σταματάει την αναζήτηση μόλις τον βρει για πρώτη φορά.***

Την χρησιμοποιούμε όταν ο πίνακας:

1. **έχει μικρό μέγεθος (Ν = 20)**
2. **δεν είναι ταξινομημένος**
3. **δεν κάνουμε συχνά αναζήτηση**

Στην σειριακή αναζήτηση ξεκινάμε και ψάχνουμε αν υπάρχει το στοιχείο που αναζητάμε μέσα στα στοιχεία του πίνακα και μας εμφανίζει αν υπάρχει ή όχι. Στο παράδειγμα που ακολουθεί ψάχνουμε αν υπάρχει ο αριθμός Χ μέσα στα στοιχεία του πίνακα Α που έχει 20 αριθμούς.

```
FOUND ← ΨΕΥΔΗΣ! είναι μία λογική μεταβλητή που φυλάμε αν τον βρήκαμε ή όχι
THESI ← 0            ! αν τον βρούμε φυλάμε την θέση που τον βρήκαμε
ΔΙΑΒΑΣΕ Χ
I ← 1
ΟΣΟ FOUND = ΨΕΥΔΗΣ ΚΑΙ I <=20 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Α[Ι] = Χ ΤΟΤΕ
        FOUND ← ΑΛΗΘΗΣ
        THESI ← Ι
    ΑΛΛΙΩΣ
        I ← I + 1      !ΜΠΟΡΕΙ ΝΑ ΜΠΕΙ ΚΑΙ ΕΞΩ ΑΠΟ ΤΗΝ ΑΝ ΧΩΡΙΣ ΑΛΛΙΩΣ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ FOUND = ΑΛΗΘΗΣ ΤΟΤΕ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ , '  ΒΡΕΘΗΚΕ ΣΤΗ ΘΕΣΗ ', THESI
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ , ' ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

ΤΡΟΠΟΣ ΒΙΒΛΙΟΥ

Στον αλγόριθμο της σειριακής αναζήτησης το βήμα το αυξάνουμε μόνο όταν δεν βρίσκουμε αυτό που ψάχνουμε, ώστε όταν το βρούμε το Ι να έχει τη θέση που το βρήκαμε και όχι την επόμενη. Το βήμα (Ι ← Ι + 1) μπορούμε να μην το βάλουμε στο ΑΛΛΙΩΣ της ΑΝ, αλλά μετά το ΤΕΛΟΣ_ΑΝ (έξω από τον έλεγχο) και να αυξάνει πάντοτε.

Παραλλαγή της σειριακής αναζήτησης ώστε να σταματάει την αναζήτηση όταν το βρει πολλές φορές πχ να σταματάει μόλις βρει 5 φορές αυτό που ψάχνουμε.

```
FOUND ← ΨΕΥΔΗΣ
ΠΛ ← 0
ΔΙΑΒΑΣΕ Χ
I ← 1
ΟΣΟ FOUND = ΨΕΥΔΗΣ ΚΑΙ I <=20 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Α[Ι] = Χ ΤΟΤΕ
        ΠΛ ← ΠΛ + 1
    ΤΕΛΟΣ_ΑΝ
    I ← I + 1 ! ΥΠΟΧΡΕΩΤΙΚΑ ΕΞΩ ΑΠΟ ΤΗΝ ΑΝ
    ΑΝ ΠΛ = 5 ΤΟΤΕ
        FOUND ← ΑΛΗΘΗΣ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΠΛ ← 0
ΔΙΑΒΑΣΕ Χ
I ← 1
ΟΣΟ ΠΛ < 5 ΚΑΙ I <=20 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Α[Ι] = Χ ΤΟΤΕ
        ΠΛ ← ΠΛ + 1
    ΤΕΛΟΣ_ΑΝ
    I ← I + 1 ! ΥΠΟΧΡΕΩΤΙΚΑ ΕΞΩ ΑΠΟ ΤΗΝ ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Το βήμα υποχρεωτικά έξω από την ΑΝ, ώστε όταν βρίσκει αυτό που ψάχνουμε να πηγαίνουμε και στις επόμενες θέσεις, διαφορετικά όταν το έβρισκε το Ι θα παρέμενε ίδιο και θα αύξανε συνεχώς το ΠΛ στην ίδια θέση Ι.

Ο παραπάνω αλγόριθμος μπορεί να γραφτεί και διαφορετικά χωρίς να χρειάζεται η μεταβλητή thesi

```
FOUND ← ΨΕΥΔΗΣ! είναι μία λογική μεταβλητή που φυλάμε αν τον βρήκαμε ή όχι
ΔΙΑΒΑΣΕ Χ
Ι ← 1
ΟΣΟ FOUND = ΨΕΥΔΗΣ ΚΑΙ Ι <=20 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Α[Ι] = Χ ΤΟΤΕ
        FOUND ← ΑΛΗΘΗΣ
    ΑΛΛΙΩΣ
        Ι ← Ι + 1   ! ΤΟ Ι ΑΥΞΑΝΕΙ ΟΤΑΝ ΔΕΝ ΕΙΝΑΙ Ο Χ ΣΤΗ ΘΕΣΗ ΠΟΥ ΔΕΙΧΝΕΙ
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ FOUND = ΑΛΗΘΗΣ ΤΟΤΕ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ ,'  ΒΡΕΘΗΚΕ ΣΤΗ ΘΕΣΗ ', Ι
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ ,' ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

Ο παραπάνω αλγόριθμος μπορεί να γραφτεί και διαφορετικά χωρίς να χρειάζεται η μεταβλητή thesi και Found σε σχέση με το πρώτο παράδειγμα

```
ΔΙΑΒΑΣΕ Χ
Ι ← 0
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        Ι ← Ι + 1
ΜΕΧΡΙΣ_ΟΤΟΥ Α[Ι] = Χ Ή Ι = 20
ΑΝ Α[Ι] = Χ ΤΟΤΕ    ! σημαίνει ότι σταμάτησε σε αυτή τη θέση, άρα εκεί βρίσκεται ο Χ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ ,'  ΒΡΕΘΗΚΕ ΣΤΗ ΘΕΣΗ ', Ι
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Ο ΑΡΙΘΜΟΣ  ', Χ ,' ΔΕΝ ΒΡΕΘΗΚΕ'
ΤΕΛΟΣ_ΑΝ
```

ΤΑΞΙΝΟΜΗΣΗ ΠΙΝΑΚΑ

Ταξινόμηση πίνακα κατά αύξουσα σειρά σημαίνει ότι στο πρώτο στοιχείο του πίνακα πρέπει να υπάρχει το μικρότερο στοιχείο και κάθε επόμενο στοιχείο είναι μεγαλύτερο ή ίσο με το προηγούμενο. Το τελευταίο στοιχείο του πίνακα πρέπει μετά την ταξινόμηση να περιέχει την μεγαλύτερη τιμή. Η ταξινόμηση μας χρησιμεύει στην αναζήτηση στοιχείων με δυαδική αναζήτηση ή άλλες μεθόδους, αφού γίνεται πολύ γρηγορότερα από ότι με την σειριακή αναζήτηση. **Ένας απλός τρόπος ταξινόμησης που είναι επιπλέον αργός.**

Ταξινόμηση ευθείας ανταλλαγής ή Φυσαλίδας

Με αυτόν τον τρόπο θεωρούμε ότι ο πίνακας είναι κατακόρυφος με το πρώτο στοιχείο να βρίσκεται στην κορυφή και το τελευταίο στοιχείο κάτω. Ακόμα τους αριθμούς του πίνακα τους θεωρούμε φυσαλίδες, οπότε ο μεγαλύτερος αριθμός είναι μεγαλύτερη φυσαλίδα (μεγαλύτερο βάρος) με αποτέλεσμα να πηγαίνει προς τα κάτω όταν η προηγούμενη φυσαλίδα είναι μικρότερη και να γίνεται αντιμετάθεση των φυσαλίδων δηλ. των αριθμών μας.

Ας δούμε το κομμάτι κώδικα που ταξινομεί έναν πίνακα με αυτή τη μέθοδο.

```
ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ Ν                ! αριθμός περασμάτων από κάτω προς τα πάνω
  ΓΙΑ Κ ΑΠΟ Ν ΜΕΧΡΙ Ι ΜΕ ΒΗΜΑ –1 ! το στοιχείο που συγκρίνουμε με το προηγούμενο
    ΑΝ Α[Κ] < Α[Κ-1] ΤΟΤΕ          ! ταξινόμηση κατά αύξουσα τιμή
      ΤΕΜΡ ← Α[Κ]                  ! Αυτές είναι οι εντολές αντιμετάθεσης των τιμών
      Α[Κ] ← Α[Κ-1]                ! Α[Κ] και Α[Κ-1] χρησιμοποιώντας την βοηθητική
      Α[Κ-1] ←ΤΕΜΡ                 ! μεταβλητή Temp
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Για παράδειγμα ας ταξινομήσουμε έναν πίνακα με 7 στοιχεία με τις τιμές 9, 1, 8, 6, 5, 2, 7. Η σύγκριση των φυσαλίδων ξεκινάει από κάτω προς τα προς τα πάνω και κάθε φορά ο έλεγχος σταματάει στο προηγούμενο στοιχείο (την πρώτη φορά σταματάει η σύγκριση στο 2ο με το 1ο, την επόμενη φορά σταματάει στο 3ο στοιχείο η σύγκριση με το προηγούμενο κ.ο.κ.). Δηλ την πρώτη φορά (ι = 2) θα συγκρίνουμε το 7ο στοιχείο με το 6ο, … , και τελευταία το 2ο με το 1ο, στην επόμενη επανάληψη (ι=3) το 7ο με το 6ο, …, και τέλος το 3ο με το 2ο, γιατί στην πρώτη θέση ξέρουμε ότι βρίσκεται η ελαφρύτερη φυσαλίδα. Στην 3η επανάληψη θα συγκρίνουμε το 7ο με το 6ο στοιχείο, …, τέλος το 4ο με το 3ο στοιχείο και στην τελευταία επανάληψη (ι = 7) θα συγκρίνουμε μόνο το 7ο με το 6ο στοιχείο.

Στο παρακάτω παράδειγμα φαίνονται οι τιμές που παίρνουν οι μετρητές επαναλήψεων I, K κατά την εκτέλεση του προγράμματος. Με γκρι σκίαση φαίνονται ποιοι αριθμοί συγκρίνονται και γίνεται και η αλλαγή όπου υπάρχει μικρότερος αριθμός σε μεγαλύτερη θέση στον πίνακα, ώστε στην πρώτη επανάληψη (ι=2) να τοποθετήσουμε στην πρώτη θέση του πίνακα τον μικρότερο αριθμό, στην δεύτερη επανάληψη (ι=3) να τοποθετήσουμε τον δεύτερο μικρότερο αριθμό στη δεύτερη θέση κ.ο.κ. μέχρι να φτάσουμε στην τελευταία (έκτη) επανάληψη (ι=7) όπου θα συγκρίνουμε το τελευταίο με το προτελευταίο στοιχείο του πίνακα.

Τιμή I		**I = 2** **1η επανάληψη**						**I = 3** **2η επανάληψη**					**I = 4** **3η επαν.**				**I = 5** **4η επαν.**			**I = 6** **5η επ.**		**I=7** **6η επ.**	
Τιμή K		**7**	**6**	**5**	**4**	**3**	**2**	**7**	**6**	**5**	**4**	**3**	**7**	**6**	**5**	**4**	**7**	**6**	**5**	**7**	**6**	**7**	
Στοιχείο Πίνακα	**Προ Ταξιν.**																						**Μετά Ταξιν.**
1ο	**9**	9	9	9	9	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
2ο	**1**	1	1	1	1	1	9	9	9	9	9	2	2	2	2	2	2	2	2	2	2	2	**2**
3ο	**8**	8	8	8	2	2	2	2	2	2	2	9	9	9	9	5	5	5	5	5	5	5	**5**
4ο	**6**	6	6	2	8	8	8	8	8	5	5	5	5	5	5	9	9	9	6	6	6	6	**6**
5ο	**5**	5	2	6	6	6	6	6	5	8	8	8	8	6	6	6	6	6	9	9	7	7	**7**
6ο	**2**	2	5	5	5	5	5	5	6	6	6	6	6	8	8	8	7	7	7	7	9	8	**8**
7ο	**7**	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	8	9	**9**

Στο 1ο πέρασμα I=2 και το K=7–2, 2ο πέρασμα I=3 και το K=7–3, στο 3ο πέρασμα I=4 και το K=7–4, 4ο πέρασμα I=5 και το K=7–5, 6ο πέρασμα I=6 και το K=7–6, και στο τελευταίο 6ο πέρασμα I=7 και το K=7.
Αν θέλουμε να αριθμήσουμε τις N-1 επαναλήψεις των περασμάτων από κάτω προς τα πάνω ξεκινώντας από 1 και σταματώντας στο N-1 ο αλγόριθμος που θα έδινε το ίδιο αποτέλεσμα με τον παραπάνω είναι ο ακόλουθος

```
ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ N-1                    ! αριθμός περασμάτων από κάτω προς τα πάνω
   ΓΙΑ K ΑΠΟ N ΜΕΧΡΙ I+1 ΜΕ ΒΗΜΑ –1 ! το στοιχείο που συγκρίνουμε με το προηγούμενο
        ΑΝ A[K] < A[K-1] ΤΟΤΕ
           TEMP ← A[K]                  ! Αυτές είναι οι εντολές αντιμετάθεσης των τιμών
           A[K] ← A[K-1]                ! A[K] και A[K-1] χρησιμοποιώντας την βοηθητική
           A[K-1] ←TEMP                 ! μεταβλητή Temp
        ΤΕΛΟΣ_ΑΝ
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Όπου οι μετρητές παίρνουν τις τιμές

I	1	2	3	4	5	6
K	7 – 2	7-3	7 – 4	7 – 5	7 – 6	7

Όπως και πριν δημιουργούμε τις ίδιες συγκρίσεις στοιχείων του πίνακα με το προηγούμενο τους στοιχείο (η μεταβλητή K, που δείχνει ποιο στοιχείο θα συγκριθεί με το προηγούμενο, σε κάθε πέρασμα του I).

ΟΡΙΣΜΟΣ ΤΑΞΙΝΟΜΗΣΗΣ : Δοθέντων των στοιχείων a1, a2, … , an η ταξινόμηση συνίσταται στη μετάθεση της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά ak1, ak2, … , akn έτσι ώστε δοθείσης μίας συνάρτησης διάταξης (ordering function), f, να ισχύει:

f(ak1) <= f(ak2) <= … <= f(akn)

3. 7 ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΔΕΥΤΕΡΕΥΟΥΣΑΣ ΜΝΗΜΗΣ

Στην περίπτωση που έχουμε έναν μεγάλο όγκο δεδομένων (πχ δομή πίνακα) προς επεξεργασία στην κύρια μνήμη, έχουμε πρόβλημα με την κύρια μνήμη (RAM), αφού α) το μέγεθος είναι της περιορισμένο και β) χάνονται τα δεδομένα της όταν κλείσει ο υπολογιστής. Γι' αυτό το λόγο αποθηκεύουμε τα δεδομένα στον υπολογιστή με μορφή αρχείων στην δευτερεύουσα μνήμη του υπολογιστή (συνήθως ο σκληρός δίσκος και τα υπόλοιπα αποθηκευτικά μέσα). Τα στοιχεία ενός αρχείου ονομάζονται εγγραφές (πχ μαθητές) και κάθε εγγραφή αποτελείται από πεδία (αριθμός μητρώου, επώνυμο, όνομα, τηλέφωνο κλπ), που περιγράφουν διαφορετικά στοιχεία μίας εγγραφής. Ο αριθμός μητρώου ταυτοποιεί μία εγγραφή μαθητή (αντιστοιχεί μονοσήμαντα) και λέγεται πρωτεύον κλειδί, ενώ το επώνυμο ταυτοποιεί και αυτό μία εγγραφή μαθητή και λέγεται δευτερεύον κλειδί. Η αναζήτηση μπορεί να γίνει με βάση το πρωτεύον ή το δευτερεύον κλειδί.

ΤΑΞΙΝΟΜΗΣΗ ΣΕ ΔΙΣΔΙΑΣΤΑΤΟ ΠΙΝΑΚΑ

1) **Ταξινόμηση των στοιχείων κάθε γραμμής** ενός δισδιάστατου πίνακα Β 10Χ5 κατά αύξουσα σειρά, όπως φαίνεται στο παρακάτω παράδειγμα

	1				5
1	1	5	3	2	4
	7	6	5	8	3
	..	..	..	..	..
10	3	2	7	9	6

⟶

	1				5
1	1	2	3	4	5
	3	5	6	7	8
	..	..	..	..	..
10	2	3	6	7	9

ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10 ! για κάθε γραμμή

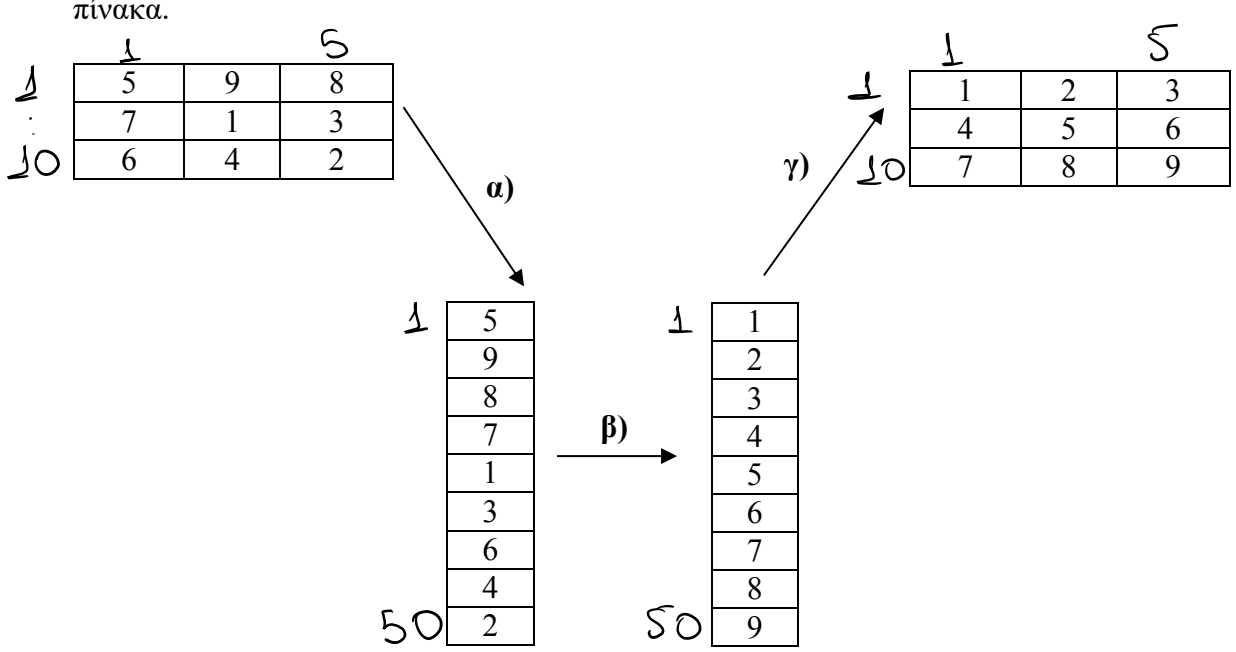
```
ΓΙΑ Λ ΑΠΟ 2 ΜΕΧΡΙ 5          ! και τώρα ταξινομούμε τα 5 στοιχεία της
    ΓΙΑ Κ ΑΠΟ 5 ΜΕΧΡΙ Λ ΜΕ ΒΗΜΑ -1
        ΑΝ Β[Ι, Κ] < Β[Ι, Κ-1] ΤΟΤΕ
            ΒΟΗΘ ← Β[Ι, Κ]
            Β[Ι, Κ] ← Β[Ι, Κ-1]
            Β[Ι, Κ-1] ← ΒΟΗΘ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

10 ταξινομήσεις των 5 στοιχείων

3 ΓΙΑ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

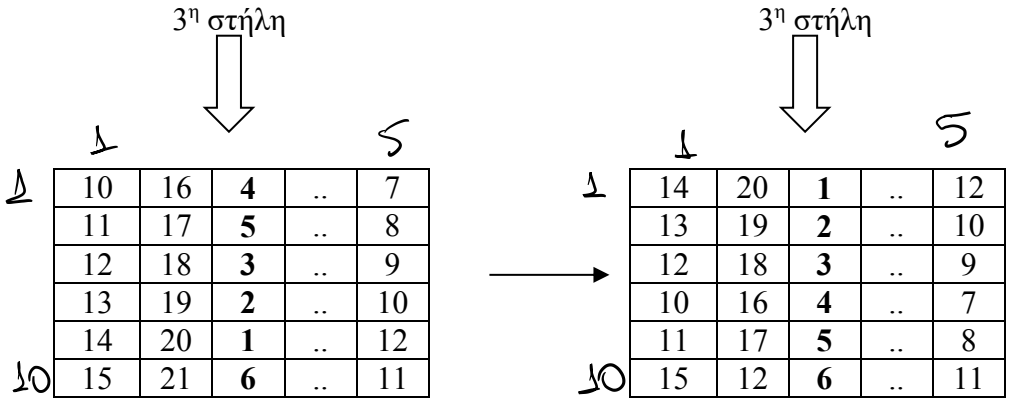
2) **Ταξινόμηση όλων των στοιχείων ενός πίνακα** 10Χ5 κατά αύξουσα σειρά, τοποθετώντας τα κατά γραμμή από πάνω προς τα κάτω σαν το παράδειγμα που ακολουθεί αλλά σε μικρότερο πίνακα.



Επειδή η απ' ευθείας ταξινόμηση ενός δισδιάστατου πίνακα Β είναι δύσκολη α) τοποθετούμε τον δισδιάστατο σε έναν μονοδιάστατο πίνακα Α β) ταξινομούμε τον μονοδιάστατο πίνακα και γ) τον τοποθετούμε πάλι πίσω στον δισδιάστατο.

α)
```
Μ ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        Μ ← Μ + 1
        Α[Μ] ← Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```
β)
```
ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 50
    ΓΙΑ Κ ΑΠΟ 50 ΜΕΧΡΙ Ι ΜΕ ΒΗΜΑ -1
        ΑΝ Α[Κ] < Α[Κ-1] ΤΟΤΕ
            ΒΟΗΘ ← Α[Κ]
            Α[Κ] ← Α[Κ-1]
            Α[Κ-1] ← ΒΟΗΘ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```
γ)
```
Μ ← 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 5
        Μ ← Μ + 1
        Β[Ι, Κ] ← Α[Μ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

3) **Ταξινόμηση όλων των στοιχείων ενός δισδιάστατου πίνακα Β 10Χ5 κατά αύξουσα σειρά σύμφωνα με τα στοιχεία της 3ης στήλης** (ταξινομούμε έναν μονοδιάστατου πίνακα Α που του έχουμε μεταφέρει την 3η στήλη του δισδιάστατου και σε κάθε αντιμετάθεση στον μονοδιάστατο πίνακα αντιμεταθέτουμε και τις αντίστοιχες γραμμές του δισδιάστατου πίνακα).



1	10	16	**4**	..	7
	11	17	**5**	..	8
	12	18	**3**	..	9
	13	19	**2**	..	10
	14	20	**1**	..	12
10	15	21	**6**	..	11

1	14	20	**1**	..	12
	13	19	**2**	..	10
	12	18	**3**	..	9
	10	16	**4**	..	7
	11	17	**5**	..	8
10	15	12	**6**	..	11

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10                ! μεταφορά του δισδιάστατου στον μονοδιάστατο
    Α[Ι] ← Β[Ι,3]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 10 ΜΕΧΡΙ Ι ΜΕ ΒΗΜΑ -1
        ΑΝ Α[Κ] < Α[Κ-1] ΤΟΤΕ          ! ταξινόμηση του μονοδιάστατου
            ΒΟΗΘ ← Α[Κ]
            Α[Κ] ← Α[Κ-1]
            Α[Κ-1] ← ΒΟΗΘ
            ΓΙΑ Λ ΑΠΟ 1 ΜΕΧΡΙ 5        ! αντιμετάθεση των αντίστοιχων γραμμών του
                ΒΟΗΘ ← Β[Κ, Λ]         ! δισδιάστατου
                Β[Κ, Λ] ← Β[Κ-1, Λ]
                Β[Κ-1, Λ] ← ΒΟΗΘ
            ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
! ΧΩΡΙΣ ΧΡΗΣΗ ΜΟΝΟΔΙΑΣΤΑΤΟΥ
! ΠΙΟ ΕΥΚΟΛΟΣ ΤΡΟΠΟΣ

ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 10 ΜΕΧΡΙ Ι ΜΕ ΒΗΜΑ -1
        ΑΝ Β[Κ, 3] < Β[Κ-1, 3] ΤΟΤΕ
            ΓΙΑ Λ ΑΠΟ 1 ΜΕΧΡΙ 5
                ΒΟΗΘ ← Β[Κ, Λ]
                Β[Κ, Λ] ← Β[Κ-1, Λ]
                Β[Κ-1, Λ] ← ΒΟΗΘ
            ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

ΑΛΛΑΖΩ ΚΑΙ ΤΑ 5 ΣΤΟΙΧΕΙΑ ΤΗΣ ΓΡΑΜΜΗΣ

ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ

Ο αλγόριθμος της δυαδικής αναζήτησης (binary search) εφαρμόζεται μόνο σε πίνακες που έχουν ταξινομημένα στοιχεία. Αν τα στοιχεία δεν είναι ταξινομημένα, τότε δεν μπορεί να εφαρμοστεί.

Ο αλγόριθμος λειτουργεί ως εξής:

- Βρίσκουμε το μεσαίο στοιχείο του ταξινομημένου πίνακα.
- Εάν το προς αναζήτηση στοιχείο είναι ίσο με το μεσαίο στοιχείο, τότε σταματάμε την αναζήτηση αφού το στοιχείο βρέθηκε.
- Εάν δε βρέθηκε, τότε ελέγχουμε αν το στοιχείο που αναζητούμε είναι μικρότερο ή μεγαλύτερο από το μεσαίο στοιχείο του πίνακα. Αν είναι μικρότερο, περιορίζουμε την αναζήτηση στο πρώτο μισό του πίνακα (με την προϋπόθεση ότι τα στοιχεία είναι διατεταγμένα κατά αύξουσα σειρά), ενώ αν είναι μεγαλύτερο περιορίζουμε την αναζήτηση στο δεύτερο μισό του πίνακα.
- Η διαδικασία αυτή επαναλαμβάνεται για το κατάλληλο πρώτο ή δεύτερο μισό του πίνακα, μετά για το 1/4 του πίνακα κ.ο.κ. μέχρι, είτε να βρεθεί το στοιχείο, είτε να μην είναι δυνατό να χωρισθεί ο πίνακας περαιτέρω σε δύο νέα μέρη.

```
5   ΑΡΧΗ
6     ΓΡΑΨΕ 'Εισάγετε αριθμούς σε αύξουσα διατάξη'
7     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
8       ΓΡΑΨΕ 'Δώσε το', i, ' στοιχείο του πίνακα'
9       ΔΙΑΒΑΣΕ A[i]
10    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
11    ΓΡΑΨΕ 'Δωσε τιμή για αναζήτηση: '
12    ΔΙΑΒΑΣΕ S   !τιμή που γάχω
13    Left <- 1
14    Right <- 20
15    k <- 0   !Θέση
16    f <- ΨΕΥΔΗΣ
17    ΟΣΟ (Left <= Right) ΚΑΙ (f = ΨΕΥΔΗΣ) ΕΠΑΝΑΛΑΒΕ
18      M <- (Left + Right) div 2
19      ΑΝ A[M] = S ΤΟΤΕ
20        k <- M
21        f <- ΑΛΗΘΗΣ
22      ΑΛΛΙΩΣ
23        ΑΝ A[M] < S ΤΟΤΕ
24          Left <- M + 1
25        ΑΛΛΙΩΣ
26          Right <- M - 1
27        ΤΕΛΟΣ_ΑΝ
28      ΤΕΛΟΣ_ΑΝ
29    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
30    ΑΝ f = ΑΛΗΘΗΣ ΤΟΤΕ
31      ΓΡΑΨΕ "Το στοιχείο,", S, "υπάρχει στη θέση:", M
32    ΑΛΛΙΩΣ
33      ΓΡΑΨΕ "Το στοιχείο,", S, " δεν υπάρχει στον πίνακα"
34    ΤΕΛΟΣ_ΑΝ
35  ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΑΥΞΟΥΣΑ

ΣΟΣ

ΘΕΩΝΟΥΣΑ ΑΝΑΠΟΔΑ

Δίνεται ο πίνακας

1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Αναζήτηση του στοιχείου 38 (υπάρχει στον πίνακα)

Βήμα 1	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 2	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 3	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 4	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Με κίτρινο σημειώνεται το στοιχείο του πίνακα που εξετάζεται (στο μέσον)

Με πράσινο σημειώνεται το τμήμα του πίνακα που απομένει για αναζήτηση

Με κόκκινο σημειώνεται το τμήμα του πίνακα που έχει αποκλειστεί

Αναζήτηση του στοιχείου 39 (δεν υπάρχει στον πίνακα)

Βήμα 1	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 2	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 3	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 4	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 5	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Αριθμός συγκρίσεων στη δυαδική αναζήτηση

Στοιχεία Ν	Συγκρίσεις
10	4
100	7
1.000	10
10.000	14
100.000	17
1.000.000	20
10.000.000	24
100.000.000	27
1.000.000.000	30

ΤΑΞΙΝΌΜΗΣΗ ΜΕ ΕΠΙΛΟΓΉ (SELECTION SORT)

Η ταξινόμηση με επιλογή (selection sort), αποτελεί βασικό τρόπο ταξινόμησης, που υλοποιείται σε έναν μονοδιάστατο πίνακα σε τρία βήματα:

1. Επιλογή του ελάχιστου στοιχείου
2. Ανταλλαγή του ελάχιστου με το πρώτο στοιχείο
3. Επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία του πίνακα

Ο αλγόριθμος ταξινόμησης με επιλογή είναι ο παρακάτω.

Ο αλγόριθμος ταξινόμησης με επιλογή, να διδαχθεί ως άσκηση και να υλοποιηθεί με πρόγραμμα, όπως παρακάτω. Πέρα από το τμήμα δηλώσεων, το πρόγραμμα έχει δύο επιπλέον τμήματα, ένα τμήμα για το "γέμισμα" του πίνακα με στοιχεία και ένα τμήμα για την εκτύπωση του ταξινομημένου πίνακα.

```
1  ΠΡΟΓΡΑΜΜΑ Selection_Sort
2  ΜΕΤΑΒΛΗΤΕΣ
3    ΑΚΕΡΑΙΕΣ: A[20], k, x, z, j
4  ΑΡΧΗ
5    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
6      ΓΡΑΨΕ 'Δώσε το ', i, ' στοιχείο του πίνακα'
7      ΔΙΑΒΑΣΕ A[i]
8    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
9    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 19
10     k <- i
11     x <- A[i]
12     ΓΙΑ j ΑΠΟ i + 1 ΜΕΧΡΙ 20
13       ΑΝ x > A[j] ΤΟΤΕ
14         k <- j
15         x <- A[j]
16       ΤΕΛΟΣ_ΑΝ
17     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
18     A[k] <- A[i]
19     A[i] <- x
20   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
21   ΓΡΑΨΕ 'Εκτύπωση με ταξινομημένα τα στοιχεία'
22   ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
23     ΓΡΑΨΕ A[i]
24   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
25 ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παράδειγμα

Αν υποθέσουμε ότι έχουμε τον πίνακα Α[8] με στοιχεία τους αριθμούς 46, 55, 12, 42, 94, 18, 06, 67.

Δηλαδή σε μορφή μονοδιάστατου πίνακα:

46	55	12	42	94	18	6	67

τότε παρακάτω φαίνεται πως μετακινούνται τα στοιχεία με τον αλγόριθμο SelectionSort

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 1:	46	55	12	42	94	18	6	67

εύρεση του ελάχιστου των στοιχείων και ανταλλαγή με το πρώτο

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 2:	6	55	12	42	94	18	46	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το δεύτερο στοιχείο και έπειτα

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 3:	6	12	55	42	94	18	46	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το τρίτο στοιχείο και έπειτα

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 4:	6	12	18	42	94	55	46	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το τέταρτο στοιχείο και έπειτα

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 5:	6	12	18	42	94	55	46	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το πέμπτο στοιχείο και έπειτα

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 6:	6	12	18	42	46	55	94	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το έκτο στοιχείο και έπειτα

	1ο	2ο	3ο	4ο	5ο	6ο	7ο	8ο
Βήμα 7:	6	12	18	42	46	55	94	67

επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το έβδομο στοιχείο και έπειτα

Τελική μορφή ταξινομημένου πίνακα (δε χρειάζεται 8η επανάληψη σύγκρισης, αφού όταν απομένουν δύο μόνο κελιά και στο πρώτο τοποθετηθεί ο μικρότερος αριθμός, τότε στο δεύτερο αναγκαστικά τίθεται ο μεγαλύτερος).

6	12	18	42	46	55	67	94

ΚΕΦΑΛΑΙΟ 4ο
ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

4.1 Ανάλυση προβλημάτων

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

1. την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
2. την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
3. την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του και στη συνέχεια:
4. την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
5. την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

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

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

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

Γενικότερα, οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον για τους εξής λόγους:

- παρέχουν ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
- μπορούν να αναπαρασταθούν με κοινές δομές δεδομένων και ελέγχου (που υποστηρίζονται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού)
- παρέχουν τη δυνατότητα καταγραφής των χρονικών και "χωρικών" απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

ΚΕΦΑΛΑΙΟ 6ο
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Φυσικές και τεχνητές γλώσσες

Οι γλώσσες προγραμματισμού ή τεχνητές γλώσσες χρησιμοποιούνται για την επικοινωνία του ανθρώπου με τον υπολογιστή, σε αντίθεση με τις φυσικές γλώσσες που χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπων. Οι τεχνητές γλώσσες ακολουθούν τους κανόνες των φυσικών γλωσσών δηλαδή έχουν το αλφάβητο τους, το λεξιλόγιο τους, τη γραμματική τους και τη σημασιολογία τους.

Το αλφάβητο

Καλείται το σύνολο των στοιχείων (χαρακτήρων) που χρησιμοποιείται από τη γλώσσα.

Το λεξιλόγιο

Αποτελείται από ένα υποσύνολο όλων των συνδυασμών που δημιουργούνται από το αλφάβητο, λέξεις που είναι αποδεκτές από μία γλώσσα. Πχ η λέξη *ζσαα* δεν είναι αποδεκτή στην ελληνική γλώσσα.

Η Γραμματική

Η γραμματική αποτελείται από το **τυπικό ή τυπολογικό** και το **συντακτικό**.

Τυπικό είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μία λέξη είναι αποδεκτή πχ οι κλίσεις μίας λέξης.

Συντακτικό είναι το σύνολο των κανόνων που καθορίζει τη ορθότητα της διάταξης και σύνδεσης των λέξεων της γλώσσας για την δημιουργία των προτάσεων. Πχ η εντολή *ΔΙΑΒΑΣΕ a b c* δεν είναι αποδεκτή γιατί οι μεταβλητές a, b, c δεν χωρίζονται μεταξύ τους με κόμμα «,».

Η σημασιολογία

Είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων, όπως συμβαίνει στις τεχνητές γλώσσες.

Διαφορές φυσικών και τεχνητών γλωσσών

Η σημαντικότερη διαφορά είναι ότι οι φυσικές γλώσσες εξελίσσονται συνεχώς (λέξεις, συντακτικό, γραμματική), αφού αλλάζει και εξελίσσεται συνεχώς και η επικοινωνία των ανθρώπων σε αντίθεση με τις τεχνητές γλώσσες.

Ωστόσο και οι γλώσσες προγραμματισμού μεταβάλλονται με καινούργιες εκδόσεις που διορθώνουν αδυναμίες προηγούμενων εκδόσεων και προσθέτουν νέες δυνατότητες. Οι αλλαγές μπορεί να είναι σε επίπεδο διαλέκτου (GW Basic σε Qbasic) ή σε επίπεδο επέκτασης (Basic σε Visual Basic).

Τεχνικές σχεδίασης προγραμμάτων

Ιεραρχική σχεδίαση προγράμματος

Η επίλυση ενός προβλήματος σχεδιάζεται «από πάνω προς τα κάτω» δηλ περιλαμβάνει τον καθορισμό των βασικών λειτουργιών σε ανώτερο επίπεδο, και στη συνέχεια διασπάται αυτή η λειτουργία σε μικρότερες λειτουργίες συνεχώς, ώστε στο τελευταίο επίπεδο οι λειτουργίες αυτές να είναι εύκολο να επιλυθούν.

Σκοπός είναι ένα πρόβλημα να χωρίζεται συνεχώς σε μικρότερα, ώστε να είναι εύκολη η επίλυση τους και κατ' επέκταση και του αρχικού προβλήματος.

Τμηματικός προγραμματισμός

Η ιεραρχική σχεδίαση πραγματοποιείται με τον τμηματικό προγραμματισμό, αφού μετά τον χωρισμό του προβλήματος σε υποπροβλήματα, το κάθε υποπρόβλημα αποτελεί μία ανεξάρτητη ενότητα που η λύση της γράφεται ξεχωριστά. Τα **πλεονεκτήματα** του τμηματικού προγραμματισμού είναι η εύκολη δημιουργία του προγράμματος, η μείωση λαθών, η εύκολη κατανόηση και συντήρηση του.

Δομημένος προγραμματισμός

Ο δομημένος προγραμματισμός στηρίζεται στην χρήση των τριών δομών ακολουθίας, επιλογής και επανάληψης. Όλα τα προγράμματα μπορούν να γραφούν με τη χρήση αυτών των τριών δομών με αποτέλεσμα την αποφυγή της εντολής GOTO που χρησιμοποιείτο μέχρι τότε κατά κόρον. 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*.

ΜΗ ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

```
ΔΙΑΒΑΣΕ Β
ΑΝ Β < 0 Η Β > 20 ΤΟΤΕ GOTO 1
ΑΝ Β>= 9,5 ΤΟΤΕ GOTO 2
ΓΡΑΨΕ 'ΔΕΝ ΠΡΟΑΓΕΤΑΙ'
GOTO 3
1 :     ΓΡΑΨΕ 'ΛΑΘΟΣ'
        GOTO 3
2 :     ΓΡΑΨΕ 'ΠΡΟΑΓΕΤΑΙ'
3:      …. ! ΣΥΝΕΧΕΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

```
ΔΙΑΒΑΣΕ Β
ΑΝ Β < 0 Η Β > 20 ΤΟΤΕ
    ΓΡΑΨΕ 'ΛΑΘΟΣ'
ΑΛΛΙΩΣ_ΑΝ Β>= 9,5 ΤΟΤΕ
    ΓΡΑΨΕ 'ΠΡΟΑΓΕΤΑΙ'
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΔΕΝ ΠΡΟΑΓΕΤΑΙ'
ΤΕΛΟΣ_ΑΝ
```

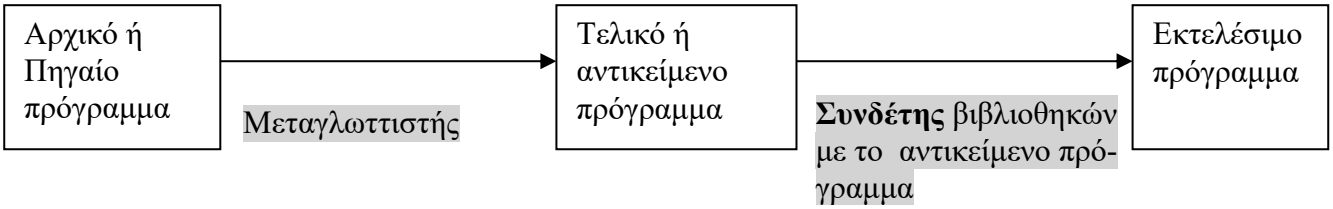
Πλεονεκτήματα του δομημένου προγραμματισμού:

- Δημιουργία απλούστερων προγραμμάτων
- Άμεση μεταφορά των αλγορίθμων σε προγράμματα
- Διευκόλυνση ανάλυσης προγράμματος σε τμήματα
- Περιορισμός λαθών του κώδικα
- Ευκολότερη ανάγνωση και κατανόηση από τρίτους
- Ευκολότερη διόρθωση και συντήρηση

Προγραμματιστικά περιβάλλοντα

Όταν γράφουμε ένα πρόγραμμα σε μία γλώσσα υψηλού επιπέδου ουσιαστικά χρησιμοποιούμε το λεξικό και το συντακτικό της γλώσσας, ουσιαστικά γράφουμε ένα κείμενο με κάποιους κανόνες με έναν μικρό επεξεργαστή κειμένου που καλείται **συντάκτης**. Τα προγράμματα που γράφουμε με τις γλώσσες προγραμματισμού δεν είναι κατανοητά από τον υπολογιστή που καταλαβαίνει μόνο την δυαδική μορφή (γλώσσα μηχανής). Η μετατροπή αυτή γίνεται με τη χρήση ειδικών προγραμμάτων που χωρίζονται σε δύο κατηγορίες:

Μεταγλωττιστές (compiler) δέχεται σαν είσοδο το πρόγραμμα γραμμένο σε μία γλώσσα προγραμματισμού (ένα κείμενο μη κατανοητό από τον υπολογιστή) και μας επιστρέφει ένα πρόγραμμα σε γλώσσα μηχανής το οποίο μπορεί να εκτελεστεί από οποιονδήποτε υπολογιστή.



Για την δημιουργία ενός προγράμματος χρειάζονται τουλάχιστον τα προγράμματα: συντάκτης, συνδέτης και μεταγλωττιστής.

Για την αρχική σύνταξη του κάθε προγράμματος χρειάζεται ένα ειδικό πρόγραμμα που καλείται **συντάκτης** (editor).

Μετά ο **μεταγλωττιστής** μετατρέπει το **αρχικό πρόγραμμα** σε **αντικείμενο πρόγραμμα,** το οποίο είναι κατανοητό από τον υπολογιστή αλλά δεν μπορεί να εκτελεστεί, αφού πρέπει να συνδεθεί με άλλα έτοιμα τμήματα προγράμματος που έχει γράψει ο προγραμματιστής ή βρίσκονται στις βιβλιοθήκες της γλώσσας.

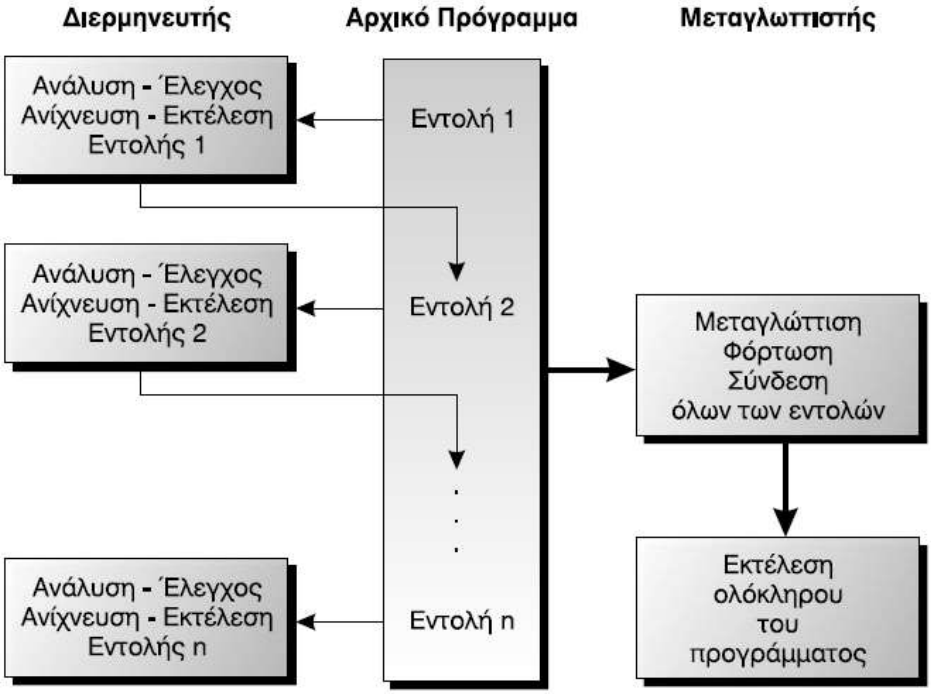
Την σύνδεση του αντικείμενου προγράμματος με τα έτοιμα τμήματα προγραμμάτων αναλαμβάνει ο **συνδέτης – φορτωτής** και έτσι έχουμε το **εκτελέσιμο πρόγραμμα,** που μπορεί να εκτελεστεί χωρίς την ύπαρξη πλέον της γλώσσας προγραμματισμού που αναπτύχθηκε.

Η **μετατροπή του πηγαίου προγράμματος σε εκτελέσιμο** γίνεται μόνο στην περίπτωση που **δεν υπάρχουν λάθη συντακτικά.** Αν το πρόγραμμα έχει λάθη σταματάει η μετατροπή του σε εκτελέσιμο, μας εμφανίζει τα λάθη τα οποία πρέπει να διορθώσουμε και το επαναυποβάλλουμε για μεταγλώττιση, μέχρι να μην ανιχνεύσει κανένα λάθος και να πραγματοποιήσει την μετατροπή.

Τα λάθη είναι δύο ειδών:

Συντακτικά λάθη εμφανίζονται κατά την μεταγλώττιση του πηγαίου προγράμματος και είναι εύκολη η διόρθωση τους, αφού μας εμφανίζεται μήνυμα σχετικό με το λάθος και τη θέση του στο πρόγραμμα.
Λογικά λάθη εμφανίζονται κατά την εκτέλεση του προγράμματος και είναι πολύ δύσκολη η εύρεση και διόρθωση τους, αφού δεν ξέρουμε το σημείο του λάθους.

Διερμηνευτές (interpreter) ο οποίος μεταφράζει και εκτελεί μία-μία τις εντολές του αρχικού προγράμματος, χωρίς να δημιουργεί το αντίστοιχο πρόγραμμα σε γλώσσα μηχανής. Πλεονέκτημα η άμεση εκτέλεση και συνεπώς η άμεση διόρθωση. Και σε αυτή την περίπτωση εκτελούνται οι εντολές όταν δεν υπάρχουν συντακτικά λάθη, αλλιώς πρέπει να διορθωθούν, ώστε να μπορεί να εκτελεστεί. Η εκτέλεση του προγράμματος είναι πιο αργή σε σχέση με τον μεταγλωττιστή και προτιμάται στην περίπτωση δοκιμών του προγράμματος, όπου είναι άμεση η εκτέλεση και η εκσφαλμάτωση του προγράμματος.



Οπότε όταν δημιουργούμε ένα πρόγραμμα κάνουμε τις δοκιμές και τους ελέγχους με τον διερμηνευτή χρησιμοποιώντας τον από την γλώσσα προγραμματισμού αφού είναι υποπρόγραμμα της, ενώ όταν έχουμε τελειώσει το πρόγραμμα κάνουμε μεταγλώττιση και το τελικό πρόγραμμα μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστή, χωρίς να απαιτείται η γλώσσα προγραμματισμού για την εκτέλεση του με μεγαλύτερη ταχύτητα.

Όπως αναφέρει και νωρίτερα τα προγράμματα που γράφουμε σε μία γλώσσα υψηλού επιπέδου δεν μπορούν να εκτελεστούν απ' ευθείας, αλλά πρέπει να μετατραπούν σε γλώσσα μηχανής (ακολουθίες 0 και 1). Αυτό μπορούμε να το επιτύχουμε **με τον διερμηνευτή ή με τον μεταγλωττιστή**.

Ομοιότητες:

1. Χρησιμοποιούνται για την μετατροπή ενός προγράμματος από γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής.
2. Και τα δύο προγράμματα δεν μετατρέπουν το αρχικό πρόγραμμα σε γλώσσα μηχανής αν έχει συντακτικά λάθη. Αν κατά τη διαδικασία μετατροπής εντοπίσουν συντακτικά λάθη, σταματάει η μετατροπή σε εκείνο το σημείο για να το διορθώσουμε. Στη συνέχεια υποβάλλουμε ξανά το αρχικό πρόγραμμα σε μετατροπή μέχρι να μην υπάρχουν συντακτικά λάθη.

Διαφορές:

1. Ο διερμηνευτής *μετατρέπει και εκτελεί ταυτόχρονα μία μία εντολή*, σε αντίθεση με το μεταγλωττιστή *που δημιουργεί ένα νέο αρχείο (χωρίς να το εκτελεί) σε γλώσσα μηχανής* από την μετατροπή όλων μαζί των εντολών του αρχικού προγράμματος.
2. Το εκτελέσιμο πρόγραμμα που παράγεται από τον μεταγλωττιστή είναι πλέον ανεξάρτητο από την γλώσσα προγραμματισμού και εκτελείται χωρίς να απαιτείται η γλώσσα.
3. Ο διερμηνευτής χρησιμοποιείται για άμεση εκτέλεση του προγράμματος και διόρθωση του αρχικού προγράμματος, ενώ ο μεταγλωττιστής για την μετατροπή του αρχικού προγράμματος σε ένα νέο ανεξάρτητο εκτελέσιμο πρόγραμμα έτοιμο προς χρήση.
4. Ο διερμηνευτής είναι πιο αργός από τον μεταγλωττιστή, αφού το αρχείο που δημιουργεί ο μεταγλωττιστής είναι σε γλώσσα μηχανής και δεν χρειάζεται μετατροπή ξανά, αλλά εκτελείται απ' ευθείας σε σχέση με το διερμηνευτή που μετατρέπει κάθε φορά εντολή – εντολή σε γλώσσα μηχανής για να την εκτελέσει.

ΚΕΦΑΛΑΙΟ 7ο

Δομή ενός προγράμματος στη 'ΓΛΩΣΣΑ'

ΠΡΟΓΡΑΜΜΑ ΟΝΟΜΑ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΤΑΘΕΡΕΣ

```
    Ν = 20                  ! ΑΚΕΡΑΙΕΣ
    Χ = 13,5                ! ΠΡΑΓΜΑΤΙΚΕΣ
    ΥΛΙΚΟ = ΄ΞΥΛΟ΄          ! ΧΑΡΑΚΤΗΡΕΣ
    ΒΡΕΘΗΚΕ = ΨΕΥΔΗΣ        ! ΛΟΓΙΚΟΥ ΤΥΠΟΥ
```

ΜΕΤΑΒΛΗΤΕΣ

```
    ΑΚΕΡΑΙΕΣ : Ι, Κ
    ΠΡΑΓΜΑΤΙΚΕΣ : Λ, Μ
    ΧΑΡΑΚΤΗΡΕΣ : ΕΠΩΝΥΜΟ[Ν]
    ΛΟΓΙΚΕΣ : FOUND
```

ΑΡΧΗ

```
    ΓΡΑΨΕ ΄Δώσε έναν αριθμό ΄
    ΔΙΑΒΑΣΕ Λ, Μ
    ……
    ……
```

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Όπως φαίνεται από τη δομή ενός προγράμματος σε ΓΛΩΣΣΑ σαν επικεφαλίδα χρησιμοποιούμε την λέξη **ΠΡΟΓΡΑΜΜΑ** ακολουθούμενη από το όνομα του προγράμματος. Στο επόμενο τμήμα με τις δηλώσεις πρώτα αναφέρουμε τις **ΣΤΑΘΕΡΕΣ** και στη συνέχεια τις **ΜΕΤΑΒΛΗΤΕΣ**.

Σε μία **σταθερά** ορίζουμε θέσεις μνήμης με κάποιο όνομα, όπου το περιεχόμενο τους δεν μπορεί να αλλάξει κατά την εκτέλεση του προγράμματος αλλά διατηρεί την τιμή που δηλώνουμε μετά το όνομα της καθ' όλη τη διάρκεια του .

Σε μία **μεταβλητή** η τιμή που έχουν οι θέσεις μνήμης μπορεί να αλλάξει κατά την διάρκεια εκτέλεσης του προγράμματος και να πάρει μία νέα τιμή του ιδίου τύπου με τον τύπο που δηλώσαμε όταν ορίσαμε τις μεταβλητές.

Άρα και οι σταθερές και οι μεταβλητές είναι θέσεις της μνήμης RAM με συγκεκριμένο όνομα, όπου στις πρώτες δεν μπορούμε να μεταβάλλουμε την τιμή τους κατά τη διάρκεια εκτέλεσης του προγράμματος σε αντίθεση με τις μεταβλητές.

Οι τύποι των μεταβλητών και των σταθερών είναι τέσσερις: **ΑΚΕΡΑΙΕΣ, ΠΡΑΓΜΑΤΙΚΕΣ, ΧΑΡΑΚΤΗΡΕΣ και ΛΟΓΙΚΕΣ.**

Το κυρίως πρόγραμμα ξεκινάει με την εντολή **ΑΡΧΗ** και τελειώνει με την εντολή **ΤΕΛΟΣ_ ΠΡΟΓΡΑΜΜΑΤΟΣ**.

Επιπλέον όταν θέλουμε να εμφανίσουμε κάτι στην οθόνη χρησιμοποιούμε την εντολή **ΓΡΑΨΕ** και ότι μηνύματα **βάζουμε μονά εισαγωγικά ΄.**

Όλες οι εντολές είναι δεσμευμένες λέξεις, που σημαίνει ότι δεν μπορούν να χρησιμοποιηθούν σαν ονόματα μεταβλητών ή σταθερών. Οι λέξεις που χρησιμοποιούμε για ονόματα προγραμμάτων, υποπρογραμμάτων, μεταβλητών και σταθερών λέγονται και αναγνωριστικά. Κάθε αναγνωριστικό πρέπει να ξεκινάει από γράμμα, να μην έχει κενά ανάμεσα, και να μην υπάρχουν ειδικοί χαρακτήρες μέσα στο όνομα εκτός από την κάτω γραμμή '_'.

Στην ΓΛΩΣΣΑ είναι <u>υποχρεωτική η δήλωση των σταθερών και των μεταβλητών</u>

Ένα άλλο πλεονέκτημα της ΓΛΩΣΣΑΣ είναι και οι **έτοιμες συναρτήσεις** (που μας επιστρέφουν κάποια αποτελέσματα ανάλογα με τις τιμές που τους στέλνουμε), *<u>που μπορούμε να χρησιμοποιήσουμε μόνο όταν γράφουμε σε ΓΛΩΣΣΑ</u>*:

ΗΜ(Χ)	Ημίτονο του Χ
ΣΥΝ(Χ)	Συνημίτονο του Χ
ΕΦ(Χ)	Εφαπτομένη του Χ
Τ_Ρ(Χ)	Τετραγωνική Ρίζα του Χ
ΛΟΓ(Χ)	Λογάριθμος του Χ
Ε(Χ)	Υπολογισμός του e^x
Α_Μ(Χ)	Ακέραιο μέρος του Χ
Α_Τ(Χ)	Απόλυτη τιμή του Χ

Παράδειγμα το διάβασμα ενός αριθμού και η εμφάνιση στην οθόνη του διπλάσιου του.

```
ΠΡΟΓΡΑΜΜΑ ΔΙΠΛΑΣΙΟ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: Χ
ΑΡΧΗ
    ΓΡΑΨΕ ΄ΔΩΣΕ ΕΝΑΝ ΑΡΙΘΜΟ: ΄
    ΔΙΑΒΑΣΕ Χ
    ΓΡΑΨΕ ΄ΤΟ ΔΙΠΛΑΣΙΟ ΤΟΥ ΑΡΙΘΜΟΥ ΠΟΥ ΕΔΩΣΕΣ ΕΙΝΑΙ : ΄,2*Χ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΚΕΦΑΛΑΙΟ 8ο

Εντολές Επιλογής και Επανάληψης

Εντολές Επιλογής

Στις εντολές ελέγχου έχει γίνει λεπτομερής αναφορά στο 2ο Κεφάλαιο.

Την απλή επιλογή την χρησιμοποιούμε όταν έχουμε να ελέγξουμε μόνο μία περίπτωση, δηλ αν θα εκτελεστεί μία ομάδα εντολών (όταν ισχύει η συνθήκη) ή όχι (όταν δεν ισχύει η συνθήκη).

Την σύνθετη επιλογή την χρησιμοποιούμε όταν έχουμε να ελέγξουμε μόνο δύο περιπτώσεις, και να εκτελεστούν οι εντολές της μίας περίπτωσης ή της άλλης.

Πολλαπλή επιλογή έχουμε όταν πρέπει να ελέγξουμε ποια από τις περιπτώσεις (τουλάχιστον 3 περιπτώσεις) ισχύει και να εκτελεστούν οι αντίστοιχες εντολές.

Εμφωλευμένη επιλογή στην περίπτωση που έχουμε έλεγχο μέσα σε έναν άλλο έλεγχο.

Εντολές επανάληψης όταν μία ομάδα εντολών μπορεί να εκτελεστεί από καμία, μέχρι πολλές. Οι επαναλήψεις ελέγχονται από μία συνθήκη, από την οποία καθορίζεται η έξοδος από τον βρόχο. Επίσης πρέπει να μεταβάλλονται οι τιμές των μεταβλητών της συνθήκης, διότι να μπει και δεν μεταβάλλονται δεν θα τερματιστεί ποτέ ο βρόχος.

Την ΕΝΤΟΛΗ ΓΙΑ την χρησιμοποιούμε όταν είναι γνωστό το πλήθος των επαναλήψεων, αφού πρέπει να γνωρίζουμε την αρχική τιμή του μετρητή, την τελική τιμή και το βήμα. Ο έλεγχος της συνθήκης γίνεται πριν την εκτέλεση της ομάδας εντολών, άρα η ομάδα εντολών μπορεί να μην εκτελεστεί καμία φορά.

Την ΕΝΤΟΛΗ ΟΣΟ την χρησιμοποιούμε όταν δεν ξέρουμε το πλήθος των επαναλήψεων (μπορούμε και όταν το ξέρουμε να την χρησιμοποιήσουμε, αλλά σε αυτή την περίπτωση είναι καλύτερη η ΓΙΑ) και ο έλεγχος της συνθήκης γίνεται πριν εκτελέσουμε την ομάδα εντολών. Άρα στην ΟΣΟ μπορεί να μην ισχύει η συνθήκη από τον πρώτο έλεγχο και να μην μπει καμία φορά στις εντολές επανάληψης, όπως και η ΓΙΑ. Η επανάληψη τελειώνει όταν η συνθήκη γίνει ψευδής.

Την ΕΝΤΟΛΗ ΜΕΧΡΙΣ_ΟΤΟΥ την χρησιμοποιούμε όταν δεν ξέρουμε το πλήθος των επαναλήψεων (μπορούμε και όταν το ξέρουμε να την χρησιμοποιήσουμε, αλλά σε αυτή την περίπτωση είναι καλύτερη η ΓΙΑ) και ο έλεγχος της συνθήκης γίνεται μετά την εκτέλεση της ομάδα εντολών. Οπότε στην ΜΕΧΡΙΣ_ΟΤΟΥ οι εντολές επανάληψης θα εκτελεστούν τουλάχιστον μία φορά και μετά θα γίνει ο έλεγχος. Η επανάληψη τελειώνει όταν η συνθήκη γίνει αληθής.

Η χρήση των εμφωλευμένων δομών πρέπει να ακολουθεί τους παρακάτω κανόνες:

1) Ο εσωτερικός βρόχος πρέπει να είναι ολόκληρος μέσα στον εξωτερικό και ο βρόχος που ξεκινάει τελευταίος πρέπει να ολοκληρώνεται πρώτος.
2) Η είσοδος κάθε βρόχου γίνεται από την αρχή του.

3) Δεν πρέπει να χρησιμοποιείται η ίδια μεταβλητή ως μετρητής δύο εμφωλευμένων βρόχων που ο ένας είναι μέσα στον άλλο.

Ας δούμε σαν παράδειγμα ένα πρόγραμμα που διαβάζει τρεις ακέραιους αριθμούς και εμφανίζει στην οθόνη το άθροισμα των δύο μικρότερων. Σε αυτό το πρόγραμμα βρίσκουμε τον μεγαλύτερο και προσθέτουμε τους άλλους δύο.

```
ΠΡΟΓΡΑΜΜΑ ΑΘΡΟΙΣΜΑ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Α1, Α2, Α3, Σ
ΑΡΧΗ
    ΓΡΑΨΕ ΄ΔΩΣΕ ΤΡΕΙΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ: ΄
    ΔΙΑΒΑΣΕ Α1, Α2, Α3
    ΑΝ Α1 >= Α2 ΚΑΙ Α1>= Α3 ΤΟΤΕ      !Α1 ΜΕΓΑΛΥΤΕΡΟΣ
        ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ ΔΥΟ ΜΙΚΡΟΤΕΡΩΝ ΕΙΝΑΙ : ΄,Α2+Α3
    ΑΛΛΙΩΣ_ΑΝ Α2>=Α3 ΤΟΤΕ            !Α2 ΜΕΓΑΛΥΤΕΡΟΣ
        ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ ΔΥΟ ΜΙΚΡΟΤΕΡΩΝ ΕΙΝΑΙ : ΄,Α1+Α3
    ΑΛΛΙΩΣ                            !Α3 ΜΕΓΑΛΥΤΕΡΟΣ
        ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ ΔΥΟ ΜΙΚΡΟΤΕΡΩΝ ΕΙΝΑΙ : ΄,Α1+Α2
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ας δούμε άλλο ένα πρόγραμμα που θα διαβάζει 10 αριθμούς και θα εμφανίζει το άθροισμα και το γινόμενο τους.

```
ΠΡΟΓΡΑΜΜΑ ΑΘΡΟΙΣΜΑ_ΓΙΝΟΜΕΝΟ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Ι, Σ, Γ, Χ
ΑΡΧΗ
    Γ ← 1
    Σ ← 0
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟΝ ΄,Ι,΄ ΑΡΙΘΜΟ : ΄
        ΔΙΑΒΑΣΕ Χ
        Γ ← Γ * Χ
        Σ ← Σ +Χ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ ΑΡΙΘΜΩΝ ΕΊΝΑΙ ΄, Σ , ΄ ΚΑΙ ΤΟ ΓΙΝΌΜΕΝΟ ΄,Γ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΚΕΦΑΛΑΙΟ 9ο

Πίνακες

Πίνακας είναι ένα σύνολο αντικειμένων ιδίου τύπου, τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγονται **στοιχείο** του πίνακα. Η αναφορά ξεχωριστά σε κάθε στοιχείο του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν δείκτη.

Υπάρχει μία δυσκολία στο να καταλάβουμε πότε χρειαζόμαστε πίνακες και πότε όχι. Ο κανόνας είναι ότι να χρειαζόμαστε τα δεδομένα μας παραπάνω από μία φορά, για να μην τα εισάγουμε ξανά τα τοποθετούμε σε έναν πίνακα για να μην τα χάσουμε. Για παράδειγμα αν πρέπει να διαβάζουμε 100 αριθμούς και να βρούμε τον μεγαλύτερο δεν χρειαζόμαστε πίνακα, αφού μπορούμε κατά την εισαγωγή να κάνουμε και τον έλεγχο αν ο καινούργιος αριθμός που διαβάζαμε είναι μεγαλύτερος από τον όλους τους προηγούμενους. Αν όμως ήθελε να βρούμε ποιοι μαθητές σε μία τάξη έχουν βαθμό μεγαλύτερο από το ΜΟ, τότε πρέπει υποχρεωτικά να χρησιμοποιήσουμε πίνακα, αφού τα δεδομένα τα χρειαζόμαστε δύο φορές, μία φορά διαβάζουμε τους βαθμούς και βρίσκουμε το άθροισμα τους ταυτόχρονα και τον ΜΟ αμέσως μετά, και άλλη μία φορά για να ελέγξουμε αν ο βαθμός του κάθε μαθητή είναι μεγαλύτερος από το ΜΟ.

Από τα παραπάνω συμπεραίνουμε ότι οι πίνακες είναι μία *στατική δομή δεδομένων*, που την χρησιμοποιούμε όταν έχουμε μεγάλο πλήθος ιδίου τύπου δεδομένων, που θέλουμε να τα επεξεργαστούμε παραπάνω από μία φορά. Κάθε ένα από τα αντικείμενα που αποτελούν τον πίνακα καλείται *στοιχείο* του πίνακα. Η αναφορά σε κάθε στοιχείο του πίνακα γίνεται με το όνομα του ακολουθούμενο από έναν δείκτη (για μονοδιάστατο πίνακα) ή δύο (για δισδιάστατο πίνακα).

Στατική δομή σημαίνει ότι καταλαμβάνουν συνεχόμενες θέσεις στην μνήμη RAM και έχει σταθερό μέγεθος καθ' όλη τη διάρκεια εκτέλεσης του προγράμματος.

Οι πίνακες έχουν δύο μειονεκτήματα:

1. Απαιτούν ένα μεγάλο τμήμα της RAM πχ 1000 θέσεις με χαρακτήρες, καθ' όλη τη διάρκεια εκτέλεσης του προγράμματος. Η υπερβολική χρήση πινάκων μπορεί να οδηγήσει ακόμα και σε αδυναμία εκτέλεσης του προγράμματος
2. Περιορίζουν τις δυνατότητες του προγράμματος, αφού έχουν συγκεκριμένο μέγεθος και το δηλώνουμε στις μεταβλητές του προγράμματος και δεν μπορεί να αλλάξει εκτός και κάνουμε διορθώσεις στον πηγαίο κώδικα πριν την επόμενη εκτέλεση του.

Τυπικές επεξεργασίες πινάκων

- Υπολογισμός Αθροίσματος στοιχείων πίνακα
- Εύρεση Μέγιστου ή Ελάχιστου στοιχείου
- Ταξινόμηση των στοιχείων πίνακα
- Αναζήτηση ενός στοιχείου του πίνακα
  - Σειριακή (αργή για μικρούς πίνακες)
  - Δυαδική (γρήγορη για ταξινομημένους πίνακες)
- Συγχώνευση πινάκων

ΑΠΑΓΟΡΕΥΟΝΤΑΙ

1) ΕΙΣΑΓΩΓΗ
2) ΔΙΑΓΡΑΦΗ
3) ΑΝΤΙΓΡΑΦΗ
4) ΔΙΑΧΩΡΙΣΜΟΣ?

Ας δούμε ένα πρόγραμμα που διαβάζει 30 ακέραιους αριθμούς μεταξύ 0-20 και να μας εμφανίζει στο τέλος τους αριθμούς που είναι μεγαλύτεροι του Μ.Ο. των αριθμών που δώσαμε.

```
ΠΡΟΓΡΑΜΜΑ ΠΙΝΑΚΑ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Ι, ΠΙΝ [ 30 ], Σ
    ΠΡΑΓΜΑΤΙΚΕΣ : ΜΟ
ΑΡΧΗ
    Σ ← 0
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟ ΄ , Ι , ΄ ΒΑΘΜΟ ΄
        ΔΙΑΒΑΣΕ ΠΙΝ [ Ι ]
        ΟΣΟ ΠΙΝ [ Ι ] < 0 Ή ΠΙΝ [ Ι ] > 20 ΕΠΑΝΑΛΑΒΕ          !ΕΛΕΓΧΟΣ 0-20
            ΓΡΑΨΕ ΄ΕΔΩΣΕΣ ΛΑΘΟΣ ΒΑΘΜΟ. ΞΑΝΑΔΩΣΕ ΤΟΝ ΄, Ι , ΄ ΒΑΘΜΟ΄
            ΔΙΑΒΑΣΕ ΠΙΝ [ Ι ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        Σ ← Σ + ΠΙΝ [ Ι ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΟ ← Σ / 30
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30
        ΑΝ ΠΙΝ [ Ι ] > ΜΟ ΤΟΤΕ
            ΓΡΑΨΕ ΄Ο ΄, Ι ,΄ΒΑΘΜΟΣ ΕΙΝΑΙ ΜΕΓΑΛΥΤΕΡΟΣ ΤΟΥ ΜΟ ΚΑΙ ΕΙΝΑΙ ΄, ΠΙΝ[Ι]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΚΕΦΑΛΑΙΟ 10ο

ΤΜΗΜΑΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΔΙΑΔΙΚΑΣΙΕΣ-ΣΥΝΑΡΤΗΣΕΙΣ

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.

Για την υλοποίηση του τμηματικού προγραμματισμού, αναλύουμε το πρόβλημα σε μικρότερα ανεξάρτητα υποπροβλήματα (συναρτήσεις ή διαδικασίες) που είναι εύκολη η υλοποίηση τους.

Υποπρόγραμμα είναι ένα τμήμα προγράμματος που επιτελεί ένα αυτόνομο έργο και έχει γραφτεί ξεχωριστά από το υπόλοιπο πρόγραμμα και βρίσκεται μετά το τέλος του κυρίως προγράμματος.

Το κάθε υποπρόγραμμα καλείται και εκτελείται από το αρχικό πρόγραμμα ή άλλο υποπρόγραμμα που καλείται κύριο (καλών) πρόγραμμα. **Μία συνάρτηση δεν μπορεί να καλέσει μία διαδικασία**, αφού στη διαδικασία μπορούμε να διαβάσουμε και να γράψουμε, ενώ στη συνάρτηση όχι.

Τα **χαρακτηριστικά ενός υποπρογράμματος** είναι:

1. Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο. Στην πραγματικότητα μπορεί να μην έχει καμία ή και πολλές εισόδους και τουλάχιστον μία έξοδο.
2. Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα, χωρίς να επηρεάζει και επηρεάζεται από τα υπόλοιπα (με χρήση παραμέτρων).
3. Κάθε υποπρόγραμμα δεν πρέπει να είναι πολύ μεγάλο (πρέπει να επιτελεί μόνο μία λειτουργία).

Τα **πλεονεκτήματα του τμηματικού προγραμματισμού** είναι:

1. Διευκολύνει την ανάπτυξη του αλγόριθμου και του αντίστοιχου προγράμματος, όπου με την επίλυση των υποπροβλημάτων επιλύουμε και το συνολικό πρόβλημα
2. Διευκολύνει την κατανόηση και διόρθωση του προγράμματος, αφού οι διορθώσεις γίνονται σε μικρά τμήματα χωρίς να επηρεάζουν το υπόλοιπο και πιο κατανοητό αφού έχει να καταλάβει μικρά προβλήματα και όχι ένα μεγάλο.
3. Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος, αφού ένα υποπρόγραμμα γράφεται μόνο μία φορά και το καλούμε όσες φορές θέλουμε, μειώνοντας το μέγεθος και το χρόνο συγγραφής του προγράμματος.
4. Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού, εφόσον κάθε υποπρόγραμμα μπορεί να τοποθετηθεί σε βιβλιοθήκη και να καλείται όπως οι υπόλοιπες εντολές της γλώσσας προγραμματισμού.

Συνάρτηση είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή μέσω του ονόματός της όπως οι μαθηματικές συναρτήσεις. *(έχουν περιορισμένη λειτουργία, υπολογίζουν και επιστρέφουν μόνο μία τιμή, όχι διάβασμα και εμφάνιση μέσα σε συνάρτηση και όχι κλήση της χωρίς παράμετρο)*

Διαδικασία είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος. *(διαβάζουν τιμές, τυπώνουν τιμές, υπολογίζουν, αλλάζουν τιμές μεταβλητών)*

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

Όταν καλούμε ένα υποπρόγραμμα μπορούμε να περάσουμε σε αυτό κάποια δεδομένα (μέσω μεταβλητών) από το κύριο πρόγραμμα που το καλούμε με την μορφή παραμέτρων. Χωρίζονται στις **πραγματικές παραμέτρους** που είναι μεταβλητές μέσω των οποίων στέλνουμε τιμές από το καλών τμήμα προγράμματος στο υποπρόγραμμα και τις **τυπικές παραμέτρους** που είναι οι μεταβλητές του υποπρογράμματος που δέχονται τις αντίστοιχες τιμές των πραγματικών παραμέτρων από το καλών τμήμα προγράμματος που έγινε η κλήση του. Στο πρόγραμμα ΠΑΡΑΔΕΙΓΜΑ_ΔΙΑΔΙΚΑΣΙΑΣ που ακολουθεί πραγματικές παράμετροι είναι τα X, Y, SUM και τυπικές παράμετροι τα A, B, C που όταν καλείται η διαδικασία παίρνουν αντίστοιχα τις τιμές των X, Y, SUM. Ο αριθμός των πραγματικών παραμέτρων είναι υποχρεωτικά ίδιος με των τυπικών παραμέτρων, αφού η πρώτη πραγματική παράμετρος αντιστοιχεί στην πρώτη τυπική παράμετρο, ή δεύτερη πραγματική παράμετρος αντιστοιχεί στην δεύτερη τυπική παράμετρο κ.ο.κ..

ΤΟΥΛΑΧΙΣΤΟΝ
ΜΙΑ ΠΑΡΑΜΕΤΡΟ

Ορισμός και κλήση συναρτήσεων

Η δομή μίας συνάρτησης είναι η ακόλουθη:

```
ΣΥΝΑΡΤΗΣΗ όνομα_συνάρτησης (λίστα παραμέτρων) : τύπος συνάρτησης
Τμήμα Δηλώσεων
……..! Δήλωση παραμέτρων και τοπικών μεταβλητών
ΑΡΧΗ
        ……
        Όνομα_συνάρτησης ← Έκφραση        ! υποχρεωτικά
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Η κλήση μίας συνάρτησης γίνεται απλά με το όνομα της ακολουθούμενο από τις παραμέτρους μέσα σε παρένθεση

Ορισμός και κλήση διαδικασιών

Η δομή μίας διαδικασίας είναι η ακόλουθη:

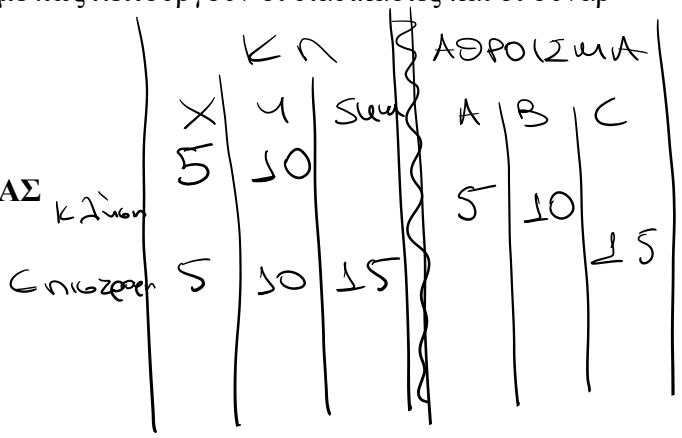
```
ΔΙΑΔΙΚΑΣΙΑ όνομα_διαδικασίας (λίστα παραμέτρων) ! ΜΠΟΡΕΙ ΝΑ ΕΙΝΑΙ ΚΑΙ ΚΕΝΗ
Τμήμα Δηλώσεων
……..! Δήλωση παραμέτρων και τοπικών μεταβλητών
ΑΡΧΗ
        ……
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Η κλήση μίας διαδικασίας γίνεται χρησιμοποιώντας την εντολή ΚΑΛΕΣΕ μετά το όνομα της διαδικασίας με τις παραμέτρους *(που θα χρειαστούμε για να στείλουμε και να πάρουμε στη διαδικασία μέσα σε παρένθεση)* στο καλών τμήμα προγράμματος. ***Μία διαδικασία μπορεί να μην έχει καμία είσοδο ή καμία έξοδο ή τίποτα και από τα δύο σαν παράμετρο.***

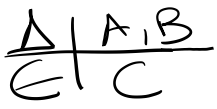
Η κλήση μίας συνάρτησης γίνεται με το όνομα της ακολουθούμενο από τις παραμέτρους μέσα σε παρένθεση και επιστρέφει μόνο μία τιμή με το όνομα της που μπορούμε:
1) να την εκχωρούμε σε μία μεταβλητή πχ ΜΕΤΑΒΛΗΤΗ ← όνομα_συνάρτησης (λίστα παραμ)
2) να την εμφανίζουμε σε μία εντολή εξόδου πχ ΓΡΑΨΕ όνομα_συνάρτησης (λίστα παραμέτρων)
3) να την χρησιμοποιούμε σε μία έκφραση πχ ΑΘΡ ← ΑΘΡ + όνομα_συνάρτησης (λίστα παραμ)
και γενικά όλους τους τρόπους που μπορούμε να χρησιμοποιήσουμε μία μεταβλητή.

Ας δούμε κάποια παραδείγματα για να καταλάβουμε πως λειτουργούν οι διαδικασίες και οι συναρτήσεις:

Παράδειγμα διαδικασίας




```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ_ΔΙΑΔΙΚΑΣΙΑΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : SUM, X, Y
ΑΡΧΗ
    ΓΡΑΨΕ ΄ΔΩΣΕ 2 ΑΡΙΘΜΟΥΣ : ΄
    ΔΙΑΒΑΣΕ X, Y
    ΚΑΛΕΣΕ ΑΘΡΟΙΣΜΑ ( X,  Y, SUM)  ! X, Y, SUM ΠΡΑΓΜΑΤΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ
    ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ 2 ΑΡΙΘΜΩΝ ΠΟΥ ΕΔΩΣΕΣ ΕΙΝΑΙ : ΄, SUM
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΑΘΡΟΙΣΜΑ (A,  B,  C)        ! A, B, C ΤΥΠΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : A, B, C
 ΑΡΧΗ
    C ← A + B
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```



Ας δούμε τώρα σύμφωνα με το παραπάνω παράδειγμα πως μπορούμε να δημιουργήσουμε και να καλέσουμε μία διαδικασία.

Θα πρέπει να γράψουμε τη διαδικασία μετά το τέλος του κυρίως προγράμματος. Έχει την ίδια μορφή με το πρόγραμμα με τη μόνη διαφορά, ότι η διαδικασία ξεκινάει με τη λέξη **ΔΙΑΔΙΚΑΣΙΑ** ακολουθούμενη από το όνομα της διαδικασίας και τις τυπικές παραμέτρους (τις οποίες δηλώνουμε

και ως μεταβλητές του υποπρογράμματος) και τελειώνει με την λέξη **ΤΕΛΟΣ_ ΔΙΑΔΙΚΑΣΙΑΣ.** Τις τυπικές παραμέτρους τις χρησιμοποιούμε για να περάσουμε από το αρχικό πρόγραμμα κάποιες τιμές στο υποπρόγραμμα, αλλά και το αντίστροφο. Στο παράδειγμα μας τις παραμέτρους Α, Β τις χρησιμοποιούμε για να περάσουμε τις τιμές των μεταβλητών Χ, Υ αντίστοιχα από το κυρίως πρόγραμμα, ενώ την παράμετρο C την χρησιμοποιούμε για να περάσουμε το άθροισμα των Α+Β από τη διαδικασία στο κυρίως πρόγραμμα μέσω της πραγματικής παραμέτρου SUM.

Για να καλέσουμε μία διαδικασία από το καλών τμήμα προγράμματος χρησιμοποιούμε την εντολή **ΚΑΛΕΣΕ** ακολουθούμενη από το όνομα της διαδικασίας και τις πραγματικές παραμέτρους μέσα σε παρένθεση πχ **ΚΑΛΕΣΕ ΑΘΡΟΙΣΜΑ (Χ, Υ, SUM).** Μόλις το πρόγραμμα εκτελέσει αυτήν την εντολή αντιστοιχίζει τις πραγματικές παραμέτρους στις τυπικές, δηλ. στέλνει την τιμή του Χ στο Α, του Υ στο Β και του SUM στο C και η επόμενη εντολή που θα εκτελεστεί είναι η πρώτη της διαδικασίας. Στην συνέχεια εκτελούνται όλες οι εντολές της διαδικασίας μέχρι να τελειώσουν. Όταν αλλάζει η τιμή μέσα στην διαδικασία μίας τυπικής παραμέτρου αλλάζει και τις πραγματικής παραμέτρου. Στο παραπάνω παράδειγμα όταν αλλάζει το C αλλάζει και η SUM.

Μετά την εκτέλεση όλων των εντολών της διαδικασίας και την ολοκλήρωση της, το πρόγραμμα εκτελεί την επόμενη εντολή από την **ΚΑΛΕΣΕ ΟΝΟΜΑ_ΔΙΑΔΙΚΑΣΙΑΣ** του κύριου προγράμματος, στο παράδειγμα μας την εντολή

ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ 2 ΑΡΙΘΜΩΝ ΠΟΥ ΕΔΩΣΕΣ ΕΙΝΑΙ: ΄, SUM.

Πρέπει επίσης να γνωρίζουμε ότι οι μεταβλητές Χ, Υ, SUM ισχύουν και στο κύριο πρόγραμμα, καθώς και μέσα στη διαδικασία, σε αντίθεση με τις μεταβλητές Α, Β, C που μπορούν να χρησιμοποιηθούν μόνο μέσα στη Διαδικασία Άθροισμα.

Τις πραγματικές παραμέτρους μπορούμε να τις χρησιμοποιήσουμε:

Α) μόνο για να στείλουμε τιμές (συναρτήσεις)

Β) για να στείλουμε και να πάρουμε τιμές (διαδικασίες)

Οι λίστες παραμέτρων πρέπει να έχουν τα εξής χαρακτηριστικά:

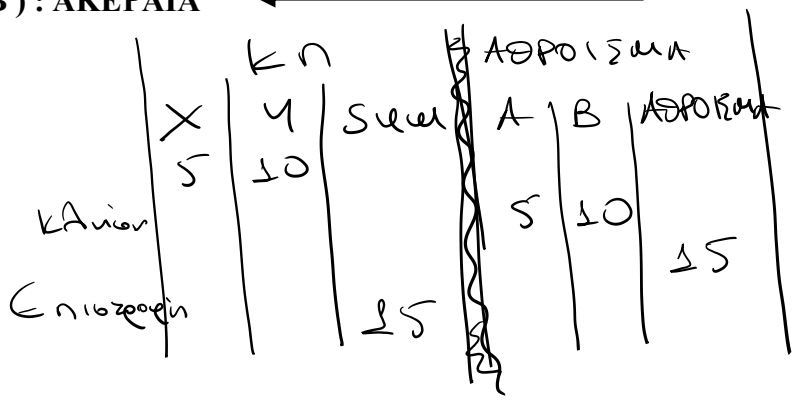
1) **Ο αριθμός των πραγματικών παραμέτρων να είναι ίδιος με των τυπικών παραμέτρων**
2) **Η πρώτη πραγματική παράμετρος αντιστοιχεί στην πρώτη τυπική, η δεύτερη πραγματική στην δεύτερη τυπική κ.ο.κ.**
3) **Οι αντίστοιχες παράμετροι (1η πραγματική με 1η τυπική, 2η πραγματική με 2η τυπική κ.ο.κ.) πρέπει να είναι του ιδίου τύπου**

Παράδειγμα συνάρτησης

ΠΛΕΟΝΕΚΤΗΜΑ ΣΥΝΑΡΤΗΣΗΣ

Οι πραγματικές παράμετροι στην κλήση μίας συνάρτησης δεν επηρεάζονται, αφού και να αλλάξουμε τις τιμές των τυπικών παραμέτρων μέσα στην συνάρτηση δεν τις επιστρέφουνε.

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ_ΣΥΝΑΡΤΗΣΗΣ

ΜΕΤΑΒΛΗΤΕΣ
      ΑΚΕΡΑΙΕΣ : SUM, X, Y

ΑΡΧΗ
      ΓΡΑΨΕ ΄ΔΩΣΕ 2 ΑΡΙΘΜΟΥΣ :΄
      ΔΙΑΒΑΣΕ X, Y
      SUM ← ΑΘΡΟΙΣΜΑ ( X,    Y )
      ΓΡΑΨΕ ΄ΤΟ ΑΘΡΟΙΣΜΑ ΤΩΝ 2 ΑΡΙΘΜΩΝ ΠΟΥ ΕΔΩΣΕΣ ΕΙΝΑΙ : ΄, SUM
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ ΑΘΡΟΙΣΜΑ(Α,    Β ) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
      ΑΚΕΡΑΙΕΣ : Α, Β

ΑΡΧΗ
      ΑΘΡΟΙΣΜΑ ← Α + Β
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```



Η διαφορά που έχει μία συνάρτηση από μία διαδικασία είναι ότι η **συνάρτηση υπολογίζει και επιστρέφει μόνο μία τιμή** μέσα από το όνομα της στο καλών τμήμα προγράμματος.

Για να καλέσουμε μία συνάρτηση από το κύριο πρόγραμμα ή από ένα άλλο υποπρόγραμμα γράφουμε το όνομα της ακολουθούμενο μέσα σε παρένθεση από τις πραγματικές παραμέτρους και την εκχωρούμε σε μία μεταβλητή, αφού όπως προαναφέραμε επιστρέφει μέσα από το όνομα της μία τιμή. Στο παράδειγμα μας θα μας επιστρέψει μία τιμή η συνάρτηση Άθροισμα που θα την βάλουμε στην μεταβλητή SUM του κύριου προγράμματος πχ **SUM ← ΑΘΡΟΙΣΜΑ (X, Y)**

Για να επιστρέψουμε την τιμή πρέπει μέσα στην συνάρτηση να υπάρχει μία εντολή εκχώρησης στο όνομα της συνάρτησης με την τιμή που θέλουμε να επιστρέψουμε στο κύριο πρόγραμμα πχ **ΑΘΡΟΙΣΜΑ ← Α + Β**.

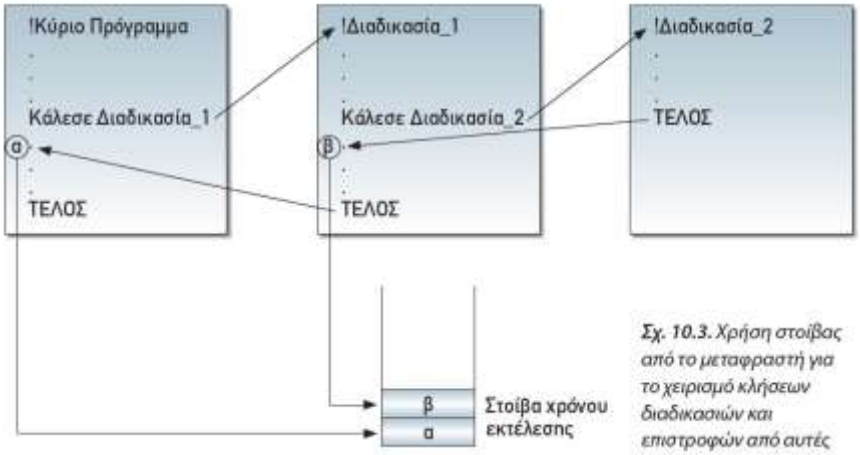
Ο τρόπος εκτέλεσης της συνάρτησης κατά τα υπόλοιπα είναι ίδιος με της διαδικασίας.

Μία συνάρτηση πρέπει να έχει:

1) ***να έχει τουλάχιστον μία παράμετρο, ώστε να μοιάζει με μαθηματική συνάρτηση***
2) ***να μας επιστρέφει μόνο μία τιμή (μέσα από το όνομα της)***
3) ***να μην έχει ΔΙΑΒΑΣΕ ή ΓΡΑΨΕ μέσα στην συνάρτηση***

ΧΡΗΣΗ ΣΤΟΙΒΑΣ ΣΤΗΝ ΚΛΗΣΗ ΥΠΟΠΡΟΓΡΑΜΜΑΤΩΝ

Όταν μία διαδικασία ή μία συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η επόμενη θέση του κύριου προγράμματος που λέγεται και ***διεύθυνση επιστροφής*** αποθηκεύεται σε μία στοίβα που ονομάζεται ***στοίβα χρόνου εκτέλεσης***. Μετά την ολοκλήρωση του υποπρογράμματος απωθείται από την στοίβα η διεύθυνση επιστροφής και το πρόγραμμα συνεχίζει την εκτέλεσή του από εκείνο το σημείο του κύριου προγράμματος.



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

Εμβέλεια μεταβλητών-σταθερών

Το **τμήμα του προγράμματος που ισχύουν οι μεταβλητές** λέγεται εμβέλεια (scope) μεταβλητών.

Απεριόριστη εμβέλεια

Σύμφωνα με αυτή την αρχή όλες οι **μεταβλητές και όλες οι σταθερές** είναι γνωστές και μπορούν να **χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, άσχετα που δηλώθηκαν**. Όλες οι μεταβλητές είναι καθολικές.

Η απεριόριστη εμβέλεια **καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων**, δημιουργεί πολλά προβλήματα και τελικά είναι αδύνατη για μεγάλα προγράμματα με πολλά υποπρογράμματα, αφού ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.

Περιορισμένη εμβέλεια

Η περιορισμένη εμβέλεια υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα. **Όλες οι μεταβλητές είναι τοπικές, ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν.** Στη ΓΛΩΣΣΑ έχουμε περιορισμένη εμβέλεια.

Τα **πλεονεκτήματα της** περιορισμένης εμβέλειας είναι η απόλυτη **αυτονομία όλων των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα**, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.

Μερικώς περιορισμένη εμβέλεια

Σύμφωνα με αυτή την αρχή **άλλες μεταβλητές είναι τοπικές και άλλες καθολικές**. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές.

Η μερικώς περιορισμένη εμβέλεια προσφέρει μερικά **πλεονεκτήματα στον πεπειραμένο προγραμματιστή**, αλλά για τον **αρχάριο περιπλέκει το πρόγραμμα** δυσκολεύοντας την ανάπτυξή του.

ΛΥΜΕΝΗ ΑΣΚΗΣΗ ΧΩΡΙΣ ΚΑΙ ΜΕ ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ (χωρίς πίνακες)

1) Να φτιαχτεί πρόγραμμα που θα διαβάζει έναν αριθμό και θα μας εμφανίζει στην οθόνη την απόλυτη τιμή του α) χωρίς υποπρόγραμμα β) με διαδικασία που υπολογίζει και επιστρέφει την απόλυτη τιμή του αριθμού που εισαγάγαμε και γ) με συνάρτηση που υπολογίζει και επιστρέφει την απόλυτη τιμή του αριθμού που εισαγάγαμε.

ΛΥΣΗ ΧΩΡΙΣ ΥΠΟΠΡΟΓΡΑΜΜΑ	ΛΥΣΗ ΜΕ ΔΙΑΔΙΚΑΣΙΑ	ΛΥΣΗ ΜΕ ΣΥΝΑΡΤΗΣΗ
ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ: Χ, Υ ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ ΕΝΑΝ ΑΡΙΘΜΟ' ΔΙΑΒΑΣΕ Χ **ΑΝ Χ >=0 ΤΟΤΕ** **Υ ← Χ** **ΑΛΛΙΩΣ** **Υ ← -Χ** **ΤΕΛΟΣ_ΑΝ** ΓΡΑΨΕ 'Η ΑΤ ΤΟΥ ',Χ,' ΕΙΝΑΙ ',Υ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ	ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ: Χ, Υ ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ ΕΝΑΝ ΑΡΙΘΜΟ' ΔΙΑΒΑΣΕ Χ **ΚΑΛΕΣΕ ΔΙΑΔ1(Χ, Υ)** ΓΡΑΨΕ 'Η ΑΤ ΤΟΥ ',Χ,' ΕΙΝΑΙ ',Υ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ **ΔΙΑΔΙΑΚΑΣΙΑ ΔΙΑΔ1 (Α, Β)** ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ: Α,Β ΑΡΧΗ **ΑΝ Α >=0 ΤΟΤΕ** **Β ← Α** **ΑΛΛΙΩΣ** **Β ← -Α** **ΤΕΛΟΣ_ΑΝ** ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ	ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ: Χ, Υ ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ ΕΝΑΝ ΑΡΙΘΜΟ' ΔΙΑΒΑΣΕ Χ **Υ ← ΣΥΝ1(Χ)** ΓΡΑΨΕ 'Η ΑΤ ΤΟΥ ',Χ,' ΕΙΝΑΙ ',Υ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ **ΣΥΝΑΡΤΗΣΗ ΣΥΝ1 (Α):ΠΡΑΓΜΑΤΙΚΗ** ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ: Α ΑΡΧΗ **ΑΝ Α >=0 ΤΟΤΕ** **ΣΥΝ1 ← Α** **ΑΛΛΙΩΣ** **ΣΥΝ1 ← -Α** **ΤΕΛΟΣ_ΑΝ** ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

Στο ΚΠ (Κύριο Πρόγραμμα) έντονες είναι οι εντολές που μπήκαν στα υποπρογράμματα.

Η Διαδικασία έχει 2 παραμέτρους, μία για να δέχεται τον αριθμό που διαβάσαμε στο ΚΠ και μία για να επιστρέφει την απόλυτη τιμή του.

Η Συνάρτηση έχει μόνο μία παράμετρο αφού στέλνουμε μόνο μία τιμή, ενώ το αποτέλεσμα της το επιστρέφει με το όνομα της.

Παράδειγμα διαδικασίας και συνάρτησης στο ίδιο πρόγραμμα

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ_ΔΙΑΔΙΚΑΣΙΑΣ2
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ :  Χ, Υ, ΜΑΧ
ΑΡΧΗ
    ΓΡΑΨΕ ΄ΔΩΣΕ 2 ΑΡΙΘΜΟΥΣ : ΄
    ΔΙΑΒΑΣΕ Χ, Υ
    ΓΡΑΨΕ Χ, Υ
    ΚΑΛΕΣΕ ΑΝΤΙΜΕΤΑΘΕΣΕ ( Χ,  Υ )
    ΓΡΑΨΕ ΄ΟΙ 2 ΑΡΙΘΜΟΙ ΜΕΤΑ ΤΗΝ ΑΝΤΙΜΕΤΑΘΕΣΗ  ΕΙΝΑΙ : ΄,  Χ, Υ
    ΜΑΧ ← ΜΕΓΙΣΤΟΣ (Χ, Υ)
    ΓΡΑΨΕ ΄Ο ΜΕΓΑΛΥΤΕΡΟΣ ΑΡΙΘΜΟΣ ΑΠΟ ΤΟΥΣ 2 ΕΙΝΑΙ Ο : ΄,  ΜΑΧ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ


ΔΙΑΔΙΚΑΣΙΑ ΑΝΤΙΜΕΤΑΘΕΣΕ (Α,  Β)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Α, Β, ΤΕΜΡ
ΑΡΧΗ
    ΤΕΜΡ ← Α
    Α ← Β
    Β ← ΤΕΜΡ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


ΣΥΝΑΡΤΗΣΗ ΜΕΓΙΣΤΟΣ (C, D) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : C, D

ΑΡΧΗ
    ΑΝ C >= D ΤΟΤΕ
        ΜΕΓΙΣΤΟΣ ← C
    ΑΛΛΙΩΣ
        ΜΕΓΙΣΤΟΣ ← D
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα διαδικασιών με πίνακες. Ο πίνακας στο κύριο πρόγραμμα είναι ο Α, ενώ στις διαδικασίες ΠΙΝ1, ΠΙΝ2, ΠΙΝ3, όπου γίνεται και η επεξεργασία του. Κάθε φορά στέλνουμε τον πίνακα Α στις διαδικασίες όπου γίνεται διάβασμα των τιμών του, εμφάνιση ή ταξινόμηση. Επίσης το μέγεθος του πίνακα Α είναι ίδιο υποχρεωτικά με των πινάκων που δέχονται και επιστρέφουν τις τιμές των στοιχείων του.

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ_ΠΙΝΑΚΕΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Α [ 100 ]
ΑΡΧΗ
    ΓΡΑΨΕ ΄ΘΑ ΓΙΝΕΙ ΕΙΣΑΓΩΓΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΠΙΝΑΚΑ΄
    ΚΑΛΕΣΕ ΔΙΑΒΑΣΕ_ΠΙΝΑΚΑ (Α)
    ΓΡΑΨΕ ΄ΘΑ ΓΙΝΕΙ ΕΜΦΑΝΙΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΠΙΝΑΚΑ΄
    ΚΑΛΕΣΕ ΓΡΑΨΕ_ΠΙΝΑΚΑ (Α)
    ΚΑΛΕΣΕ ΤΑΞΙΝΟΜΗΣΕ_ΠΙΝΑΚΑ (Α)
    ΓΡΑΨΕ ΄Ο ΠΙΝΑΚΑΣ ΜΕΤΑ ΤΗΝ ΤΑΞΙΝΟΜΗΣΗ ΕΙΝΑΙ΄
    ΚΑΛΕΣΕ ΓΡΑΨΕ_ΠΙΝΑΚΑ (Α)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ


ΔΙΑΔΙΚΑΣΙΑ ΔΙΑΒΑΣΕ_ΠΙΝΑΚΑ(ΠΙΝ)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Ι, ΠΙΝ [ 100 ]
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟ ΄ , Ι , ΄ ΣΤΟΙΧΕΊΟ ΤΟΥ ΠΙΝΑΚΑ : ΄
        ΔΙΑΒΑΣΕ ΠΙΝ [ Ι ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


ΔΙΑΔΙΚΑΣΙΑ ΓΡΑΨΕ_ΠΙΝΑΚΑ(ΠΙΝ1)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Ι, ΠΙΝ1 [ 100 ]
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΄ΤΟ ΄, Ι , ΄ ΣΤΟΙΧΕΊΟ ΤΟΥ ΠΙΝΑΚΑ ΕΙΝΑΙ: ΄, ΠΙΝ1[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

ΔΙΑΔΙΚΑΣΙΑ ΤΑΞΙΝΟΜΗΣΕ_ΠΙΝΑΚΑ(ΠΙΝ2)

```
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Ι, Κ, ΤΕΜΡ, ΠΙΝ2 [ 100 ]
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 100
        ΓΙΑ Κ ΑΠΟ 100 ΜΕΧΡΙ Ι ΜΕ ΒΗΜΑ -1
            ΑΝ ΠΙΝ2 [ Κ ] < ΠΙΝ2 [ Κ-1 ] ΤΟΤΕ
                ΤΕΜΡ ← ΠΙΝ2 [ Κ ]
                ΠΙΝ2 [ Κ ] ←  ΠΙΝ2 [ Κ-1 ]
                ΠΙΝ2 [ Κ-1 ] ← ΤΕΜΡ
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

ΛΥΜΕΝΗ ΑΣΚΗΣΗ ΧΩΡΙΣ ΚΑΙ ΜΕ ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ (με μονοδιάστατο πίνακα)

2) Να φτιαχτεί πρόγραμμα που θα διαβάζει τις ηλικίες 30 ανθρώπων και θα μας εμφανίζει στην οθόνη για κάθε έναν μήνυμα με το αν είναι "ενήλικος" ή "ανήλικος" α) χωρίς υποπρόγραμμα β) με διαδικασία που καλείται μία φορά και επιστρέφει μέσα σε πίνακα τον χαρακτηρισμό της κάθε ηλικίας που εισαγάγαμε και γ) με συνάρτηση που την καλούμε 30 φορές (αφού η συνάρτηση επιστρέφει μόνο μία τιμή) και βρίσκει και επιστρέφει κάθε φορά τον χαρακτηρισμό της κάθε ηλικίας που εισαγάγαμε.

ΛΥΣΗ ΧΩΡΙΣ ΥΠΟΠΡΟΓΡΑΜΜΑ	ΛΥΣΗ ΜΕ ΔΙΑΔΙΚΑΣΙΑ	ΛΥΣΗ ΜΕ ΣΥΝΑΡΤΗΣΗ
ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ : ΗΛ[30] ΑΚΕΡΑΙΕΣ : Ι ΧΑΡΑΚΤΗΡΕΣ : ΧΑΡ[30] ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ 30 ΗΛΙΚΙΕΣ' ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΔΙΑΒΑΣΕ ΗΛ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ **ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30** **ΑΝ ΗΛ[Ι] < 18 ΤΟΤΕ** **ΧΑΡ[Ι] ← 'ΑΝΗΛΙΚΟΣ'** **ΑΛΛΙΩΣ** **ΧΑΡ[Ι] ← 'ΕΝΗΛΙΚΟΣ'** **ΤΕΛΟΣ_ΑΝ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΓΡΑΨΕ ΧΑΡ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ	ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ : ΗΛ[30] ΑΚΕΡΑΙΕΣ : Ι ΧΑΡΑΚΤΗΡΕΣ : ΧΑΡ[30] ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ 30 ΗΛΙΚΙΕΣ' ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΔΙΑΒΑΣΕ ΗΛ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ **ΚΑΛΕΣΕ ΔΙΑΔ1(ΗΛ, ΧΑΡ)** ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΓΡΑΨΕ ΧΑΡ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ ΔΙΑΔΙΑΚΑΣΙΑ ΔΙΑΔ1(ΗΛ, ΧΑΡ) ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ : ΗΛ[30] ΑΚΕΡΑΙΕΣ : Ι ΧΑΡΑΚΤΗΡΕΣ : ΧΑΡ[30] ΑΡΧΗ **ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30** **ΑΝ ΗΛ[Ι] < 18 ΤΟΤΕ** **ΧΑΡ[Ι] ← 'ΑΝΗΛΙΚΟΣ'** **ΑΛΛΙΩΣ** **ΧΑΡ[Ι] ← 'ΕΝΗΛΙΚΟΣ'** **ΤΕΛΟΣ_ΑΝ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ	ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ : ΗΛ[30] ΑΚΕΡΑΙΕΣ : Ι ΧΑΡΑΚΤΗΡΕΣ : ΧΑΡ[30] ΑΡΧΗ ΓΡΑΨΕ 'ΔΩΣΕ 30 ΗΛΙΚΙΕΣ' ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΔΙΑΒΑΣΕ ΗΛ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ **ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30** **ΧΑΡ[Ι] ← ΣΥΝ1(ΗΛ, Ι)** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΓΡΑΨΕ ΧΑΡ[Ι] ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΥΝΑΡΤΗΣΗ ΣΥΝ1 (ΗΛ, Ι):ΧΑΡΑΚΤΗΡΑ ΜΕΤΑΒΛΗΤΕΣ ΠΡΑΓΜΑΤΙΚΕΣ : ΗΛ[30] ΑΚΕΡΑΙΕΣ : Ι ΑΡΧΗ **ΑΝ ΗΛ[Ι] < 18 ΤΟΤΕ** **ΣΥΝ1 ← 'ΑΝΗΛΙΚΟΣ'** **ΑΛΛΙΩΣ** **ΣΥΝ1 ← 'ΕΝΗΛΙΚΟΣ'** **ΤΕΛΟΣ_ΑΝ** ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

Τη Διαδικασία την καλούμε μία φορά και επιστρέφουμε και τους 30 χαρακτηρισμούς μέσω πίνακα. Δηλαδή μέσα στη διαδικασία βάζουμε και την ΓΙΑ οπότε ελέγχουμε όλες τις ηλικίες.

Αντίθετα τη συνάρτηση την καλούμε 30 φορές, αφού επιστρέφει μία τιμή κάθε φορά. Δηλαδή η ΓΙΑ παραμένει στο ΚΠ για να καλέσουμε τη Συνάρτηση 30 φορές και δεν τη χρησιμοποιούμε και μέσα στη Συνάρτηση, αλλά γράφουμε στη Συνάρτηση μόνο τις εντολές που είχε μέσα της η ΓΙΑ στο ΚΠ.

ΠΑΡΑΔΕΙΓΜΑ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ ΜΕ ΟΛΟΥΣ ΤΟΥΣ ΤΡΟΠΟΥΣ ΜΕ ΔΙΣΔΙΑΣΤΑΤΟ ΠΙΝΑΚΑ

Να φτιαχτεί πρόγραμμα που θα διαβάζει για 100 υπαλλήλους το επώνυμο και τις μηνιαίες αποδοχές που είχαν τον προηγούμενο χρόνο. Στη συνέχεια μέσω υποπρογράμματος θα βρίσκει και θα επιστρέφει την μικρότερη μηνιαία είσπραξη του κάθε υπαλλήλου και θα την εμφανίζει στο κύριο πρόγραμμα μαζί με το επώνυμο του.

Η εύρεση του μικρότερου να γίνεται για κάθε υπάλληλο ξεχωριστά. Μπορεί να γίνει με τρεις τρόπους.

Α) με συνάρτηση για κάθε ένα ξεχωριστά

```
ΠΡΟΓΡΑΜΜΑ ΕΛΑΧΙΣΤΟΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΧΑΡΑΚΤΗΡΕΣ: ΕΠ[100]
    ΑΚΕΡΑΙΕΣ: Ι, Κ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12], ΕΛ[100]
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟ ΄, Ι ,΄ ΕΠΩΝΥΜΟ΄
        ΔΙΑΒΑΣΕ ΕΠ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι]
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΓΡΑΨΕ ΄ΔΩΣΕ ΓΙΑ΄, Κ ,΄ΜΗΝΑ΄
            ΔΙΑΒΑΣΕ ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΕΛ[Ι]← ΕΛΑΧ(ΕΙΣ, Ι)
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι] , ΕΛ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ ΕΛΑΧ(ΕΙΣ, Ι):ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12],ΜΙΚ
    ΑΚΕΡΑΙΕΣ: Ι, Κ
ΑΡΧΗ
    ΜΙΚ ← ΕΙΣ[Ι, 1]
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
        ΑΝ ΕΙΣ[Ι, Κ] < ΜΙΚ ΤΟΤΕ
            ΜΙΚ ← ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΕΛΑΧ ← ΜΙΚ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Β) με διαδικασία για κάθε ένα ξεχωριστά

```
ΠΡΟΓΡΑΜΜΑ ΕΛΑΧΙΣΤΟΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΧΑΡΑΚΤΗΡΕΣ: ΕΠ[100]
    ΑΚΕΡΑΙΕΣ: Ι, Κ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12], ΕΛ[100], ΜΙΚ
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟ ΄, Ι ,΄ ΕΠΩΝΥΜΟ΄
        ΔΙΑΒΑΣΕ ΕΠ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι]
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΓΡΑΨΕ ΄ΔΩΣΕ ΓΙΑ΄, Κ ,΄ΜΗΝΑ΄
            ΔΙΑΒΑΣΕ ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΚΑΛΕΣΕ ΕΛΑΧ(ΕΙΣ, Ι, ΜΙΚ)
        ΕΛ[Ι] ← ΜΙΚ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι] , ΕΛ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΕΛΑΧ(ΕΙΣ, Ι, ΜΙΚ)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12],ΜΙΚ
    ΑΚΕΡΑΙΕΣ: Ι, Κ
ΑΡΧΗ
    ΜΙΚ ← ΕΙΣ[Ι, 1]
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
        ΑΝ ΕΙΣ[Ι, Κ] < ΜΙΚ ΤΟΤΕ
            ΜΙΚ ← ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Γ) με διαδικασία μέσω άλλου πίνακα

```
ΠΡΟΓΡΑΜΜΑ ΕΛΑΧΙΣΤΟΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΧΑΡΑΚΤΗΡΕΣ: ΕΠ[100]
    ΑΚΕΡΑΙΕΣ: Ι, Κ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12], ΕΛ[100],
                        ΜΗΝ[12], ΜΙΚ
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ 'ΔΩΣΕ ΤΟ ΄, Ι ,΄ ΕΠΩΝΥΜΟ΄
        ΔΙΑΒΑΣΕ ΕΠ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι]
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΓΡΑΨΕ 'ΔΩΣΕ ΓΙΑ΄, Κ ,΄ΜΗΝΑ ΄
            ΔΙΑΒΑΣΕ ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΜΗΝ[Κ] ← ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΚΑΛΕΣΕ ΕΛΑΧ(ΜΗΝ, ΜΙΚ)
        ΕΛ[Ι] ← ΜΙΚ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι] , ΕΛ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΕΛΑΧ(ΜΗΝ, ΜΙΚ)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΜΗΝ[12],ΜΙΚ
    ΑΚΕΡΑΙΕΣ: Ι, Κ
ΑΡΧΗ
    ΜΙΚ ← ΜΗΝ[1]
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
        ΑΝ ΜΗΝ[Κ] < ΜΙΚ ΤΟΤΕ
            ΜΙΚ ← ΜΗΝ[Κ]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Επιστροφή όλων των ΜΙΝ μέσω πίνακα

```
ΠΡΟΓΡΑΜΜΑ ΕΛΑΧΙΣΤΟΣ
ΜΕΤΑΒΛΗΤΕΣ
    ΧΑΡΑΚΤΗΡΕΣ: ΕΠ[100]
    ΑΚΕΡΑΙΕΣ: Ι, Κ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12], ΕΛ[100], ΜΙΚ
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΄ΔΩΣΕ ΤΟ ΄, Ι ,΄ ΕΠΩΝΥΜΟ΄
        ΔΙΑΒΑΣΕ ΕΠ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι]
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΓΡΑΨΕ ΄ΔΩΣΕ ΓΙΑ΄, Κ ,΄ΜΗΝΑ΄
            ΔΙΑΒΑΣΕ ΕΙΣ[Ι, Κ]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΚΑΛΕΣΕ ΕΛΑΧ(ΕΙΣ, ΕΛ)
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΓΡΑΨΕ ΕΠ[Ι] , ΕΛ[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΕΛΑΧ(ΕΙΣ, ΕΛ)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: ΕΙΣ[100,12], ΕΛ[100], ΜΙΚ
    ΑΚΕΡΑΙΕΣ: Ι, Κ
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
        ΜΙΚ ← ΕΙΣ[Ι, 1]
        ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 12
            ΑΝ ΕΙΣ[Ι, Κ] < ΜΙΚ ΤΟΤΕ
                ΜΙΚ ← ΕΙΣ[Ι, Κ]
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΕΛ[Ι] ← ΜΙΚ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Ενότητα 1. Δομές Δεδομένων και Αλγόριθμοι

Στοίβα (stack), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία.

Οι κύριες λειτουργίες σε μια στοίβα είναι δύο:

1. **Η ώθηση** (push) στοιχείου στην κορυφή της στοίβας.

Στη διαδικασία της ώθησης ελέγχουμε αν η στοίβα είναι γεμάτη.

Στην περίπτωση που προσπαθήσουμε να «προσθέσουμε» ένα στοιχείο σε μια ήδη γεμάτη στοίβα, έχουμε **υπερχείλιση** (overflow) της στοίβας.

2. **Η απώθηση** (pop) στοιχείου από τη στοίβα.

Στη διαδικασία της απώθησης ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα.

Στην περίπτωση που προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία κενή στοίβα, έχουμε **υποχείλιση** (underflow) της στοίβας.

Σε μια κενή στοίβα/πίνακα θεωρούμε ότι η αρχική τιμή της μεταβλητής top είναι μηδέν (top←0).

Η μεταβλητή top είναι η μεταβλητή που δείχνει τη θέση που τοποθετήθηκε το τελευταίο στοιχείο στη στοίβα/πίνακα (δηλ. δείχνει την κορυφή της στοίβας).

Κατά την ώθηση ενός στοιχείου στη στοίβα (εισαγωγή ενός στοιχείου στον πίνακα), 1) πρώτα αυξάνεται η τιμή της μεταβλητής top κατά ένα, δηλ. top←top+1, και στη συνέχεια 2) γίνεται η ώθηση του στοιχείου στην κορυφή της στοίβας.

Κατά την απώθηση ενός στοιχείου από τη στοίβα, 1) γίνεται εξαγωγή στοιχείου από τον πίνακα και στη συνέχεια 2) μειώνεται η τιμή της μεταβλητής top κατά ένα, δηλ. top ← top-1. Στην απώθηση δε διαγράφεται το στοιχείο, στην πραγματικότητα δε γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα. Απλώς ο δείκτης κορυφή δείχνει στην προηγούμενη θέση.

Σε κάθε λειτουργία στη στοίβα έχω 2 περιπτώσεις:
1) Πότε γίνεται
2) Πότε δεν γίνεται

```
ΠΡΟΓΡΑΜΜΑ ΣΤΟΙΒΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ΤΟΠ, ΣΤ[10], ΕΠΙΛ, ΤΙΜΗ
ΑΡΧΗ
  ΤΟΠ <- 0
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

    !ΜΕΝΟΥ ΕΠΙΛΟΓΩΝ
    ΓΡΑΨΕ '1. ΕΠΙΒΙΒΑΣΗ'
    ΓΡΑΨΕ '2. ΑΠΟΒΙΒΑΣΗ'
    ΓΡΑΨΕ '3. ΕΞΟΔΟΣ'
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ ΕΠΙΛ
    ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ >= 1 ΚΑΙ ΕΠΙΛ <= 3

    ΑΝ ΕΠΙΛ=1 ΤΟΤΕ                          ! ΩΘΗΣΗ
       ΔΙΑΒΑΣΕ ΤΙΜΗ
       ΑΝ ΤΟΠ < 10 ΤΟΤΕ                        ! ΕΧΕΙ ΧΩΡΟ Η ΣΤΟΙΒΑ
         ΤΟΠ <- ΤΟΠ+1
         ΣΤ[ΤΟΠ] <- ΤΙΜΗ
       ΑΛΛΙΩΣ
         ΓΡΑΨΕ 'ΓΕΜΑΤΗ ΥΠΕΡΧΕΙΛΗΣΗ'
       ΤΕΛΟΣ_ΑΝ

    ΑΛΛΙΩΣ_ΑΝ ΕΠΙΛ = 2 ΤΟΤΕ                 !ΑΠΩΘΗΣΗ
       ΑΝ ΤΟΠ > 0 ΤΟΤΕ                         ! ΕΧΕΙ ΤΙΜΕΣ Η ΣΤΟΙΒΑ
         ΓΡΑΨΕ ΣΤ[ΤΟΠ]
         ΤΟΠ <- ΤΟΠ -1
       ΑΛΛΙΩΣ
         ΓΡΑΨΕ 'ΑΔΕΙΑ-ΥΠΟΧΕΙΛΗΣΗ'
       ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ

  ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 3                       !ΤΕΡΜΑΤΙΣΜΟΣ-ΕΞΟΔΟΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ουρά (Queue), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα.
Οι **κύριες λειτουργίες** που εκτελούνται σε μια ουρά είναι δύο:

1. Η **εισαγωγή** ενός νέου στοιχείου γίνεται από το πίσω άκρο της ουράς και η τιμή της μεταβλητής rear αλλάζει ως εξής:
   Α) πρώτα αυξάνουμε τον δείκτη rear κατά ένα rear ← rear+1
   Β) και μετά εισάγουμε το στοιχείο στον πίνακα.

2. Η **εξαγωγή** ενός στοιχείου γίνεται από το εμπρός άκρο της ουράς και η τιμή της μεταβλητής front αλλάζει ως εξής:
   Α) πρώτα γίνεται παίρνω το στοιχείο που δείχνει ο δείκτης front
   Β) αυξάνεται ο δείκτης front κατά ένα front ← front +1 (δείχνει στην επόμενη θέση του πίνακα) χωρίς στην πραγματικότητα να γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα (χωρίς να διαγράφεται κάποιο στοιχείο).

```
ΠΡΟΓΡΑΜΜΑ ΟΥΡΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: FRONT, REAR, ΟΥΡ[10], ΕΠΙΛ, ΤΙΜΗ
ΑΡΧΗ

  FRONT <- 0                                  !ΑΡΧΙΚΟΠΟΙΗΣΗ ΔΕΙΚΤΩΝ
  REAR <- 0

  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

    ΓΡΑΨΕ '1. ΕΙΣΑΓΩΓΗ'                       !ΜΕΝΟΥ ΕΠΙΛΟΓΩΝ
    ΓΡΑΨΕ '2. ΕΞΑΓΩΓΗ'
    ΓΡΑΨΕ '3. ΕΞΟΔΟΣ'
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ ΕΠΙΛ
    ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ >= 1 ΚΑΙ ΕΠΙΛ <= 3

    ΑΝ ΕΠΙΛ = 1 ΤΟΤΕ                          ! ΕΙΣΑΓΩΓΗ
      ΔΙΑΒΑΣΕ ΤΙΜΗ
      ΑΝ REAR = 10 ΤΟΤΕ                       ! ΓΕΜΑΤΗ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ   ! ΑΔΕΙΑ
        FRONT <- 1
        REAR <- 1
        ΟΥΡ[REAR] <- ΤΙΜΗ
      ΑΛΛΙΩΣ
        REAR <- REAR + 1
        ΟΥΡ[REAR] <- ΤΙΜΗ
      ΤΕΛΟΣ_ΑΝ

    ΑΛΛΙΩΣ_ΑΝ ΕΠΙΛ = 2 ΤΟΤΕ                   ! ΕΞΑΓΩΓΗ

      ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ           ! ΑΔΕΙΑ
        ΓΡΑΨΕ 'ΑΔΕΙΑ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = REAR ΤΟΤΕ             ! ΕΧΕΙ ΜΟΝΟ ΜΙΑ ΤΙΜΗ
        ΓΡΑΨΕ ΟΥΡ[FRONT]
        FRONT <- 0
        REAR <- 0
      ΑΛΛΙΩΣ
        ΓΡΑΨΕ ΟΥΡ[FRONT]
        FRONT <- FRONT + 1
      ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΑΝ

  ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 3                        !ΤΕΡΜΑΤΙΣΜΟΣ-ΕΞΟΔΟΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Σε κάθε λειτουργία στην ουρά έχω 3 περιπτώσεις:
1) Πότε δεν γίνεται
2) Η ειδική περίπτωση
3) Η γενική περίπτωση

ΤΡΟΠΟΠΟΙΗΣΕΙΣ ΚΑΝΟΝΙΚΗΣ ΟΥΡΑΣ

Α) **Τροποποίηση** σε κανονική ουρά, ώστε κατά την εισαγωγή, στην περίπτωση που είναι γεμάτη η ουρά να **πηγαίνει όλες τις υπάρχουσες τιμές μία θέση αριστερά** και να βάζει στην τελευταία θέση τη νέα τιμή

```
ΑΝ ΕΠΙΛ = 1 ΤΟΤΕ                  ! ΕΙΣΑΓΩΓΗ
    ΔΙΑΒΑΣΕ Χ
    ΑΝ REAR = 10 ΤΟΤΕ              ! ΓΕΜΑΤΗ ΟΥΡΑ ?
      ΑΝ FRONT = 1 ΤΟΤΕ ! ΟΛΕΣ ΟΙ ΘΕΣΕΙΣ ΤΟΥ ΠΙΝΑΚΑ ΠΕΡΙΕΧΟΥΝ ΤΙΜΕΣ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ ΟΥΡΑ'
      ΑΛΛΙΩΣ
        ΓΙΑ Ι ΑΠΟ FRONT ΜΕΧΡΙ REAR!ΤΙΜΕΣ ΤΟΥ ΠΙΝΑΚΑ ΜΙΑ ΘΕΣΗ ΑΡΙΣΤΕΡΑ
            ΟΥ[Ι - 1] <- ΟΥ[Ι]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        FRONT <- FRONT - 1
        ΟΥ[REAR] <- Χ
      ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ_ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ
```

Β) **Τροποποίηση** σε κανονική ουρά, ώστε κατά την εισαγωγή, στην περίπτωση που είναι γεμάτη η ουρά να **πηγαίνει όλες τις υπάρχουσες τιμές στις πρώτες θέσεις του πίνακα** και τη νέα τιμή αμέσως μετά

```
ΑΝ ΕΠΙΛ = 1 ΤΟΤΕ                  ! ΕΙΣΑΓΩΓΗ
    ΔΙΑΒΑΣΕ ΤΙΜΗ
    ΑΝ REAR = 10 ΤΟΤΕ              ! ΓΕΜΑΤΗ ΟΥΡΑ ?
      ΑΝ FRONT = 1 ΤΟΤΕ            ! ΟΛΕΣ ΟΙ ΘΕΣΕΙΣ ΤΟΥ ΠΙΝΑΚΑ ΠΕΡΙΕΧΟΥΝ ΤΙΜΕΣ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ ΟΥΡΑ'
      ΑΛΛΙΩΣ
        ΠΛ <- 0
        ΓΙΑ Ι ΑΠΟ FRONT ΜΕΧΡΙ REAR   ! ΤΙΜΕΣ ΤΟΥ ΠΙΝΑΚΑ ΣΤΙΣ ΠΡΩΤΕΣ ΘΕΣΕΙΣ
          ΠΛ <- ΠΛ + 1
          ΟΥ[ΠΛ] <- ΟΥ[Ι]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        FRONT <- 1
        REAR <- ΠΛ          ! ΠΛ = ΠΛΗΘΟΣ ΣΤΟΙΧΕΙΩΝ
        REAR <- REAR + 1
        ΟΥ[REAR ] <- ΤΙΜΗ
      ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ_ΑΝ FRONT = 0 ΚΑΙ REAR  = 0 ΤΟΤΕ
```

Υλοποίηση λειτουργιών Ουράς σε φούρνο

```
ΠΡΟΓΡΑΜΜΑ ΟΥΡΑ_ΦΟΥΡΝΟΣ

ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ΕΠΙΛ, ΟΥ[10], Ι, REAR, Χ

ΑΡΧΗ
  REAR <- 0                    ! ΔΕΝ ΧΡΕΙΑΖΕΤΑΙ FRONT, ΕΙΝΑΙ ΠΑΝΤΑ 1
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ '1. ΕΙΣΑΓΩΓΗ'
    ΓΡΑΨΕ '2. ΕΞΑΓΩΓΗ'
    ΓΡΑΨΕ '3. ΕΞΟΔΟΣ'
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ ΕΠΙΛ
    ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 1 Η ΕΠΙΛ = 2 Η ΕΠΙΛ = 3

    ΑΝ ΕΠΙΛ = 1 ΤΟΤΕ           ! ΕΙΣΑΓΩΓΗ
      ΔΙΑΒΑΣΕ ΤΙΜΗ
      ΑΝ REAR = 10 ΤΟΤΕ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ'
      ΑΛΛΙΩΣ
        REAR <- REAR + 1
        ΟΥ[REAR] <- ΤΙΜΗ
      ΤΕΛΟΣ_ΑΝ

    ΑΛΛΙΩΣ_ΑΝ ΕΠΙΛ = 2 ΤΟΤΕ    ! ΕΞΑΓΩΓΗ
      ΑΝ REAR = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'ΑΔΕΙΑ'
      ΑΛΛΙΩΣ
        ΓΡΑΨΕ ΟΥ[1]
        ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ REAR ! ΠΑΜΕ ΟΛΑ ΤΑ ΣΤΟΙΧΕΙΑ ΜΙΑ ΘΕΣΗ ΜΠΡΟΣΤΑ
          ΟΥ[Ι - 1] <- ΟΥ[Ι]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        REAR <- REAR - 1
      ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ

  ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 3         ! ΤΕΛΟΣ ΜΕΝΟΥ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Σε κάθε λειτουργία στην ουρά σε φούρνο έχω 2 περιπτώσεις:

1) Πότε δεν γίνεται
2) Γενική περίπτωση

Υλοποίηση λειτουργιών Κυκλική Ουράς

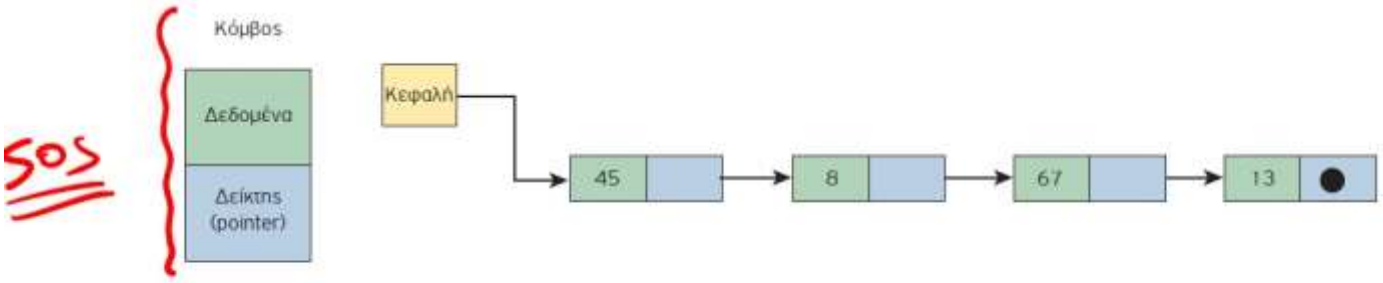
```
ΠΡΟΓΡΑΜΜΑ ΚΥΚΛΙΚΗ_ΟΥΡΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ΕΠΙΛ, ΟΥ[10], FRONT, REAR, Χ
ΑΡΧΗ
  FRONT <- 0
  REAR <- 0
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ '1. ΕΙΣΑΓΩΓΗ'
    ΓΡΑΨΕ '2. ΕΞΑΓΩΓΗ'
    ΓΡΑΨΕ '3. ΕΞΟΔΟΣ'
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ ΕΠΙΛ
    ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 1 Η ΕΠΙΛ = 2 Η ΕΠΙΛ = 3
    ΑΝ ΕΠΙΛ = 1 ΤΟΤΕ
      ΔΙΑΒΑΣΕ ΤΙΜΗ
      ΑΝ FRONT = 1 ΚΑΙ REAR = 10 Η REAR = FRONT - 1 ΤΟΤΕ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ
        FRONT <- 1
        REAR <- 1
        ΟΥ[REAR] <- ΤΙΜΗ
      ΑΛΛΙΩΣ
        ΑΝ REAR = 10 ΤΟΤΕ
          REAR <- 1
        ΑΛΛΙΩΣ
          REAR <- REAR + 1
        ΤΕΛΟΣ_ΑΝ
        ΟΥ[REAR] <- ΤΙΜΗ
      ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ_ΑΝ ΕΠΙΛ = 2 ΤΟΤΕ
      ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'ΑΔΕΙΑ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = REAR ΤΟΤΕ
        ΓΡΑΨΕ ΟΥ[FRONT]
        REAR <- 0
        FRONT <- 0
      ΑΛΛΙΩΣ
        ΓΡΑΨΕ ΟΥ[FRONT]
        ΑΝ FRONT = 10 ΤΟΤΕ
          FRONT <- 1
        ΑΛΛΙΩΣ
          FRONT <- FRONT + 1
        ΤΕΛΟΣ_ΑΝ
      ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
  ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 3
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΥΛΟΠΟΙΗΣΗ ΟΥΡΑΣ ΠΟΥ ΑΔΕΙΑΖΕΙ ΣΤΗΝ ΕΞΟΔΟ

```
ΠΡΟΓΡΑΜΜΑ ΟΥΡΑ_ΑΔΕΙΑΖΕΙ_ΣΤΗΝ_ΕΞΟΔΟ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: FRONT, REAR, ΠΛ, ΠΛΑ, ΜΑΧ
  ΧΑΡΑΚΤΗΡΕΣ: ΟΝ[10], Χ, ΕΠΙΛ
  ΛΟΓΙΚΕΣ: ΤΕΛΟΣ
ΑΡΧΗ
  FRONT <- 0
  REAR <- 0
  ΤΕΛΟΣ <- ΨΕΥΔΗΣ

  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

    ΑΝ ΤΕΛΟΣ = ΨΕΥΔΗΣ ΤΟΤΕ    !ΕΛΕΓΧΟΣ ΑΝ ΘΑ ΕΜΦΑΝΙΖΕΤΑΙ ΤΟ ΜΕΝΟΥ

      ΓΡΑΨΕ 'Π. ΠΕΛΑΤΗΣ'
      ΓΡΑΨΕ 'Τ. ΤΑΜΙΑΣ'
      ΓΡΑΨΕ 'Κ. ΚΛΕΙΣΙΜΟ'
      ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ ΕΠΙΛ
      ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 'Π' Η ΕΠΙΛ = 'Τ' Η ΕΠΙΛ = 'Κ'

    ΤΕΛΟΣ_ΑΝ

    ΑΝ ΕΠΙΛ = 'Π' ΤΟΤΕ
      ΔΙΑΒΑΣΕ Χ
      ΑΝ REAR = 10 ΤΟΤΕ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ
        FRONT <- 1
        REAR <- 1
        ΟΝ[REAR] <- Χ
      ΑΛΛΙΩΣ
        REAR <- REAR + 1
        ΟΝ[REAR] <- Χ
      ΤΕΛΟΣ_ΑΝ

    ΑΛΛΙΩΣ_ΑΝ ΕΠΙΛ = 'Τ' Η ΤΕΛΟΣ = ΑΛΗΘΗΣ ΤΟΤΕ! ΟΤΑΝ ΕΧΩ ΠΑΤΗΣΕΙ 'Κ' ΚΑΝΩ ΑΠΩΘΗΣΗ
      ΑΝ FRONT = 0 ΚΑΙ REAR = 0 ΤΟΤΕ
        ΓΡΑΨΕ 'ΑΔΕΙΑ'
      ΑΛΛΙΩΣ_ΑΝ FRONT = REAR ΤΟΤΕ
        ΓΡΑΨΕ ΟΝ[FRONT]
        FRONT <- 0
        REAR <- 0
      ΑΛΛΙΩΣ
        ΓΡΑΨΕ ΟΝ[FRONT]
        FRONT <- FRONT + 1
      ΤΕΛΟΣ_ΑΝ

    ΑΛΛΙΩΣ
      ΤΕΛΟΣ <- ΑΛΗΘΗΣ
    ΤΕΛΟΣ_ΑΝ

  ΜΕΧΡΙΣ_ΟΤΟΥ ΕΠΙΛ = 'Κ' ΚΑΙ FRONT = 0 ΚΑΙ REAR = 0! ΠΡΕΠΕΙ ΝΑ ΕΧΩ ΠΑΤΗΣΕΙ 'Κ' ΚΑΙ ΝΑ ΕΙΝΑΙ
ΑΔΕΙΑ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Λίστες

Μία (απλά) συνδεδεμένη λίστα (linked list) **είναι ένα σύνολο κόμβων διατεταγμένων γραμμικά** (ο ένας μετά τον άλλο). Κάθε κόμβος περιέχει εκτός από τα δεδομένα του και έναν δείκτη που δείχνει προς τον επόμενο κόμβο. Ο δείκτης του τελευταίου κόμβου δε δείχνει σε κάποιον κόμβο (δείκτης στο κενό). Για να το δηλώσουμε αυτό λέμε ότι το πεδίο δείκτη του τελευταίου κόμβου έχει την τιμή NULL.



Εικόνα 1.3.2. α: Αναπαράσταση κόμβου β: Αναπαράσταση απλά συνδεδεμένης λίστας με 3 κόμβους

Για να προσπελάσουμε τους κόμβους της λίστας χρειάζεται να γνωρίζουμε τη διεύθυνση (θέση στη μνήμη) του πρώτου κόμβου της λίστας. Η διεύθυνση αυτή αποθηκεύεται σε μία ειδική μεταβλητή που την ονομάζουμε συνήθως **Κεφαλή** (Head).

Πρόσβαση στους κόμβους μιας συνδεδεμένης λίστας

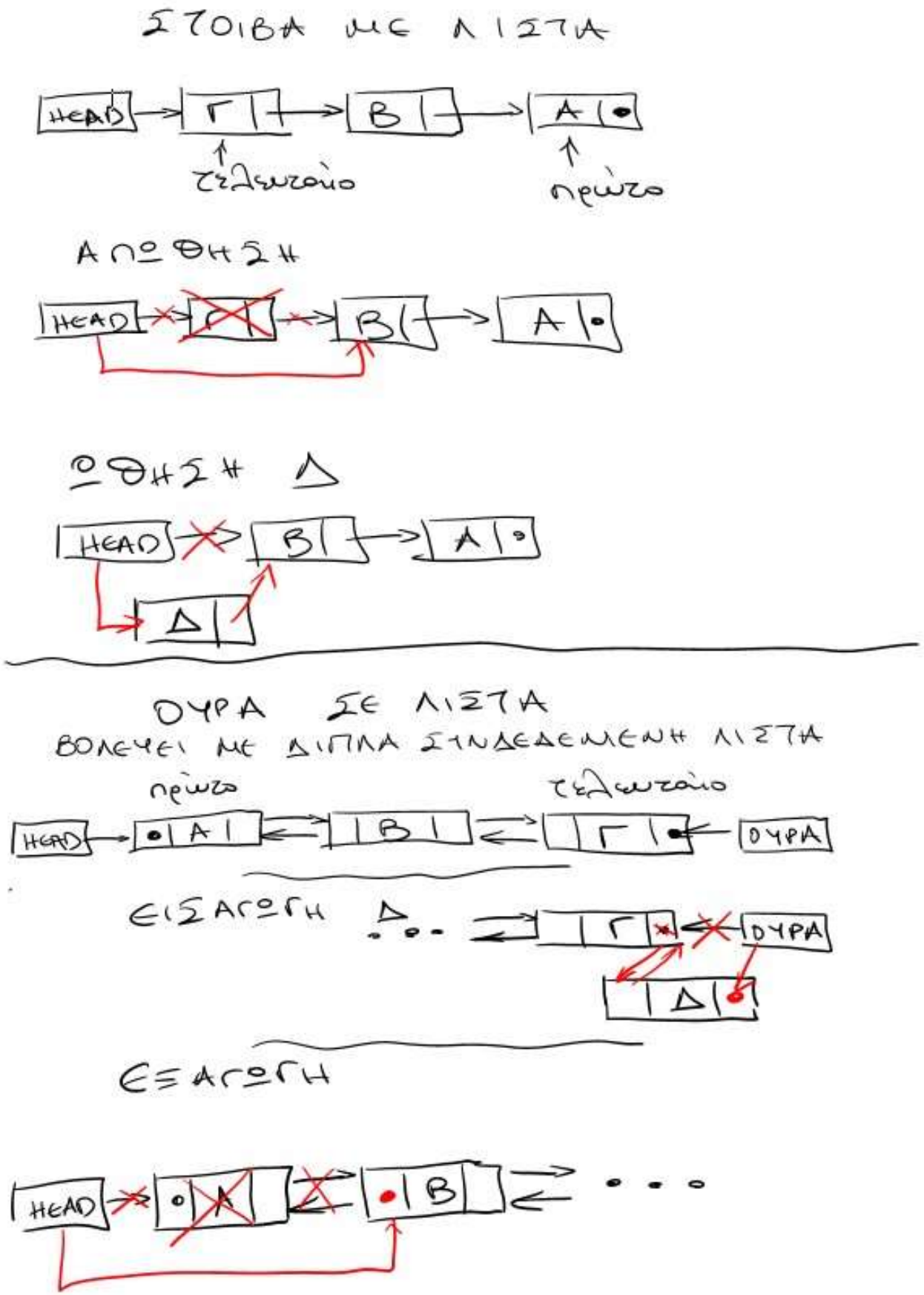
Οι κόμβοι μιας (απλά) συνδεδεμένης λίστας είναι διατεταγμένοι σε μια συγκεκριμένη σειρά, χωρίς αυτό να σημαίνει ότι αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη. Αντίθετα, είναι διασκορπισμένοι σε όλη τη μνήμη και η σύνδεση μεταξύ τους γίνεται μέσω των δεικτών. Έχουμε άμεση πρόσβαση μόνο στον πρώτο κόμβο της λίστας. Επομένως, για να εντοπίσουμε κάποιον από τους ενδιάμεσους κόμβους, πρέπει να ξεκινήσουμε από τον πρώτο κόμβο της λίστας και να ακολουθήσουμε τους δείκτες με τη σειρά, μέχρι να φτάσουμε στον επιθυμητό κόμβο.

Μια λίστα μπορεί να είναι **απλά συνδεδεμένη**, όπως στο παράδειγμα της σκυταλοδρομίας, δηλαδή να μπορούμε να κινηθούμε προς μία μόνο κατεύθυνση, ξεκινώντας από τον πρώτο κόμβο και μετακινούμενοι προς τον τελευταίο, όπως δείχνουν τα βέλη του σχήματος ή να είναι **διπλά συνδεδεμένη** (doubly linked list), δηλαδή, να μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις. Η χρήση του δεύτερου δείκτη προσφέρει τη δυνατότητα ξεκινώντας από οποιοδήποτε κόμβο της λίστας να μπορούμε να διαβάσουμε τη λίστα και προς τις δυο κατευθύνσεις. Ένα τέτοιο παράδειγμα είναι οι σταθμοί του μετρό.

Εικόνα 1.3.8. Οι σταθμοί του μετρό - Μια διπλά συνδεδεμένη λίστα

Οι συνδεδεμένες λίστες αξιοποιούνται για την υλοποίηση της στοίβας και της ουράς, λόγω της δυνατότητάς αυξομείωσης του μεγέθους τους. Μπορείτε να εξηγήσετε γιατί;

Ο βασικός λόγος είναι ότι δεν έχουμε ποτέ υπερχείλιση, ενώ η υποχείλιση συνεχίζει να υφίσταται.



Συνοψίζοντας, θα μπορούσε κάποιος πρόχειρα να συμπεράνει ότι η λίστα δε διαφέρει από τον πίνακα. Παρατηρώντας όμως πιο προσεκτικά, είναι εύκολο να εντοπίσει **σημαντικές διαφορές μεταξύ τους**, οι οποίες παρατίθενται παρακάτω:

- **πίνακας θεωρείται μια δομή τυχαίας προσπέλασης**, σε αντίθεση με μια λίστα που είναι στην ουσία μια δομή ακολουθιακής ή σειριακής προσπέλασης. Για να φθάσουμε, δηλαδή, σ' έναν κόμβο μιας λίστας πρέπει να περάσουμε από όλους τους προηγούμενους ξεκινώντας από τον πρώτο.
- **Ο πίνακας έχει σταθερό μέγεθος**, το οποίο δηλώνεται εξαρχής κατά την υλοποίηση. Αυτό γίνεται, διότι ο πίνακας είναι στατική δομή δεδομένων σε αντίθεση με τη λίστα που είναι δυναμική δομή και το μέγεθός της μπορεί να μεταβάλλεται καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.
- **Οι κόμβοι της λίστας αποθηκεύονται σε μη συνεχόμενες θέσεις** μνήμης σε αντιδιαστολή με τους πίνακες, όπου τα στοιχεία αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

Στα πλεονεκτήματα των λιστών (έναντι των πινάκων) συγκαταλέγονται τα εξής:

- Το **δυναμικό τους μέγεθος**,
- η ευκολία **εισαγωγής και διαγραφής** από οποιοδήποτε μέρος της λίστας, καθώς και
- η **μη αναγκαιότητα δήλωσης του μεγέθους** τους.

Στα μειονεκτήματα των λιστών (έναντι των πινάκων) περιλαμβάνονται τα εξής:

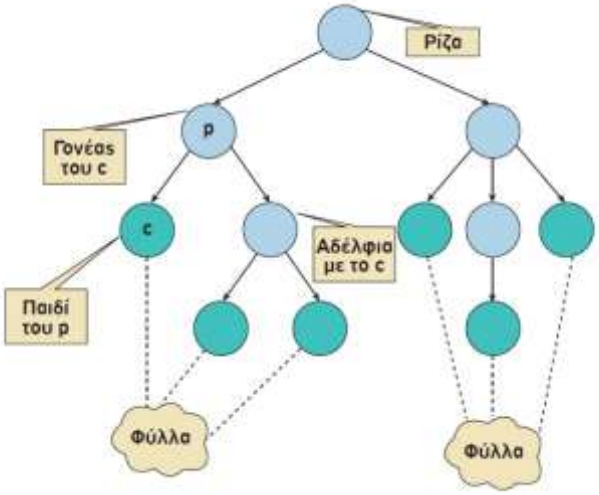
- **Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται.** Είναι αδύνατο να φτάσετε στον n-οστό κόμβο μιας απλά συνδεδεμένης λίστας χωρίς πρώτα να περάσετε από όλους τους κόμβους διαδοχικά μέχρι τον συγκεκριμένο κόμβο ξεκινώντας από τον πρώτο κόμβο. Εναλλακτικά, στην περίπτωση της διπλά συνδεμένης λίστας μπορείτε να ξεκινήσετε και από τον τελευταίο κόμβο. Επομένως, δεν μπορούμε να πραγματοποιήσουμε με αποτελεσματικό τρόπο δυαδική αναζήτηση σε συνδεδεμένες λίστες.
- **Οι συνδεδεμένες λίστες έχουν πολύ μεγαλύτερη επιβάρυνση από τους πίνακες**, αφού οι συνδεδεμένοι κόμβοι της λίστας είναι δυναμικά κατανεμημένοι (οι οποίοι είναι λιγότερο αποτελεσματικοί στη χρήση της μνήμης) και κάθε κόμβος στη λίστα πρέπει, επιπλέον, να αποθηκεύσει έναν πρόσθετο δείκτη που θα δείχνει στον επόμενο κόμβο. Στην περίπτωση των διπλά συνδεδεμένων λιστών χρειαζόμαστε επιπλέον έναν δεύτερο δείκτη που θα δείχνει στον προηγούμενο κόμβο. *(Αυτό μπορεί να γραφτεί και στις διαφορές)*

Δένδρα

Μερικές φορές τα πράγματα **δεν είναι γραμμικά**, όπως έχουμε δει μέχρι τώρα. Σε μία γραμμική δομή, μετά από κάθε στοιχείο ακολουθεί ένα άλλο στοιχείο εκτός και αν είναι το τελευταίο. Τι συμβαίνει όμως στις περιπτώσεις όπου **μετά από ένα στοιχείο ακολουθεί όχι ένα, αλλά δύο, τρία ή και περισσότερα στοιχεία**;

Έτσι λοιπόν, θα λέγαμε ότι ένα δένδρο **αποτελείται** από **κόμβους**, οι οποίοι συνδέονται μεταξύ τους με **ακμές**.

Όταν δύο κόμβοι συνδέονται μεταξύ τους με μία ακμή, τότε ονομάζουμε **«γονέα»** τον κόμβο από τον οποίο ξεκινάει η ακμή και **«παιδί»** τον κόμβο στον οποίο καταλήγει η ακμή. Στην Εικόνα 1.3.11 ο κόμβος p είναι γονέας του κόμβου c και ο κόμβος c είναι παιδί του κόμβου p. Ένας κόμβος μπορεί να έχει κανένα, ένα ή περισσότερα παιδιά. Όλοι οι κόμβοι, εκτός από έναν, έχουν ακριβώς έναν γονέα. Ο κόμβος χωρίς γονέα ονομάζεται **«ρίζα»** (root) και βρίσκεται στην κορυφή του δένδρου. Κόμβοι με τον ίδιο γονέα ονομάζονται **«αδέλφια»**. Οι κόμβοι χωρίς παιδιά ονομάζονται **«φύλλα»**.



Ένα **δένδρο** (tree) είναι μία δομή που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο ακμών μεταξύ των κόμβων με βάση τους **εξής κανόνες**:

- Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται **ρίζα**. Αυτός είναι ένας **κόμβος χωρίς γονέα**.
- **Για κάθε κόμβο c, εκτός από τη ρίζα, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p**. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p.
- **Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή**, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.

Δένδρο θεωρούμε και το κενό δένδρο, δηλαδή το δένδρο που δεν έχει ούτε κόμβους, ούτε ακμές. Το **κενό δένδρο** είναι το μόνο δένδρο χωρίς ρίζα. **Απλό Δένδρο** καλείται το δένδρο με ένα κόμβο μόνο.

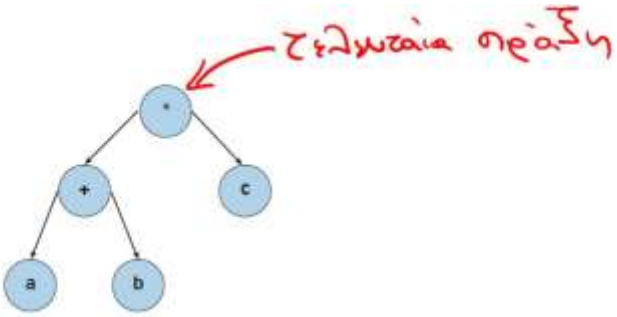
Τα δένδρα είναι μία μη-γραμμική ευέλικτη δομή δεδομένων που χρησιμοποιούνται σε πολλούς το-μείς της επιστήμης των υπολογιστών, συμπεριλαμβανομένων των λειτουργικών συστημάτων, των γραφικών, των συστημάτων βάσεων δεδομένων, των παιχνιδιών, της τεχνητής νοημοσύνης και της δικτύωσης υπολογιστών.

Έχετε σκεφτεί γιατί υπάρχουν όμως τόσα δένδρα; **Υπάρχουν δύο λόγοι για τους οποίους τα δένδρα είναι τόσο ισχυρά.**

- Ο πρώτος λόγος αναφέρεται στη **δυναμικότητα των δένδρων**. Είναι πολύ εύκολο να προσθέσετε, να αφαιρέσετε ή να αναζητήσετε ένα στοιχείο σε ένα δένδρο, όπως θα δούμε στη συνέχεια.
- Ο δεύτερος βασικός λόγος είναι ότι η δομή των δένδρων **μεταφέρει πληροφορίες**. Μοναδική διαδρομή για να φτάσουμε σε έναν κόμβο από τη ρίζα.

Τα **δένδρα απόφασης**, όπως πολύ εύκολα μπορείτε να συμπεράνετε, είναι δένδρα στα οποία κάθε κόμβος (εκτός των φύλλων) αντιπροσωπεύει ένα χαρακτηριστικό (ιδιότητα), κάθε ακμή αντιπροσωπεύει μια απόφαση (κανόνα) και κάθε φύλλο αντιπροσωπεύει ένα αποτέλεσμα.

Διαδεδομένα είναι επίσης τα **δένδρα** για την αναπαράσταση και κατ' επέκταση τον **υπολογισμό αριθμητικών εκφράσεων**.



Εικόνα 1.3.20. (a+b)*c

Ένα **δυαδικό δένδρο** (binary tree) είναι ένα **διατεταγμένο δένδρο**, στο οποίο κάθε κόμβος έχει **το πολύ δύο παιδιά**, το αριστερό και το δεξί παιδί. Μπορούμε, συνεπώς, να μιλάμε για αριστερό και δεξιό υποδένδρο ενός κόμβου.

Ένα **δυαδικό δένδρο αναζήτησης** (binary search tree) είναι ένα δυαδικό δένδρο, όπου για κάθε κόμβο u, όλοι οι κόμβοι του **αριστερού υποδένδρου έχουν τιμές μικρότερες της τιμής του κόμβου u** και όλοι οι κόμβοι του **δεξιού υποδένδρου έχουν τιμές μεγαλύτερες (ή ίσες) της τιμής του κόμβου u**. Για λόγους απλούστευσης θεωρούμε ότι δεν υπάρχουν τιμές ίσες με την τιμή του κόμβου u.

Ποιο νομίζετε ότι είναι το **πλεονέκτημα των δυαδικών δένδρων αναζήτησης**; Το πλεονέκτημα βρίσκεται στο ίδιο το όνομα και συγκεκριμένα στη λέξη «αναζήτηση». **Η αναζήτηση** για μια συγκεκριμένη τιμή **γίνεται ταχύτερα χάρη στον τρόπο αποθήκευσης των τιμών**.

Γράφοι

Αν **αγνοήσουμε αυτούς τους κανόνες των δένδρων** τότε, δεν αναφερόμαστε σε δένδρα αλλά σε μία νέα δυναμική δομή δεδομένων, που ονομάζεται γράφος. Τα δένδρα δεν είναι παρά περιορισμένοι τύποι γράφων. Ένα δένδρο θα είναι πάντα ένα γράφος, αλλά δεν είναι όλοι οι γράφοι δένδρα.

Ένα δένδρο μπορεί μόνο να ρέει προς μία κατεύθυνση, από τον κόμβο ρίζας σε κόμβους φύλλων ή κόμβους παιδιών. Ένα δένδρο μπορεί να έχει μόνο μονόδρομες συνδέσεις - ένας κόμβος παιδιού μπορεί να έχει μόνο έναν γονέα και ένα δένδρο δεν μπορεί να έχει βρόχους ή κυκλικούς δεσμούς.

Με τους γράφους, όλοι αυτοί οι περιορισμοί δεν υπάρχουν. Οι γράφοι δεν έχουν την έννοια ενός κόμβου «ρίζας». **Οι κόμβοι μπορούν να συνδεθούν με οποιονδήποτε τρόπο**.

Ένας **γράφος (graph)** είναι μία δομή που αποτελείται από ένα σύνολο κόμβων (ή σημείων ή κορυφών) και ένα σύνολο γραμμών (ή ακμών ή τόξων) που ενώνουν μερικούς ή όλους τους κόμβους. Ο γράφος αποτελεί την πιο γενική δομή δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν περιπτώσεις γράφων.

Εάν όλες **οι ακμές** σε έναν γράφο **έχουν κατεύθυνση**, ο γράφος ονομάζεται **κατευθυνόμενος γράφος**.

Εάν όλες **οι ακμές** σε έναν γράφο **δεν έχουν κατεύθυνση**, ο γράφος ονομάζεται **μη κατευθυνόμενος γράφος**.

Μέθοδος Διαίρει και Βασίλευε (παράδειγμα η δυαδική αναζήτηση)

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

Η μέθοδος σχεδίασης αλγορίθμων «Διαίρει και Βασίλευε» **μπορεί να αποδοθεί με τα επόμενα βήματα**:

1. Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.
2. Το στιγμιότυπο του προβλήματος υποδιαιρείται σε υπο-στιγμιότυπα του ίδιου προβλήματος.
3. Δίνεται ανεξάρτητη λύση σε κάθε ένα υπο-στιγμιότυπο.
4. Συνδυάζονται όλες οι μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.

```
ΠΡΟΓΡΑΜΜΑ ΤαξινόμησηΜεΕπιλογή_ΔυαδικήΑναζήτηση

ΜΕΤΑΒΛΗΤΕΣ
  ΧΑΡΑΚΤΗΡΕΣ: ΟΝ[10], MAX, ΒΟΗΘ, ΟΝΟΜΑ
  ΑΚΕΡΑΙΕΣ: Ι, Θ, Κ, Α, Τ, Μ
  ΛΟΓΙΚΕΣ: ΒΡΕΘ
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
    ΔΙΑΒΑΣΕ ΟΝ[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 9                    ! ταξινόμηση με επιλογή
    MAX <- ΟΝ[Ι]
    Θ <- Ι
    ΓΙΑ Κ ΑΠΟ Ι + 1 ΜΕΧΡΙ 10
      ΑΝ ΟΝ[Κ] >= MAX ΤΟΤΕ               ! κατά φθίνουσα σειρά
        MAX <- ΟΝ[Κ]
        Θ <- Κ
      ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΒΟΗΘ <- ΟΝ[Ι]
    ΟΝ[Ι] <- ΟΝ[Θ]
    ΟΝ[Θ] <- ΒΟΗΘ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

  ΔΙΑΒΑΣΕ ΟΝΟΜΑ
  Α <- 1                                  !δυαδική αναζήτηση
  Τ <- 10
  ΒΡΕΘ <- ΨΕΥΔΗΣ
  ΟΣΟ Α <= Τ ΚΑΙ ΒΡΕΘ = ΨΕΥΔΗΣ ΕΠΑΝΑΛΑΒΕ
    Μ <- (Α + Τ) DIV 2
    ΑΝ ΟΝΟΜΑ = ΟΝ[Μ] ΤΟΤΕ
      ΒΡΕΘ <- ΑΛΗΘΗΣ
      Θ <- Μ
    ΑΛΛΙΩΣ_ΑΝ ΟΝΟΜΑ > ΟΝ[Μ] ΤΟΤΕ
      Τ <- Μ - 1
    ΑΛΛΙΩΣ
      Α <- Μ + 1
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΑΝ ΒΡΕΘ = ΑΛΗΘΗΣ ΤΟΤΕ
    ΓΡΑΨΕ 'ΒΡΕΘΗΚΕ ', Θ
  ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'ΔΕΝ ΒΡΕΘΗΚΕ'
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Σύγχρονα Προγραμματιστικά Περιβάλλοντα

Αντικειμενοστραφής προγραμματισμός (object-oriented programming) ή αντικειμενοστραφής σχεδίαση είναι μια μεθοδολογία ανάπτυξης εφαρμογών η οποία στηρίζεται σε αυτόνομες προγραμματιστικές οντότητες με δική τους ταυτότητα και συμπεριφορά. Οι οντότητες αυτές καλούνται **αντικείμενα** (objects), αντιστοιχούν σε φυσικές οντότητες ή έννοιες του φυσικού μας κόσμου, και δομούνται με βάση δεδομένα (ιδιότητες) που προσδιορίζουν την υπόστασή τους και ενέργειες (κανόνες συμπεριφοράς) που εφαρμόζονται πάνω στα δεδομένα. Σε μια εφαρμογή, ένα αντικείμενο είναι ο ομαδοποιημένος συνδυασμός δεδομένων και κώδικα, τα οποία έχουμε τη δυνατότητα να χειριστούμε ενιαία. Τα δεδομένα αποτελούν τα χαρακτηριστικά ενός αντικειμένου και αναφέρονται ως **ιδιότητες** (properties) ενώ οι ενέργειες καθορίζουν τη συμπεριφορά του. Οι ενέργειες στον αντικειμενοστραφή προγραμματισμό αναφέρονται και ως **μέθοδοι** (methods).

Μεθοδολογία

Το μόνο που έχουμε να κάνουμε είναι να αναλύσουμε το πρόβλημα το οποίο θέλουμε να επιλύσουμε, δηλαδή να αναγνωρίσουμε και να καταγράψουμε τα βασικά συστατικά στοιχεία της διαδικασίας επίλυσής του που είναι:

- **τα αντικείμενα** που συμμετέχουν με βάση τον ρόλο τους στο συγκεκριμένο σενάριο,
- **οι ιδιότητες κάθε αντικειμένου**, δηλ. τα σχετικά με το συγκεκριμένο πρόβλημα χαρακτηριστικά του, και
- οι υπηρεσίες που προσφέρει ή **οι ενέργειες που υλοποιεί κάθε αντικείμενο (μέθοδοι)** προς αξιοποίηση από άλλες, ώστε να αναπτυχθούν οι **απαραίτητες συνεργασίες μεταξύ των αντικειμένων** για την επίλυση του προβλήματος.

Ένα **αντικειμενοστραφές πρόγραμμα** δομείται ως ένα **δίκτυο συνεργαζόμενων οντοτήτων που είναι τα αντικείμενα**. Κάθε αντικείμενο έχει ένα συγκεκριμένο ρόλο στην εφαρμογή και παρέχει μια υπηρεσία ή εκτελεί μια ενέργεια (μέθοδο) που χρησιμοποιείται από άλλα μέλη του δικτύου, δηλαδή από άλλα αντικείμενα, για την υλοποίηση της συνεργασίας που θα επιλύσει το πρόβλημα.

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

Ο γενικός τύπος ενός αντικειμένου καλείται **κλάση** (class) και καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικειμένου που προέρχεται από αυτή. Μια κλάση αποτελεί ένα **αφαιρετικό** (abstract) στοιχείο (τύπο) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων αντικειμένων.

Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων καλείται **κληρονομικότητα** (inheritance). Με βάση την κληρονομικότητα, μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης να οριστούν υποκλάσεις αντικειμένων. Η κλάση απόγονος (υποκλάση) κληρονομεί και μπορεί να χρησιμοποιήσει όλα τα δεδομένα (ιδιότητες) και τις μεθόδους που περιέχει η κλάση πρόγονος (υπερκλάση).

Στο πεντάλ αυτοκινήτου (κλάση) παρατηρούμε ότι κάθε ένα από τα 3 πεντάλ (υποκλάση) του αυτοκινήτου συμπεριφέρεται διαφορετικά όταν το πατάμε, άρα πρέπει για κάθε πεντάλ (υποκλάση) να γράψουμε διαφορετικές εντολές.

Πολυμορφισμός (polymorphism) είναι μια ιδιότητα του αντικειμενοστραφούς προγραμματισμού με την οποία μια λειτουργία μπορεί να υλοποιείται με πολλούς διαφορετικούς τρόπους. Πχ στον υπολογισμό εμβαδού σχημάτων, πρέπει να γράψουμε τη μέθοδο αυτή για κάθε υποκλάση σχήματος.

Σημειώσεις από τον συνάδελφο Συρρή Ιωάννη

Αντικειμενοστραφής προγραμματισμός

Αντικειμενοστραφής προγραμματισμός: μεθοδολογία ανάπτυξης εφαρμογών που στηρίζεται σε αυτόνομες προγραμματιστικές οντότητες, τα αντικείμενα.

Κάθε **αντικείμενο** χαρακτηρίζεται από τα δεδομένα του (**ιδιότητες**) και τις ενέργειες (**μέθοδοι**) που γίνονται πάνω σε αυτά.

Όνομα Αντικειμένου	
Ιδιότητες (μεταβλητες, σταθερές)	
Μέθοδοι (διαδικασίες)	

Παραδείγματα:
1. Το ποδήλατο του Γιώργου
2. Ο λογαριασμός τραπέζης της Μαρίας

Ποδήλατο_Γιώργου
Ταχύτητα_κίνησης: 32 Γρανάζι_ταχύτητας: 6
Αλλάζει_ταχύτητα () Φρενάρει () Επιταχύνει ()

Λογαριασμός_Μαρίας
Όνομα_δικαιούχου: Μαρία Υπόλοιπο: 1000 €
Κάνω_ερώτηση () Καταθέτω (ποσό) Κάνω_ανάληψη (ποσό) Μεταφέρω (ον_λογ,ποσό)

Αντικειμενοστραφές πρόγραμμα

Δόμηση αντικειμενοστραφούς προγράμματος: **ένα δίκτυο συνεργαζόμενων οντοτήτων (**αντικείμενα). Η συνεργασία επιτυγχάνεται μέσω της κλήσης των μεθόδων τους από τα άλλα αντικείμενα.

Μεθοδολογία κατασκευής αντικειμενοστραφών προγραμμάτων: Αναγνωρίζουμε και καταγράφουμε:

1. τα **αντικείμενα** που συμμετέχουν με βάση τον ρόλο τους στο συγκεκριμένο σενάριο,

2. τις **ιδιότητες** κάθε αντικειμένου

3. τις **υπηρεσίες** που προσφέρει ή τις **ενέργειες** που υλοποιεί κάθε αντικείμενο (μέθοδοι) ώστε να γίνουν οι **συνεργασίες** μεταξύ των αντικειμένων για την επίλυση του προβλήματος.

Όνομα Αντικειμένου 1		Όνομα Αντικειμένου 2
Ιδιότητες (μεταβλητές, σταθερές)	Περιγραφή συνεργασίας	Ιδιότητες (μεταβλητές, σταθερές)
Μέθοδοι (διαδικασίες)		Μέθοδοι (διαδικασίες)

Παραγγελία πίτσας

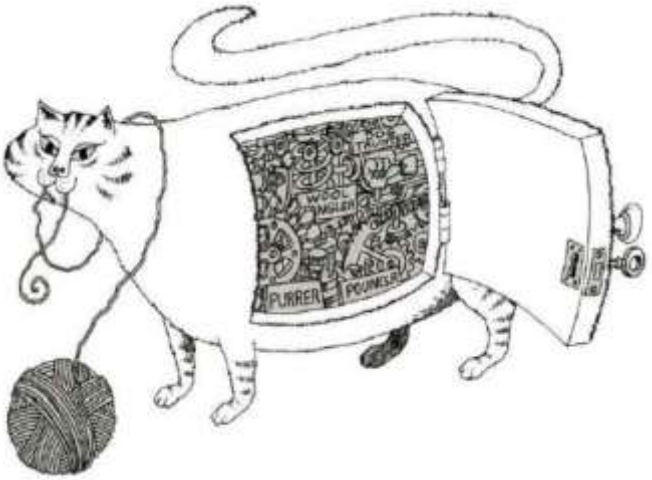
θέλετε να παραγγείλετε μία πίτσα. Τηλεφωνείτε στην τοπική πιτσαρία του κ. Αλέξανδρου, δίνετε την παραγγελία και τη διεύθυνση του σπιτιού σας. Ο κ. Αλέξανδρος ετοιμάζει την πίτσα και ζητάει από τον κ. Πέτρο, να σας παραδώσει την παραγγελία σας στη διεύθυνση που δώσατε.

1η ιδιότητα: Ενθυλάκωση

Η δυνατότητα του αντικειμένου να συνδυάζει εσωτερικά τις ιδιότητες και τις μεθόδους του ονομάζεται **ενθυλάκωση**. Οι μέθοδοι αλλάζουν τις τιμές των ιδιοτήτων.

Δεν χρειάζεται να έχουμε παραμέτρους στις μεθόδους τις ιδιότητες του αντικειμένου. Αυτό ισχύει εξ ορισμού.

Δεν μπορεί μία μέθοδος ενός αντικειμένου να μεταβάλει τις ιδιότητες ενός άλλου αντικειμένου.



2η Ιδιότητα: Αφαιρετικότητα

Ο γενικός τύπος ενός αντικειμένου καλείται **κλάση** (class) και καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικειμένου που προέρχεται από αυτή. Μια κλάση αποτελεί ένα **αφαιρετικό** (abstract) στοιχείο (τύπο) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων αντικειμένων.

Διαγραμματική αναπαράσταση κλάσεων σεναρίου e-pizza

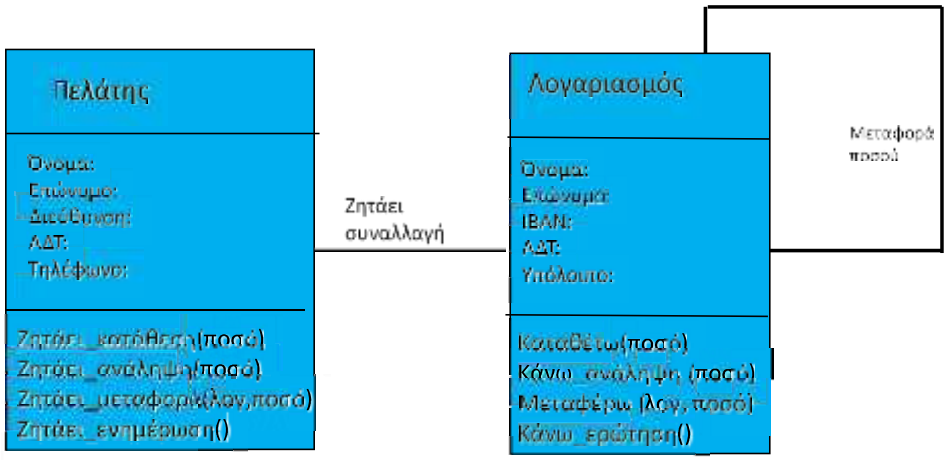
Παράδειγμα τραπεζικών συναλλαγών

Ο Νίκος έχει ένα λογαριασμό στην τράπεζα, θέλει να καταθέσει ένα ποσό στο λογαριασμό του και να μεταφέρει ένα ποσό στο λογαριασμό της Ελένης. Η Ελένη θα κάνει ανάληψη αυτού του ποσού και θα ζητήσει ενημέρωση του λογαριασμού της.

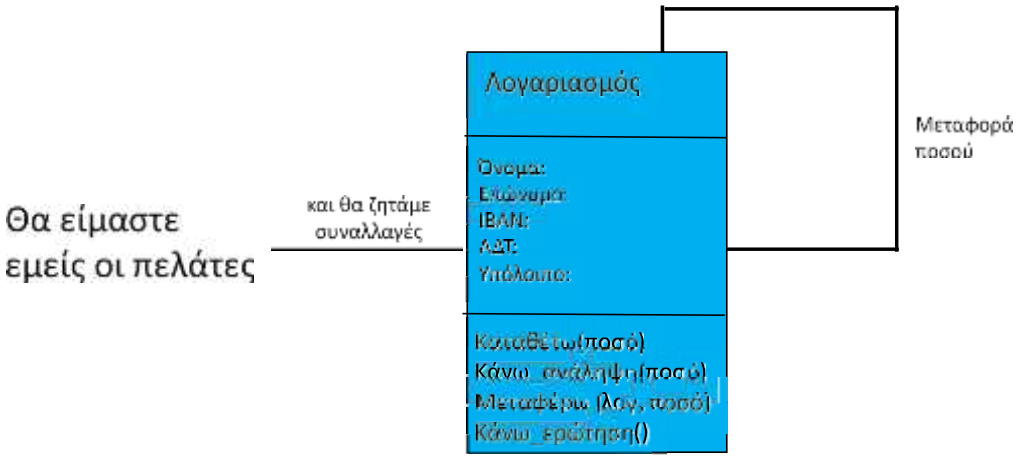
1. Ποια είναι τα αντικείμενα και οι ρόλοι τους;
2. Ποιες οι ιδιότητες του κάθε αντικειμένου;
3. Ποιες οι μέθοδοι του κάθε αντικειμένου;
4. Ποια η συνεργασία μεταξύ των αντικειμένων;

Διαγραμματική αναπαράσταση αντικειμένων

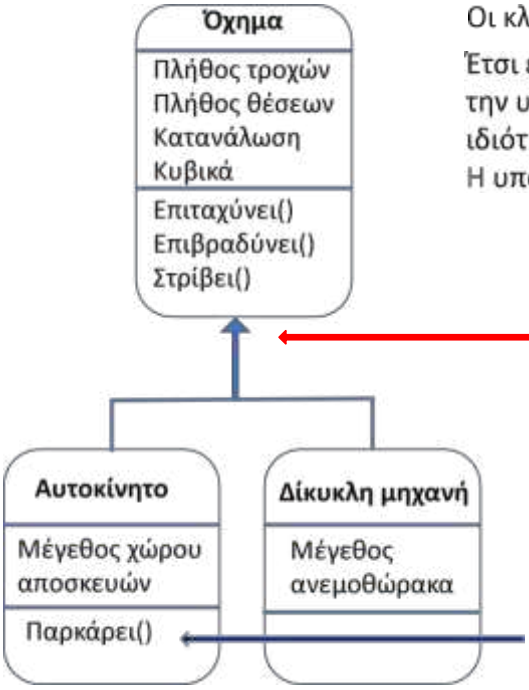
Διαγραμματική αναπαράσταση κλάσεων



Παράδειγμα υλοποίησης σε python



3η Ιδιότητα: κληρονομικότητα



Οι κλάσεις οργανώνονται σε ιεραρχίες, με μορφή πυραμίδας.

Έτσι επιτυγχάνεται η **κληρονομικότητα (inheritance)**. Κληρονομούνται από την υπερκλάση (κλάση πρόγονος) στην υποκλάση (κλάση απόγονος) όλες οι ιδιότητες και οι μέθοδοι της.

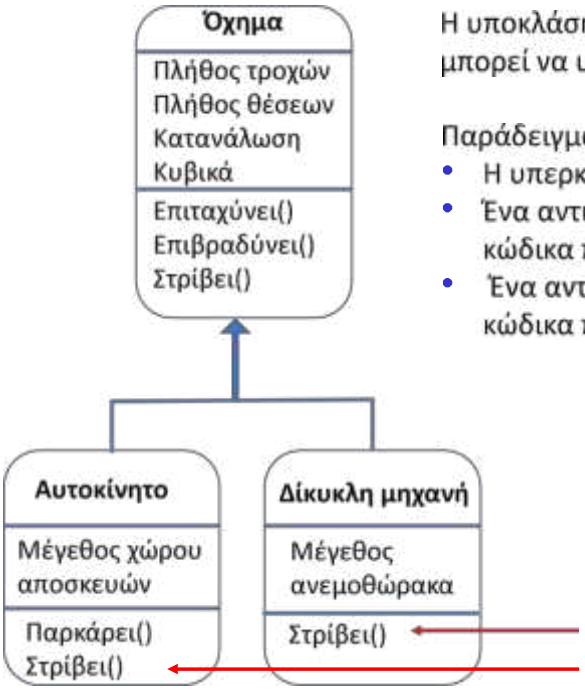
Η υποκλάση μπορεί να έχει επιπλέον ιδιότητες / μεθόδους

Ειδικό σύμβολο γενίκευσης στη διαγραμματική αναπαράσταση της σχέσης κληρονομικότητας

Μία κλάση Α μπορεί να είναι μία υποκλάση της Β αν έχει νόημα να πούμε
«ένα Α είναι ένα (is_a) / (is a kind of) Β»

Αυτόνομο παρκάρισμα !!!

4η Ιδιότητα: πολυμορφισμός



Η υποκλάση μπορεί να αλλάξει τη συμπεριφορά μίας μεθόδου Μια λειτουργία μπορεί να υλοποιείται με πολλούς διαφορετικούς τρόπους (πολυμορφισμός).

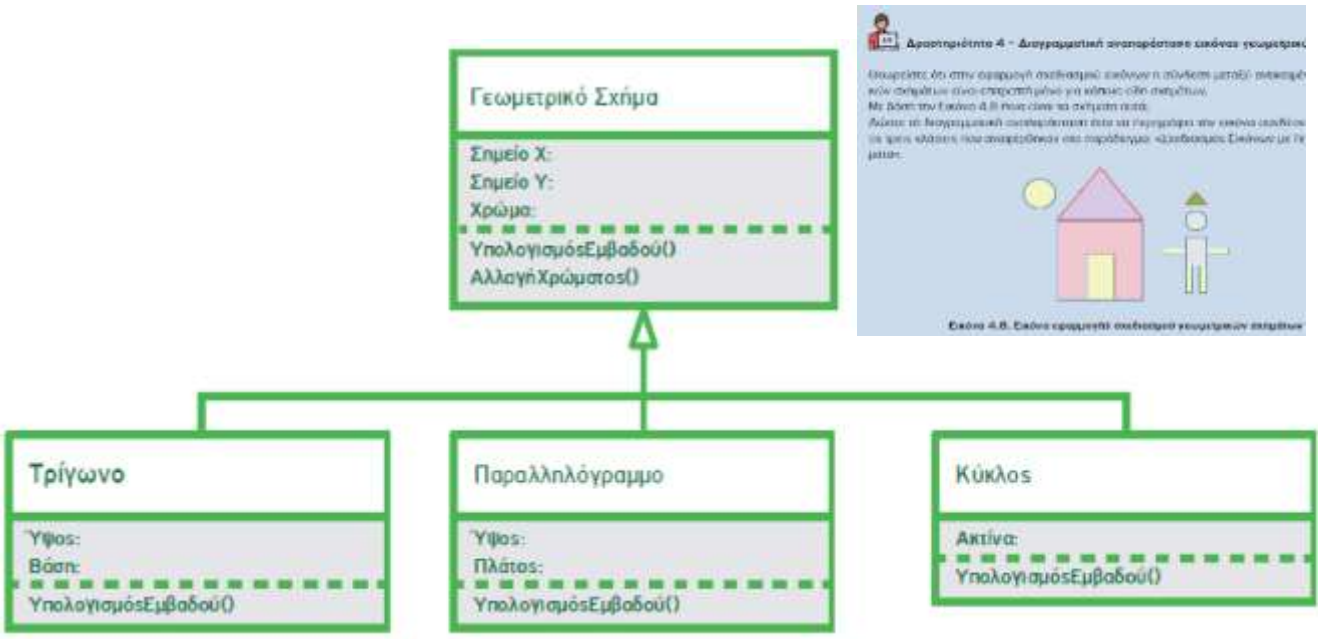
Παράδειγμα:

- Η υπερκλάση **Όχημα** έχει τη μέθοδο **Στρίβει()** χωρίς κώδικα
- Ένα αντικείμενο της κλάσης **Αυτοκίνητο** έχει την μέθοδο **Στρίβει()** με κώδικα που αντιστοιχεί στα αυτοκίνητα
- Ένα αντικείμενο της κλάσης **Δίκυκλη Μηχανή** έχει την μέθοδο **Στρίβει()** με κώδικα που αντιστοιχεί στις Μηχανές

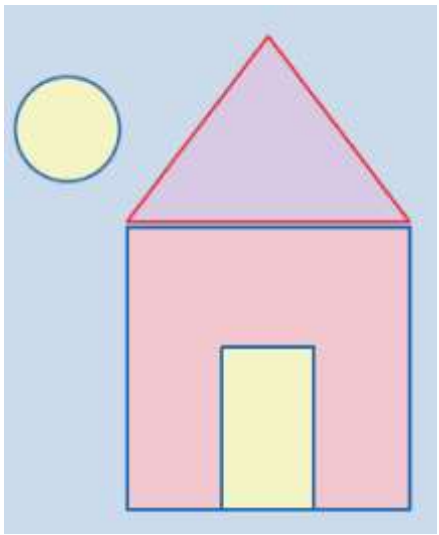
Διαφορετικά στρίβει μία δίκυκλη μηχανή

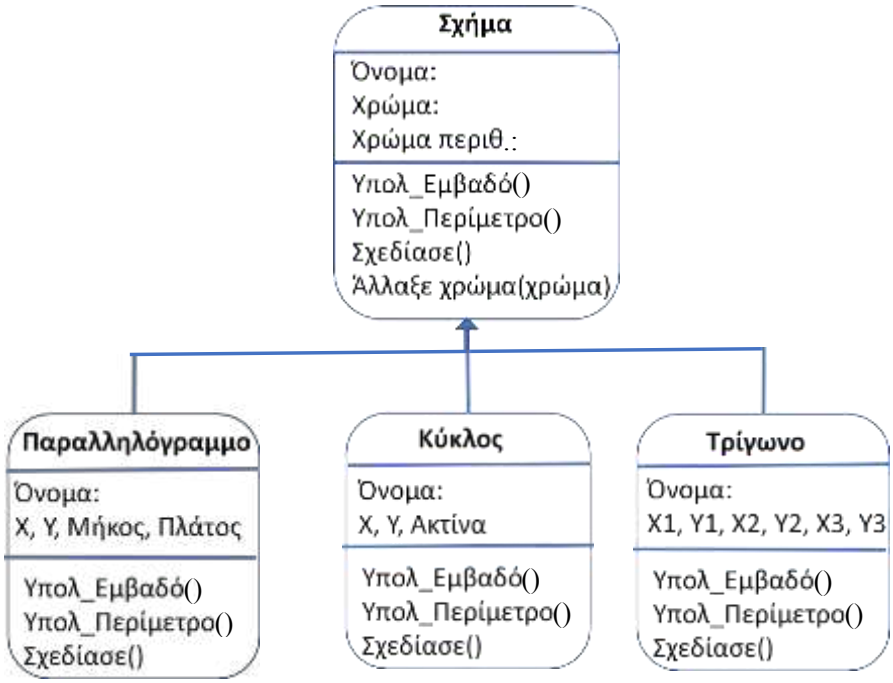
Διαφορετικά στρίβει ένα αυτοκίνητο

Άσκηση κληρονομικότητας και πολυμορφισμού

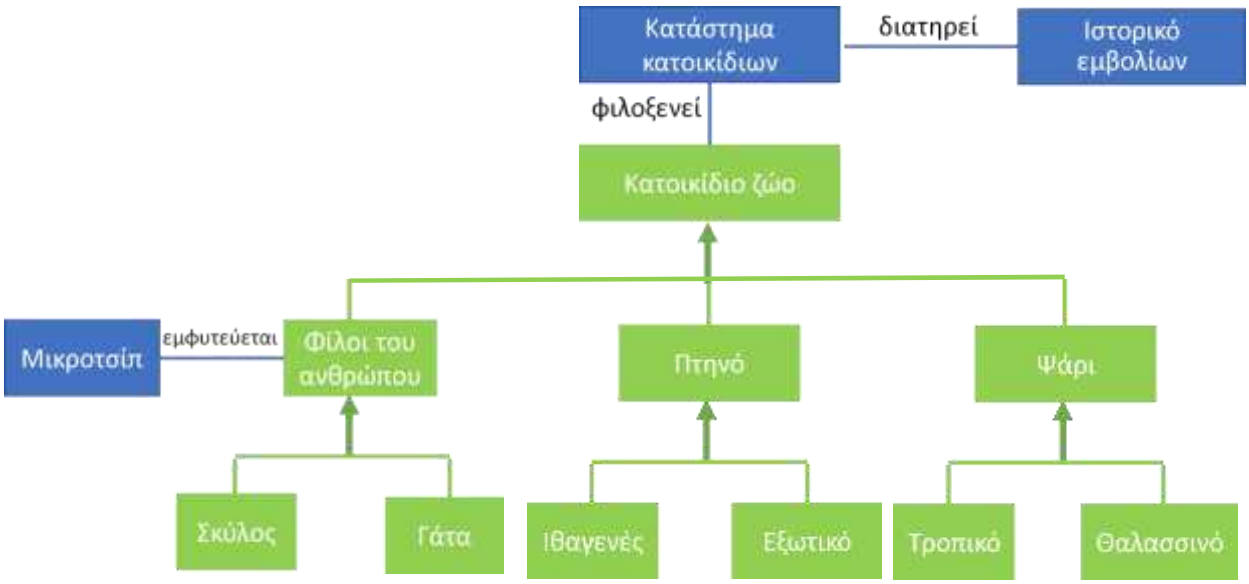


Παράδειγμα Κληρονομικότητας και Πολυμορφισμού σε Python





Διαγραμματική αναπαράσταση κλάσεων με σχέσεις κληρονομικότητας και συνεργασίας



Κλάση - αντικείμενο ή υπερκλάση - υποκλάση

Οικιακή συσκευή	Πλυντήριο, φούρνος, ψυγείο
Υπολογιστής	Laptop της Μαρίας
Ομάδα ποδοσφαίρου	ΠΑΟΚ, ΑΕΚ, ΟΛΥΜΠΙΑΚΟΣ
Γραφική ύλη	Τετράδιο
Αυτοκίνητο	Alfa Romeo Giulietta με πινακίδα ΧΙΕ3434
Laptop	Lenovo T460, i5, 4GB RAM, 500GB HD
Φωτιστικό	Πολυέλαιος, φως νυκτός, φώτα εισόδου
Ηλεκτρονική συσκευή	Κινητό τηλέφωνο, υπολογιστής, router
Βιβλίο	Βιβλίο Πληροφορικής Γ ΓΕΛ 2019

σειρά Τ

Συνδυασμός

ηλεκτρονική συσκευή - υπολογιστής - laptop - Thinkpad - Thinkpad του Βασίλη

Σχέσεις ιεραρχίας ή σχέσεις συνεργασίας;

Άνθρωπος	Κατοικία
Κατοικία	Διαμέρισμα
Πελάτης	Λογαριασμός τραπέζης, χρήματα
Όχημα	Αυτοκίνητο, μηχανή, ποδήλατο
Πελάτης	Φαγητό, κατάστημα
Αυτοκίνητο	Μηχανή αυτοκινήτου, τροχοί, καθίσματα
Ρομποτική πλατφόρμα	Micro:bit, Arduino, Raspberry Pi
Μάθημα Πληροφορικής	Εκσφαλμάτωση, Αντικειμενοστραφής, Δομές Δεδομένων
Καθηγητής	Γυμναστής, Φιλόλογος, Πληροφορικός, Φυσικός

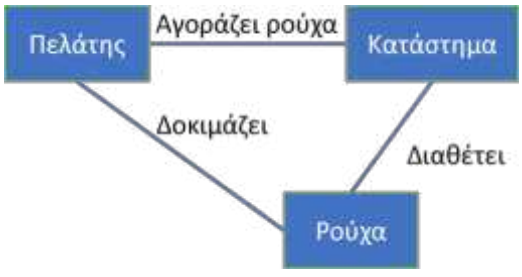
Σύνδεση κλάσεων με σχέσεις ιεραρχίας - συνεργασίας

Έστω οι παρακάτω κλάσεις. Να τις οργανώσετε χρησιμοποιώντας τις κατάλληλες σχέσεις. Προσθέσετε και δικές σας κλάσεις αν θέλετε.

Άνθρωπος, ηλεκτρικό αυτοκίνητο, υβριδικό αυτοκίνητο, ποδήλατο, επιβάτες.

Δημιουργία διαγραμμάτων αντικειμένων/κλάσεων από σενάρια

Ο Μανόλης πήγε στο κατάστημα ρούχων fashion. Δοκίμασε μία άσπρη μπλούζα και ένα μπλε παντελόνι τα οποία και αγόρασε από το μαγαζί.



Περιγραφή και διαγράμματα κλάσεων βάσει σεναρίου

Οι εργαζόμενοι στο στρατό χωρίζονται σε πολιτικό προσωπικό, μόνιμοι στρατιωτικοί και έφεδροι στρατιώτες.

Για καθένα εργαζόμενο γνωρίζουμε το όνομα και το επώνυμό του.

Οι μόνιμοι έχουν επιπλέον βαθμό, θέση και καθήκοντα.

Οι έφεδροι έχουν επιπλέον βαθμό, θέση και ημερομηνία απόλυσης.

Το πολιτικό προσωπικό έχει επιπλέον γραφείο απασχόλησης και ημερομηνία πρόσληψης.

Όλο οι εργαζόμενοι μπορούν να ζητήσουν άδεια.

Οι μόνιμοι και οι έφεδροι μπορούν να ζητήσουν μετάθεση αλλά με διαφορετική διαδικασία.

Το πολιτικό προσωπικό μπορεί να κάνει αίτηση συνταξιοδότησης.

Εκσφαλμάτωση Προγράμματος

Κατηγορίες Λαθών

Μπορούμε να διακρίνουμε τις εξής κατηγορίες λαθών:

1. **Συντακτικά λάθη (ή λάθη κατά την υλοποίηση)**
2. **Λάθη που οδηγούν σε αντικανονικό τερματισμό του προγράμματος (ή λάθη κατά την εκτέλεση)**
3. **Λογικά λάθη που παράγουν λανθασμένα αποτελέσματα**

Συντακτικά λάθη

Κάποιες φορές ένα πρόγραμμα **δεν μπορεί να εκτελεστεί**, επειδή κατά τη μετάφραση εντοπίζονται συντακτικά λάθη. Π.χ. δε γράψαμε σωστά μια δεσμευμένη λέξη, παραλείψαμε μια δεσμευμένη λέξη ή παραλείψαμε να δηλώσουμε μια μεταβλητή.

Λάθη που οδηγούν σε αντικανονικό τερματισμό του προγράμματος

Ένα πρόγραμμα μπορεί να **τερματίσει αντικανονικά** λόγω διαφόρων λαθών. Για παράδειγμα, αν επιχειρήσουμε να διαιρέσουμε με το μηδέν ή αν κατά την ανάγνωση ενός ακεραίου αριθμού εισαχθεί ένα γράμμα.

Λογικά λάθη

Ακόμη κι αν το πρόγραμμά μας δεν περιέχει συντακτικά λάθη και μπορεί να εκτελεστεί, πρέπει οπωσδήποτε να ελεγχθεί, ώστε να διαπιστώσουμε αν κατά την εκτέλεσή του εμφανίζονται λογικά λάθη. Τα λογικά λάθη έχουν ως συνέπεια το πρόγραμμα σε κάποιες περιπτώσεις να **εξάγει λανθασμένα αποτελέσματα**. Για να εντοπίσουμε τα λογικά λάθη μπορούμε να κάνουμε δοκιμαστικές εκτελέσεις του προγράμματός μας και να ελέγξουμε αν για συγκεκριμένες τιμές εισόδου, το πρόγραμμά μας εξάγει σωστά αποτελέσματα.

Εκσφαλμάτωση

Στην ενότητα αυτή θα ασχοληθούμε με τον εντοπισμό των λογικών λαθών ενός προγράμματος και την εκσφαλμάτωσή του, δηλ με **τον έλεγχο, τον εντοπισμό και τη διόρθωση των λαθών**.

Σε μια **δομή επιλογής** μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

- τη συνθήκη ή τις **συνθήκες**
- τις **ομάδες εντολών** που εκτελούνται όταν μια συνθήκη είναι αληθής ή ψευδής.

Εκσφαλμάτωση λογικών λαθών στις **δομές επανάληψης**

Στην ενότητα αυτή θα ασχοληθούμε με την εκσφαλμάτωση κάποιων συνηθισμένων λαθών στις δομές επανάληψης. Σε μια δομή επανάληψης μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

- τη **συνθήκη επανάληψης** ή τερματισμού,
- την **αρχικοποίηση της συνθήκης**,
- την **ενημέρωση της συνθήκης** εντός του βρόχου επανάληψης,
- τις **εντολές** που περιλαμβάνονται εντός του **βρόχου**.

Κατά την **εκσφαλμάτωση** προγραμμάτων που χρησιμοποιούν **πίνακες** χρειάζεται να δίνετε ιδιαίτερη προσοχή:

- στο **μέγεθος των πινάκων** κατά τη δήλωσή τους,
- στους **δείκτες των πινάκων** κατά την προσπέλασή τους,
- στη μη **υπέρβαση των ορίων** του πίνακα.

Κατά την **εκσφαλμάτωση** προγραμμάτων που χρησιμοποιούν **υποπρογράμματα** χρειάζεται να δίνεται προσοχή στον εντοπισμό λογικών λαθών που σχετίζονται με:

- την κλήση του υποπρογράμματος και το πέρασμα των παραμέτρων
- τα λοιπά λογικά λάθη που εμφανίζονται και στα προγράμματα.

Ένα **σενάριο ελέγχου** (test case) περιγράφει τα δεδομένα εισόδου ολόκληρου του προγράμματος ή τμήματος του προγράμματος (διαδικασία, συνάρτηση) και τα αναμενόμενα αποτελέσματα. Τα σενάρια ελέγχου εκτελούνται, είτε σε πραγματικό περιβάλλον προγραμματισμού είτε εικονικά με δημιουργία πίνακα τιμών των μεταβλητών. Σε περίπτωση αποκλίσεων μεταξύ των αναμενόμενων και των πραγματικών αποτελεσμάτων, υπάρχει λάθος το οποίο πρέπει να εντοπιστεί και να διορθωθεί.

Μια δημοφιλής τεχνική ελέγχου είναι ο **έλεγχος μαύρου κουτιού** (black-box testing). Ονομάζεται έτσι επειδή τα δεδομένα εισόδου στα σενάρια ελέγχου προκύπτουν από τις προδιαγραφές του προγράμματος, αγνοώντας εντελώς τον κώδικα. Δηλαδή το πρόγραμμα μοιάζει σαν να βρίσκεται μέσα σε ένα μαύρο κουτί που κρύβει το περιεχόμενό του.

Ιδανικά θα θέλαμε να ελέγξουμε όλες τις τιμές εισόδου και όλα τα πιθανά αποτελέσματα. Αυτό όμως είναι αδύνατο. Γι' αυτό προσπαθούμε να βρούμε αντιπροσωπευτικές τιμές για τα δεδομένα εισόδου που θα παράγουν αντιπροσωπευτικά αποτελέσματα. Το πρώτο βήμα είναι η **δημιουργία ισοδύναμων διαστημάτων τιμών** (equivalence partitioning) για τα δεδομένα εισόδου. Τα διαστήματα θεωρούνται ισοδύναμα, καθώς αν δεν υπάρχουν λάθη, τότε όλες οι τιμές ενός διαστήματος εισόδου θα παράγουν τιμές που θα ανήκουν στο ίδιο διάστημα αποτελεσμάτων.

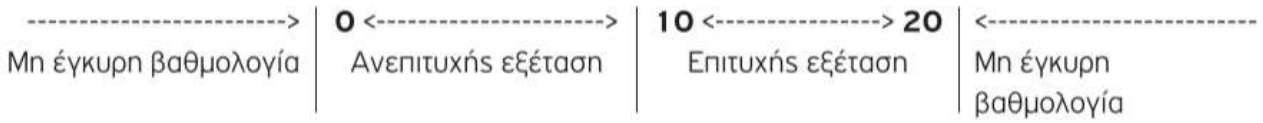
Βήμα 1ο: Δημιουργία ισοδύναμων διαστημάτων

Από την εκφώνηση είναι προφανές ότι υπάρχουν δύο διαστήματα για την είσοδο:

- 0 <= βαθμός < 10
- 10 <= βαθμός <= 20

Επίσης υπάρχουν δύο διαστήματα μη έγκυρων τιμών εισόδου:

- βαθμός < 0
- βαθμός > 20



Διάγραμμα [5.1]. Διαστήματα δεδομένων εισόδου

Βήμα 2ο: Καθορισμός ακραίων τιμών διαστημάτων

Τα διαστήματα των δεδομένων εισόδου απεικονίζονται στο Διάγραμμα 5.1, όπου φαίνεται ότι λείπουν κάποια άκρα. Για να τα υπολογίσουμε αρκεί να προσθέσουμε ή να αφαιρέσουμε 1 από το άκρο του προηγούμενου ή επόμενου διαστήματος αντίστοιχα, αφού σύμφωνα με την εκφώνηση η είσοδος είναι ένας ακέραιος αριθμός. Με αυτό τον τρόπο καταλήγουμε στο Διάγραμμα 5.2.

Διάγραμμα [5.2]. Ακραίες τιμές διαστημάτων εισόδου

Βήμα 3ο: Δημιουργία σεναρίων ελέγχου

Το τελευταία βήμα είναι να δημιουργήσουμε ένα σενάριο ελέγχου για κάθε ακραία τιμή. Κάθε σενάριο πρέπει κατ' ελάχιστο να περιλαμβάνει την τιμή εισόδου, το αναμενόμενο αποτέλεσμα (σύμφωνα με την εκφώνηση του προβλήματος) και περιγραφή της περίπτωσης που ελέγχεται. Έτσι καταλήγουμε στα σενάρια ελέγχου του Πίνακα 5.13

Πίνακας [5.13]. Σενάρια ελέγχου

Α/Α	Είσοδος	Αναμενόμενο αποτέλεσμα	Περίπτωση που ελέγχεται
1	-1	Μη έγκυρη βαθμολογία	Άνω άκρο διαστήματος βαθμός < 0
2	0	Ανεπιτυχής εξέταση	Κάτω άκρο διαστήματος 0 <= βαθμός < 10
3	9	Ανεπιτυχής εξέταση	Άνω άκρο διαστήματος 0 <= βαθμός < 10
4	10	Επιτυχής εξέταση	Κάτω άκρο διαστήματος 10 <= βαθμός <= 20
5	20	Επιτυχής εξέταση	Άνω άκρο διαστήματος 10 <= βαθμός <= 20
6	21	Μη έγκυρη βαθμολογία	Κάτω άκρο διαστήματος βαθμός > 20

Βλέπουμε λοιπόν, ότι ακόμα και ένα πολύ απλό πρόγραμμα με μία μόνο είσοδο, η οποία δεν υφίσταται καμία επεξεργασία, απαιτεί έξι σενάρια ελέγχου. Είναι γεγονός ότι ο έλεγχος ενός προγράμματος είναι διαδικασία που απαιτεί χρόνο, τόσο για τον σχεδιασμό όσο και για την υλοποίηση και εκτέλεση των σεναρίων ελέγχου, ενώ αν προκύψουν λάθη θα απαιτηθεί επιπλέον χρόνος για τον εντοπισμό και τη διόρθωση τους. Γι' αυτό και η φάση του ελέγχου και της εκσφαλμάτωσης αποτελούν σημαντικό ποσοστό του συνολικού χρόνου ανάπτυξης ενός προγράμματος. Σε μεγάλες εταιρείες λογισμικού, μάλιστα, υπάρχουν εξειδικευμένες ομάδες προγραμματιστών που ασχολούνται αποκλειστικά με τον έλεγχο και την εκσφαλμάτωση των προγραμμάτων.

ΘΕΜΑ 3ο ΠΑΝΕΛΛΑΔΙΚΩΝ (ΠΙΘΑΝΕΣ ΠΕΡΙΠΤΩΣΕΙΣ)

1. Διαβάζω τιμές μέχρι να δώσω μία λάθος τιμή.

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΔΙΑΒΑΣΕ Χ** **ΟΣΟ Χ <> 0 ΕΠΑΝΑΛΑΒΕ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΔΙΑΒΑΣΕ Χ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ αθρ*	*!αρχικοποίηση για όλους* **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ Χ <> 0 ΤΟΤΕ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ Χ = 0** *!εμφάνιση για όλες τις τιμές πχ αθρ*

2. Διαβάζω τιμές μέχρι να συμβεί κάτι που δεν έχει σχέση με το Χ, πχ το άθροισμα να γίνει >1000

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΑΘΡ ← 0** **ΟΣΟ ΑΘΡ <= 1000 ΕΠΑΝΑΛΑΒΕ** **ΔΙΑΒΑΣΕ Χ** *! διαβάζω μόνο μέσα* **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ πληθ*	*!αρχικοποίηση για όλους* **ΑΘΡ ← 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΘΡ > 1000** *!εμφάνιση για όλες τις τιμές πχ πληθ*

3. Διαβάζω τιμές μέχρι να εξαντληθεί ένα διαθέσιμο ποσό (πχ 1000€) και να γίνει μηδέν (0).

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΑΘΡ <- 0** **ΟΣΟ ΑΘΡ < 1000 ΕΠΑΝΑΛΑΒΕ** **ΔΙΑΒΑΣΕ Χ** *! διαβάζω μόνο μέσα* **ΑΝ ΑΘΡ + Χ <= 1000 ΤΟΤΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ πληθ*	*!αρχικοποίηση για όλους* **ΑΘΡ <- 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ ΑΘΡ + Χ <= 1000 ΤΟΤΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΘΡ = 1000** *!εμφάνιση για όλες τις τιμές πχ πληθ*

4. Διαβάζω τιμές μέχρι να μην μπορώ να πάρω κάποιο προϊόν (τιμή > διαθ. Ποσό 1000€)

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ (συνηθισμένο **ΛΑΘΟΣ)**
!αρχικοποίηση για όλους **ΑΘΡ <- 0** **ΔΙΑΒΑΣΕ Χ** **ΟΣΟ ΑΘΡ + Χ <= 1000 ΕΠΑΝΑΛΑΒΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΔΙΑΒΑΣΕ Χ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ πληθ*	*!αρχικοποίηση για όλους* **ΑΘΡ <- 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ ΑΘΡ + Χ <= 1000 ΤΟΤΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΘΡ + Χ > 1000** *!εμφάνιση για όλες τις τιμές πχ πληθ* ! **ΛΑΘΟΣ** Αν το άθροισμα είναι 950€ και αγοράσω κάτι με 40€ σταματάει, ενώ το αγοράζει.

Με ΜΕΧΡΙΣ_ΟΤΟΥ (1ος τρόπος)	Με ΜΕΧΡΙΣ_ΟΤΟΥ (2ος τρόπος)
αρχικοποίηση για όλους **ΑΘΡ <- 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΘΡ ← ΑΘΡ + Χ** **ΑΝ ΑΘΡ <= 1000 ΤΟΤΕ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΘΡ > 1000** *!εμφάνιση για όλες τις τιμές πχ πληθ*	*!αρχικοποίηση για όλους* **ΑΘΡ <- 0** **ΤΕΛΟΣ ← ΨΕΥΔΗΣ** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ ΑΘΡ + Χ <= 1000 ΤΟΤΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΑΛΛΙΩΣ** **ΤΕΛΟΣ ← ΑΛΗΘΗΣ** **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΤΕΛΟΣ = ΑΛΗΘΗΣ** *!εμφάνιση για όλες τις τιμές πχ πληθ*

5. Διαβάζω τιμές και στο τέλος υπάρχει μία άλλη ερώτηση για συνέχιση ή τερματισμό (ΝΑΙ/ΟΧΙ)

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ ΑΠ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΠ = 'ΝΑΙ' Ή ΑΠ = 'ΟΧΙ'** **ΟΣΟ ΑΠ = 'ΝΑΙ' ΕΠΑΝΑΛΑΒΕ** **ΔΙΑΒΑΣΕ Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ ΑΠ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΠ = 'ΝΑΙ' Ή ΑΠ = 'ΟΧΙ'** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ αθρ*	*!αρχικοποίηση για όλους* **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ ΑΠ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΠ = 'ΝΑΙ' Ή ΑΠ = 'ΟΧΙ'** **ΜΕΧΡΙΣ_ΟΤΟΥ ΑΠ = 'ΟΧΙ'** *!εμφάνιση για όλες τις τιμές πχ αθρ*

6. Διαβάζω τιμές μέχρι να πάρει μία τιμή τερματισμού το Χ πχ 0 ή να συμβεί κάτι άλλο πχ το άθροισμα>1000

Με ΟΣΟ **(ΛΑΘΟΣ)**	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΔΙΑΒΑΣΕ Χ** **ΑΘΡ ← 0** **ΟΣΟ Χ <> 0 ΚΑΙ ΑΘΡ <=1000 ΕΠΑΝΑΛΑΒΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΔΙΑΒΑΣΕ Χ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *! εμφάνιση για όλες τις τιμές πχ αθρ* *! **ΛΑΘΟΣ** γιατί όταν το άθροισμα γίνει πάνω από 1000 θα ζητήσει ακόμη μία τιμή, αφού το διάβασε είναι τελευταία εντολή μέσα στην επανάληψη*	*!αρχικοποίηση για όλους* **ΑΘΡ ← 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ Χ <> 0 ΤΟΤΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ Χ = 0 Ή ΑΘΡ > 1000** *!εμφάνιση για όλες τις τιμές πχ αθρ*

Με ΟΣΟ (ΣΩΣΤΟ)	
!αρχικοποίηση για όλους **ΔΙΑΒΑΣΕ Χ** **ΑΘΡ ← 0** **ΟΣΟ Χ <> 0 ΚΑΙ ΑΘΡ <=1000 ΕΠΑΝΑΛΑΒΕ** **ΑΘΡ ← ΑΘΡ + Χ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* ***ΑΝ ΑΘΡ <= 1000 ΤΟΤΕ** ! Διορθώνει το λάθος* **ΔΙΑΒΑΣΕ Χ** **ΤΕΛΟΣ_ΑΝ** **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *! εμφάνιση για όλες τις τιμές πχ αθρ*	

7. Διαβάζω τιμές μέχρι να δώσω 3 συνεχόμενες πχ αρνητικές

Με ΟΣΟ	Με ΜΕΧΡΙΣ_ΟΤΟΥ
!αρχικοποίηση για όλους **ΠΛ ← 0** **ΟΣΟ ΠΛ < > 3 ΕΠΑΝΑΛΑΒΕ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ Χ < 0 ΤΟΤΕ** **ΠΛ ← ΠΛ + 1** **ΑΛΛΙΩΣ** **ΠΛ ← 0** **ΤΕΛΟΣ_ΑΝ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** *!εμφάνιση για όλες τις τιμές πχ αθρ*	*!αρχικοποίηση για όλους* **ΠΛ ← 0** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΔΙΑΒΑΣΕ Χ** **ΑΝ Χ < 0 ΤΟΤΕ** **ΠΛ ← ΠΛ + 1** **ΑΛΛΙΩΣ** **ΠΛ ← 0** **ΤΕΛΟΣ_ΑΝ** *!επεξεργασία για όλους* *!επεξεργασία και εμφάνιση για καθένα* **ΜΕΧΡΙΣ_ΟΤΟΥ ΠΛ = 3** *!εμφάνιση για όλες τις τιμές πχ αθρ*

8. Να διαβάζει πολλές τιμές και να εμφανίζει πόσες φορές υπάρχει η μεγαλύτερη

```
ΜΑΧ ← -10^18
ΠΛ ← 0
! ΑΡΧΗ ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ Χ
      ….
      ΑΝ Χ > ΜΑΧ ΤΟΤΕ
          ΜΑΧ ← Χ
          ΠΛ ← 1
      ΑΛΛΙΩΣ_ΑΝ Χ = ΜΑΧ ΤΟΤΕ
          ΠΛ ← ΠΛ + 1
      ΤΕΛΟΣ_ΑΝ
      ……..
!ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ ΜΑΧ <> -10^18 ΤΟΤΕ ! Αν έχει αλλάξει
      ΓΡΑΨΕ ΜΑΧ, ΠΛ
ΑΛΛΙΩΣ
     ΓΡΑΨΕ 'ΚΑΜΙΑ ΤΙΜΗ'
ΤΕΛΟΣ_ΑΝ
```

```
! Β τρόπος με απλές ΑΝ
ΑΝ Χ > ΜΑΧ ΤΟΤΕ
    ΜΑΧ ← Χ
    ΠΛ ← 0      ! Προσοχή
ΤΕΛΟΣ_ΑΝ
ΑΝ Χ = ΜΑΧ ΤΟΤΕ
    ΠΛ ← ΠΛ + 1
ΤΕΛΟΣ_ΑΝ

! Γ τρόπος με απλές ΑΝ
ΑΝ Χ = ΜΑΧ ΤΟΤΕ
    ΠΛ ← ΠΛ + 1
ΤΕΛΟΣ_ΑΝ
ΑΝ Χ > ΜΑΧ ΤΟΤΕ
    ΜΑΧ ← Χ
    ΠΛ ← 1      ! Προσοχή
ΤΕΛΟΣ_ΑΝ
```

9. Να διαβάζει ονόματα και βαθμούς μαθητών μέχρι να δώσουμε για όνομα 'ΤΕΛΟΣ' και να εμφανίζει τα ονόματα των 2 καλύτερων μαθητών

```
ΜΑΧ1 ← -1                  ! ΠΡΩΤΟΣ ΜΕΓΑΛΥΤΕΡΟΣ
ΟΝΜΑΧ1 <- ' '
ΜΑΧ2 ← -1                  ! ΔΕΥΤΕΡΟΣ ΜΕΓΑΛΥΤΕΡΟΣ
ΔΙΑΒΑΣΕ ΟΝ
ΟΣΟ ΟΝ <> 'ΤΕΛΟΣ' ΕΠΑΝΑΛΑΒΕ
   ΔΙΑΒΑΣΕ ΒΑΘ
   ΑΝ ΒΑΘ > ΜΑΧ1 ΤΟΤΕ
      ΜΑΧ2 ← ΜΑΧ1          ! ΠΡΩΤΑ ΒΑΖΩ ΣΤΟ ΜΑΧ2 ΤΟ ΜΑΧ1 ΠΡΙΝ ΤΟ ΧΑΣΩ
      ΟΝΜΑΧ2 ← ΟΝΜΑΧ1
      ΜΑΧ1 ← ΒΑΘ
      ΟΝΜΑΧ1 ← ΟΝ
   ΑΛΛΙΩΣ_ΑΝ ΒΑΘ > ΜΑΧ2 ΤΟΤΕ
      ΜΑΧ2 ← ΒΑΘ
      ΟΝΜΑΧ2 ← ΟΝ
   ΤΕΛΟΣ_ΑΝ
   ΔΙΑΒΑΣΕ ΟΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ ΜΑΧ1 <> -1 ΚΑΙ ΜΑΧ2 <> -1 ΤΟΤΕ
   ΓΡΑΨΕ ΟΝΜΑΧ1, ΟΝΜΑΧ2
ΑΛΛΙΩΣ_ΑΝ ΜΑΧ1 < > -1 ΤΟΤΕ
   ΓΡΑΨΕ ΟΝΜΑΧ1, 'ΚΑΝΕΝΑΣ ΔΕΥΤΕΡΟΣ'
ΑΛΛΙΩΣ
   ΓΡΑΨΕ 'ΚΑΝΕΝΑΣ ΒΑΘΜΟΣ'
ΤΕΛΟΣ_ΑΝ
```

10.

ΛΥΜΕΝΕΣ ΔΥΣΚΟΛΕΣ ΑΣΚΗΣΕΙΣ ΠΙΝΑΚΕΣ

ΔΙΑΒΑΣΜΑ ΜΟΝΟΔΙΑΣΤΑΤΟΥ Α[20] ΜΕΧΡΙ ΝΑ ΓΕΜΙΣΕΙ Ή ΝΑ ΔΩΣΩ ΤΟ 0.

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ1
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[20], Ι, ΠΛ, Χ
ΑΡΧΗ
  ΠΛ <- 0
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΔΙΑΒΑΣΕ Χ
    ΑΝ Χ <> 0 ΤΟΤΕ
      ΠΛ <- ΠΛ + 1
      Α[ΠΛ] <- Χ
    ΤΕΛΟΣ_ΑΝ
  ΜΕΧΡΙΣ_ΟΤΟΥ Χ = 0 Η ΠΛ = 20
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ ΠΛ
    ΓΡΑΨΕ Α[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΔΙΑΒΑΣΜΑ ΚΆΘΕ ΓΡΑΜΜΉΣ ΔΙΔΣΔΙΑΣΤΑΤΟΥ Β[10,20] ΜΕΧΡΙ ΝΑ ΓΕΜΙΣΕΙ Ή ΝΑ ΔΩΣΩ 0.

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ2
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Β[10, 20], Ι, ΠΛ, Χ, ΠΛΘ[10], Κ
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΠΛ <- 0
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ Χ
      ΑΝ Χ <> 0 ΤΟΤΕ
        ΠΛ <- ΠΛ + 1
        Β[Ι, ΠΛ] <- Χ
      ΤΕΛΟΣ_ΑΝ
    ΜΕΧΡΙΣ_ΟΤΟΥ Χ = 0 Η ΠΛ = 20
    ΠΛΘ[Ι] <- ΠΛ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ ΠΛΘ[Ι]
      ΓΡΑΨΕ Β[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΔΙΑΒΑΣΜΑ ΠΙΝΑΚΑ Β[10,20] ΜΕ ΕΛΕΓΧΟ ΟΡΘΗΣ ΕΚΧΩΡΗΣΗΣ ΜΕ ΚΆΠΟΙΟ ΚΡΙΤΗΡΙΟ ΓΙΑ ΚΑΘΕ ΓΡΑΜΜΗ Ή ΣΤΗΛΗ. ΠΧ ΝΑ ΔΙΑΒΑΖΩ ΤΟ ΦΥΛΟ ΚΑΙ ΝΑ ΠΡΕΠΕΙ ΝΑ ΕΊΝΑΙ 10 ΑΝΔΡΕΣ ΚΑΙ 10 ΓΥΝΑΙΚΕΣ ΣΕ ΚΑΘΕ ΓΡΑΜΜΗ

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ3
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Ι, ΠΛΑ, ΠΛΓ, Κ
  ΧΑΡΑΚΤΗΡΕΣ: Φ[10, 20]
```

```
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΠΛΑ <- 0
      ΠΛΓ <- 0
      ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 20
        ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
          ΔΙΑΒΑΣΕ Φ[Ι, Κ]
        ΜΕΧΡΙΣ_ΟΤΟΥ Φ[Ι, Κ] = 'Α' Η Φ[Ι, Κ] = 'Γ'
        ΑΝ Φ[Ι, Κ] = 'Α' ΤΟΤΕ
          ΠΛΑ <- ΠΛΑ + 1
        ΑΛΛΙΩΣ
          ΠΛΓ <- ΠΛΓ + 1
        ΤΕΛΟΣ_ΑΝ
      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΕΧΡΙΣ_ΟΤΟΥ ΠΛΑ = 10 ΚΑΙ ΠΛΓ = 10
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΙΑ Κ ΑΠΟ 1 ΜΕΧΡΙ 20
      ΓΡΑΨΕ Φ[Ι, Κ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΣΥΓΧΩΝΕΥΣΗ ΔΥΟ ΤΑΞΙΝΟΜΗΜΕΜΩΝ ΚΑΤΑ ΑΥΞΟΥΣΑ ΣΕΙΡΑ ΠΙΝΑΚΩΝ Α[5], Β[10] ΣΕ ΕΝΑΝ ΕΠΙΣΗΣ ΤΑΞΙΝΟΜΗΜΕΝΟ Γ[15] ΠΑΛΙ ΚΑΤΑ ΑΥΞΟΥΣΑ ΣΕΙΡΑ

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ4
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[5], Β[10], Γ[15], Ι, ΘΑ, ΘΒ
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
    Α[Ι] <- Ι*5
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    Β[Ι] <- Ι*2
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΘΑ <- 1
  ΘΒ <- 1
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 15
    ΑΝ ΘΑ <= 5 ΚΑΙ ΘΒ <= 10 ΤΟΤΕ
      ΑΝ Α[ΘΑ] < Β[ΘΒ] ΤΟΤΕ
        Γ[Ι] <- Α[ΘΑ]
        ΘΑ <- ΘΑ + 1
      ΑΛΛΙΩΣ
        Γ[Ι] <- Β[ΘΒ]
        ΘΒ <- ΘΒ + 1
      ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ_ΑΝ ΘΑ <= 5 ΤΟΤΕ
      Γ[Ι] <- Α[ΘΑ]
      ΘΑ <- ΘΑ + 1
    ΑΛΛΙΩΣ
      Γ[Ι] <- Β[ΘΒ]
      ΘΒ <- ΘΒ + 1
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 15
    ΓΡΑΨΕ Γ[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
ΚΑΤΑ ΦΘΙΝΟΥΣΑ Ο ΝΕΟΣ ΠΙΝΑΚΑΣ Γ
  ΘΑ <- 5
  ΘΒ <- 10
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 15
    ΑΝ ΘΑ >= 1 ΚΑΙ ΘΒ >= 1 ΤΟΤΕ
      ΑΝ Α[ΘΑ] > Β[ΘΒ] ΤΟΤΕ
        Γ[Ι] <- Α[ΘΑ]
        ΘΑ <- ΘΑ - 1
      ΑΛΛΙΩΣ
        Γ[Ι] <- Β[ΘΒ]
        ΘΒ <- ΘΒ - 1
      ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ_ΑΝ ΘΑ >= 1 ΤΟΤΕ
      Γ[Ι] <- Α[ΘΑ]
      ΘΑ <- ΘΑ - 1
    ΑΛΛΙΩΣ
      Γ[Ι] <- Β[ΘΒ]
      ΘΒ <- ΘΒ - 1
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 15
    ΓΡΑΨΕ Γ[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΕΛΕΓΧΟΣ ΑΝ ΜΙΑ ΤΙΜΉ ΠΡΕΠΕΙ ΝΑ ΜΠΕΙ ΣΕ ΕΝΑ ΤΑΞΙΝΟΜΗΜΕΝΟ ΠΙΝΑΚΑ ΚΑΤΑ ΦΘΊΝΟΥΣΑ ΣΕΙΡΑ ΚΑΙ ΝΑ ΤΟΝ ΤΟΠΟΘΕΤΕΙ ΣΤΗ ΣΩΣΤΉ ΘΕΣΗ

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ5
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Ι, Α[10], Χ, ΒΟΗΘ
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    Α[Ι] <- 110 - Ι*10
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΔΙΑΒΑΣΕ Χ
  ΑΝ Χ > Α[10] ΤΟΤΕ
    Α[10] <- Χ
    ΓΙΑ Ι ΑΠΟ 10 ΜΕΧΡΙ 2 ΜΕ ΒΗΜΑ -1
      ΑΝ Α[Ι] > Α[Ι - 1] ΤΟΤΕ
        ΒΟΗΘ <- Α[Ι]
        Α[Ι] <- Α[Ι - 1]
        Α[Ι - 1] <- ΒΟΗΘ
      ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΤΕΛΟΣ_ΑΝ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΓΡΑΨΕ Α[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΝΑ ΔΙΑΒΑΖΟΥΜΕ ΤΟ ΟΝΟΜΑ ΚΑΙ ΤΟ ΒΑΘΜΟ ΜΑΘΗΤΩΝ ΜΕΧΡΙ ΝΑ ΔΩΣΟΥΜΕ ΤΟ ΟΝΟΜΑ ‘ΤΕΛΟΣ’ ΚΑΙ ΝΑ ΜΑΣ ΕΜΦΑΝΙΖΕΙ ΤΟΥΣ 10 ΚΑΛΥΤΕΡΟΥΣ ΜΑΘΗΤΕΣ

```
ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ6
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Ι, ΠΛ, Θ
  ΧΑΡΑΚΤΗΡΕΣ: ΟΝ, ΜΑΧΟΝ[10]
  ΠΡΑΓΜΑΤΙΚΕΣ: ΒΑΘ, ΜΑΧΒΑΘ[10], ΜΙΝ
ΑΡΧΗ
  ΠΛ <- 0
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΔΙΑΒΑΣΕ ΟΝ
    ΑΝ ΟΝ <> 'ΤΕΛΟΣ' ΤΟΤΕ
      ΔΙΑΒΑΣΕ ΒΑΘ
      ΠΛ <- ΠΛ + 1
      ΑΝ ΠΛ <= 10 ΤΟΤΕ
        ΜΑΧΒΑΘ[ΠΛ] <- ΒΑΘ
        ΜΑΧΟΝ[ΠΛ] <- ΟΝ
      ΑΛΛΙΩΣ
        ΜΙΝ <- ΜΑΧΒΑΘ[1] !ΒΡΙΣΚΩ ΤΗ ΜΙΚΡΟΤΕΡΗ ΤΙΜΗ ΤΟΥ ΠΙΝΑΚΑ
        Θ <- 1
        ΓΙΑ Ι ΑΠΟ 2 ΜΕΧΡΙ 10
          ΑΝ ΜΑΧΒΑΘ[Ι] < ΜΙΝ ΤΟΤΕ
            ΜΙΝ <- ΜΑΧΒΑΘ[Ι]
            Θ <- Ι
          ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ ΒΑΘ > ΜΙΝ ΤΟΤΕ
          ΜΑΧΒΑΘ[Θ] <- ΒΑΘ
          ΜΑΧΟΝ[Θ] <- ΟΝ
        ΤΕΛΟΣ_ΑΝ
      ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
  ΜΕΧΡΙΣ_ΟΤΟΥ ΟΝ = 'ΤΕΛΟΣ'
```

```
  ΑΝ ΠΛ > 10 ΤΟΤΕ
    ΠΛ <- 10
  ΤΕΛΟΣ_ΑΝ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ ΠΛ
    ΓΡΑΨΕ ΜΑΧΒΑΘ[Ι], ΜΑΧΟΝ[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΝΑ ΒΡΟΥΜΕ ΤΗ ΘΕΣΗ ΚΑΙ ΤΗ ΜΕΓΑΛΥΤΕΡΗ ΤΙΜΗ ΕΝΟΣ ΠΙΝΑΚΑ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΜΟΝΟ ΜΙΑ ΕΞΤΡΑ ΜΕΤΑΒΛΗΤΗ

```
ΠΡΟΓΡΑΜΜΑ ΜΑΧ_Θ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[10], Ι, Θ
ΑΡΧΗ
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΔΙΑΒΑΣΕ Α[Ι]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  Θ <- 1
  ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
    ΑΝ Α[Ι] > Α[Θ] ΤΟΤΕ
      Θ <- Ι
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΡΑΨΕ Α[Θ], Θ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```