1. ΕΝΝΟΙΑ ΑΛΓΟΡΙΘΜΟΥ

Ο όρος **αλγόριθμος** περιγράφει τις μεθόδους επίλυσης προβλημάτων.

Συγκεκριμένα:

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

- ΚΡΙΤΗΡΙΑ ΑΛΓΟΡΙΘΜΩΝ

Κάθε αλγόριθμος ικανοποιεί ***απαραίτητα*** τα παρακάτω κριτήρια:

1. **ΕΙΣΟΔΟΣ (INPUT)** το σύνολο των τιμών που δέχεται ο αλγόριθμος ως δεδομένα για την επίλυση του προβλήματος (μπορεί να είναι και το κενό σύνολο)
2. **ΕΞΟΔΟΣ (OUTPUT)** το σύνολο των τιμών που δίνει ο αλγόριθμος ως αποτέλεσμα (***<u>δεν</u>*** μπορεί να είναι το κενό σύνολο)
3. **ΚΑΘΟΡΙΣΤΙΚΟΤΗΤΑ (DEFINITENESS)** οι εντολές πρέπει να είναι αυστηρά καθορισμένες, να μην δημιουργείται καμία αμφιβολία για το πώς θα εκτελεστούν και να μην απαιτούνται πρόσθετες επεξηγήσεις. Π.χ διαίρεση με το 0
4. **ΠΕΡΑΤΟΤΗΤΑ (FINITENESS)** ο αλγόριθμος πρέπει να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών και σε πεπερασμένο χρόνο. Μια διαδικασία που δεν τελειώνει μετά από συγκεκριμένο αριθμό βημάτων λέγεται απλά υπολογιστική διαδικασία.
5. **ΑΠΟΤΕΛΕΣΜΑΤΙΚΟΤΗΤΑ (EFFECTIVENESS)** κάθε εντολή πρέπει να είναι απλή δηλ. όχι μόνο να έχει ορισθεί αλλά και να είναι εκτελέσιμη και σε πεπερασμένο χρόνο.

2. ΣΠΟΥΔΑΙΟΤΗΤΑ ΑΛΓΟΡΙΘΜΟΥ

Η έννοια του αλγόριθμου είναι θεμελιώδης για τη πληροφορική και μας ενδιαφέρει από πολλές σκοπιές:

- *Υλικού* ο ίδιος αλγόριθμος μπορεί να διαφέρει σε ταχύτητα εκτέλεσης από μηχάνημα σε μηχάνημα
- *Γλωσσών Προγραμματισμού* ο αλγόριθμος εξαρτάται από τη γλώσσα προγραμματισμού που χρησιμοποιούμε

- *Θεωρητική* αφορά τη μελέτη του κατά πόσο υπάρχει αποδοτικός αλγόριθμος για την επίλυση κάποιου συγκεκριμένου προβλήματος
- *Αναλυτική* μελετώνται οι πόροι που χρειάζεται ο αλγόριθμος (μνήμη, αποθήκευση, χρόνος CPU)

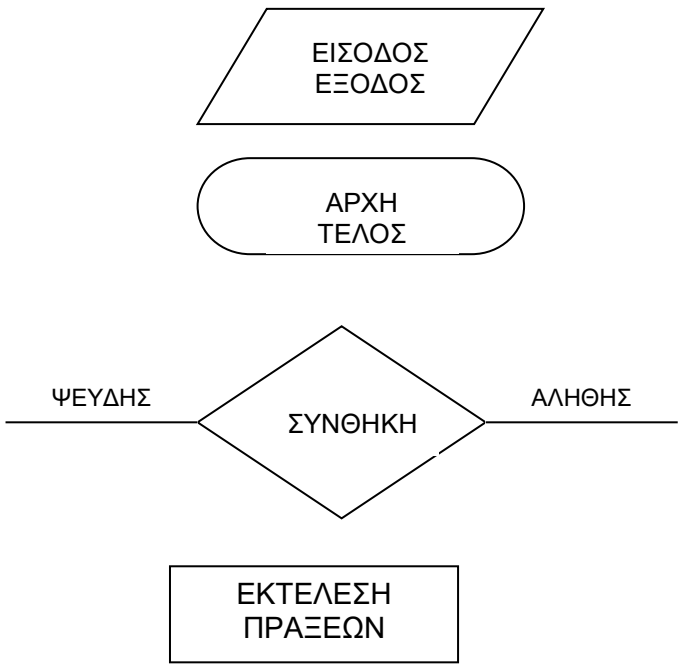
3. ΑΝΑΠΑΡΑΣΤΑΣΗ ΑΛΓΟΡΙΘΜΩΝ

Υπάρχουν διάφοροι τρόποι αναπαράστασης αλγορίθμων:

1. **Ελεύθερο κείμενο**. Έχει το μειονέκτημα ότι είναι αδόμητο και μπορεί να παραβιαστεί η αρχή της αποτελεσματικότητας (αν π.χ. υπάρχουν εντολές που στην πράξη δεν εκτελούνται)
2. **Φυσική Γλώσσα Κατά Βήματα**. Καλύτερη από το ελεύθερο κείμενο γιατί έχει μεγαλύτερη αυστηρότητα αλλά χρειάζεται προσοχή ως προς την αρχή της καθοριστικότητας γιατί μπορεί να υπάρχουν αμφιβολίες στις εντολές.
3. **Διαγραμματικές Τεχνικές** με γνωστότερη τα flow charts αποτελούν καλή λύση για μικρούς αλγορίθμους και όχι πολύ σύνθετους.
4. **Κωδικοποίηση** πρόγραμμα δηλ σε συγκεκριμένη γλώσσα προγραμματισμού που εισάγεται στον Η/Υ και δίνει κατευθείαν τα αποτελέσματα του αλγορίθμου.

Εμείς θα χρησιμοποιήσουμε **ψευδογλώσσα** που είναι κάτι ανάμεσα σε φυσική γλώσσα κατά βήματα και κωδικοποίηση σε γλώσσα προγραμματισμού και κάποια **flow charts.**

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ (FLOW CHART)



ΨΕΥΔΟΓΛΩΣΣΑ

Η γενική μορφή των αλγορίθμων / προγραμμάτων που θα φτιάξουμε είναι η εξής:

Αλγόριθμος (ή πρόγραμμα) όνομα

Σταθερές

Μεταβλητές

Αρχή

Εντολές

Τέλος_όνομα (ή τέλος_προγράμματος)

Δεσμευμένες λέξεις: καθορισμένες λέξεις για τον αλγόριθμο ή τη γλώσσα βάσει των κανόνων της. Συνήθως στη βιβλιογραφία εμφανίζονται με διαφορετική γραφή.

Π.χ. Διάβασε, Εμφάνισε, αν, όσο κλπ

Σταθερές: τιμές που δεν αλλάζουν μέσα στον αλγόριθμο. Δηλώνονται στην αρχή του αλγορίθμου π.χ. N=3 ή π=3,14

Μεταβλητές: τιμές που αλλάζουν μέσα στον αλγόριθμο. Τις δηλώνουμε επίσης στην αρχή του αλγορίθμου δίνοντας ένα όνομα που φροντίζουμε για λόγους κατανοησιμότητας να μας παραπέμπει στα δεδομένα του προβλήματος.

Αποδεκτά ονόματα είναι για παράδειγμα όνομα, μέσος_όρος, α100.

Δεν επιτρέπεται να χρησιμοποιούμε μορφές του τύπου 100α, μέσος όρος, τιμή$, ~τεστ καθώς και δεσμευμένες λέξεις.

Εκτός από το όνομα είναι απαραίτητο να δηλώσουμε και τον τύπο της κάθε μεταβλητής (όπως κάνουμε με το πεδίο ορισμού των συναρτήσεων στα μαθηματικά).

Έτσι δηλώνουμε για παράδειγμα:

Μεταβλητές ακέραιες: χ, ψ, sum

Πραγματικές: ΜΟ, ρ, συνολικό_ποσό

Χαρακτήρες: όνομα, χρώμα_ματιών

Λογικές: βρέθηκε

Ακέραιες σημαίνει ότι οι μεταβλητές παίρνουν ακέραιες τιμές, όπως αντίστοιχα και οι *πραγματικές*.

Χαρακτήρες σημαίνει ότι οι μεταβλητές είναι ένας ή περισσότεροι χαρακτήρες και οι τιμές τους μπαίνουν σε εισαγωγικά.

Λογικές σημαίνει ότι οι μεταβλητές παίρνουν τις τιμές Αληθής ή Ψευδής *μόνο* .

Κάθε φορά που δηλώνουμε μια μεταβλητή ή μια σταθερά δεσμεύεται μια θέση στη μνήμη του υπολογιστή για να εκχωρούνται οι τιμές.

Οι αριθμητικοί τελεστές είναι οι εξής:

+	πρόσθεση
-	αφαίρεση
*****	πολλαπλασιασμός
/	διαίρεση
^	ύψωση σε δύναμη
Div	ακέραιο μέρος διαίρεσης
Mod	υπόλοιπο διαίρεσης

Οι πράξεις διατηρούν την προτεραιότητα όπως στα μαθηματικά.

Οι συγκριτικοί τελεστές είναι οι εξής:

>, >=, <, <=, <>, =

Οι λογικοί τελεστές (βλέπε παρακάτω λογικές πράξεις) είναι:

ΚΑΙ Ή ΟΧΙ

Η προτεραιότητα των τελεστών είναι αριθμητικοί, συγκριτικοί και λογικοί. Για ίδιας προτεραιότητας τελεστών εκτελούμε από αριστερά προς δεξιά ό,τι βρίσκουμε πρώτο.

Επίσης στη ΓΛΩΣΣΑ υπάρχουν και ενσωματωμένες συναρτήσεις για μαθηματικούς υπολογισμούς:

ΗΜ()	Ημίτονο
ΣΥΝ()	Συνημίτονο
ΕΦ()	Εφαπτομένη
Τ_Ρ()	Τετραγωνική ρίζα
Α_Τ()	Απόλυτη τιμή
ΛΟΓ()	Φυσικός λογάριθμος
Ε()	Νεπέριος λογάριθμος
Α_Μ()	Ακέραιο μέρος

Εντολή Εκχώρησης: Αποδίδει τιμή σε μεταβλητή

Χρησιμοποιούμε το σύμβολο **:=** ή ← και όχι το = γιατί η εντολή εκχώρησης λόγω δυναμικότητας δεν είναι απλή ισότητα.

Επιτρέπεται στο δεξί μέλος της εντολής εκχώρησης να εμφανίζεται η μεταβλητή που υπάρχει στο αριστερό.

Π.χ. α←2*α

Δεν επιτρέπεται στο αριστερό μέλος να υπάρχει παράσταση

Π.χ. ~~α*3~~ ← 3*χ+12

Οι κανόνες που πρέπει να ισχύουν είναι:

Α. όλες οι μεταβλητές στο δεξί μέλος πρέπει να έχουν τιμές για παράδειγμα αν έχουμε α← 3*α θα πρέπει οπωσδήποτε κάπου παραπάνω στον αλγόριθμο το α να έχει ξαναπάρει κάποια άλλη τιμή, αλλιώς θα είναι αόριστη.

Β. στο τέλος οι μεταβλητές που αντιπροσωπεύουν τα αποτελέσματα θα έχουν τιμές που πήραν από τις εντολές εκχώρησης.

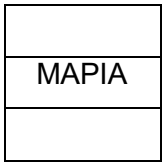
ΠΑΡΑΔΕΙΓΜΑΤΑ:

1. Α← "ΜΑΡΙΑ"

   Σημαίνει ότι έχουμε ορίσει μια μεταβλητή χαρακτήρα με το όνομα Α, δηλ. έχει δεσμευτεί μια θέση στη μνήμη με το όνομα Α και της εκχωρούμε την τιμή ΜΑΡΙΑ

← Α

η μνήμη του Η/Υ
με τη δήλωση της μεταβλητής



← Α

η μνήμη του Η/Υ
μετά την εντολή εκχώρησης

2. Β← "Α"

   Ομοίως με το προηγούμενο (πρόκειται για εκχώρηση του χαρακτήρα Α)

3. Β ← Α

   Εδώ η μεταβλητή Β δεν ξέρουμε τι τύπου είναι ξέρουμε όμως ότι σίγουρα είναι ίδιου τύπου με την Α. Το αποτέλεσμα αυτής της εντολής είναι η μεταβλητή Β να πάρει όποια τιμή έχει η μεταβλητή Α.

4. Α← Α+3

   Εδώ η μεταβλητή Α θα πάρει ότι είχε η ίδια μεταβλητή από πριν αυξημένο κατά 3

Εντολές Εισόδου / Εξόδου

ΔΙΑΒΑΣΕ: με την εντολή ΔΙΑΒΑΣΕ εισάγονται τα δεδομένα στον υπολογιστή από το πληκτρολόγιο.

Η σύνταξη της εντολής είναι:

ΔΙΑΒΑΣΕ μεταβλητή (ή μεταβλητές) (1)

Το πρόγραμμα όταν συναντήσει μια εντολή ΔΙΑΒΑΣΕ σταματάει τη λειτουργία του και περιμένει από το χρήστη (αυτόν που χρησιμοποιεί το πρόγραμμα) να εισάγει την τιμή της μεταβλητής (1) από το πληκτρολόγιο.

ΕΜΦΑΝΙΣΕ ή ΓΡΑΨΕ: με την εντολή αυτή το πρόγραμμα εμφανίζει στην οθόνη κάποιες τιμές ή μηνύματα.
Η σύνταξη της εντολής είναι:
ΕΜΦΑΝΙΣΕ μεταβλητή (ή μεταβλητές) (2) ή
ΕΜΦΑΝΙΣΕ "Σήμερα ο καιρός είναι καλός" (3)

Το πρόγραμμα όταν συναντήσει μια εντολή ΕΜΦΑΝΙΣΕ (ή ΓΡΑΨΕ) θα εμφανίσει στην οθόνη την τιμή της μεταβλητής (2) ή θα εμφανίσει Σήμερα ο καιρός είναι καλός (3)

ΤΥΠΩΣΕ (ή ΕΚΤΥΠΩΣΕ) δουλεύει όπως η ΕΜΦΑΝΙΣΕ μόνο που τα αποτελέσματα εκτυπώνονται στον εκτυπωτή και δεν εμφανίζονται στην οθόνη.

ΠΑΡΑΤΗΡΗΣΕΙΣ**:**

- Με το σύμβολο " ! " μπορούμε να εισάγουμε σχόλια στον αλγόριθμο που μας διευκολύνουν να καταλάβουμε τι κάνει ο αλγόριθμος χωρίς να επηρεάζουν την εκτέλεσή του (ο αλγόριθμος αγνοεί ότι υπάρχει μετά το !).
- Με το σύμβολο " & " μπορούμε να συνεχίσουμε μια εντολή σε δεύτερη γραμμή αν δεν έχει χωρέσει σε μία
- Οι εντολές εισοδου/εξόδου και εκχώρησης είναι **εκτελεστέες** ενώ οι εντολές π.χ. αλγόριθμος τάδε, μεταβλητές ακέραιες χ κλπ είναι **δηλωτικές**

Οι βασικές δομές που χρησιμοποιούμε στους αλγορίθμους είναι:

ΔΟΜΗ ΑΚΟΛΟΥΘΙΑΣ
Πρόκειται για μια δομή που αντιμετωπίζει πολύ απλά προβλήματα και συνίσταται στην μια προς μια εκτέλεση εντολών.

ΛΟΓΙΚΕΣ ΠΡΑΞΕΙΣ
Εκτός από τις αριθμητικές πράξεις στους αλγορίθμους χρησιμοποιούμε και λογικές πράξεις (ΚΑΙ, Ή, ΟΧΙ). Ο παρακάτω πίνακας δείχνει τα αποτελέσματα αυτών των πράξεων για δύο προτάσεις ανάλογα με την τιμή των προτάσεων αυτών.

ΠΡΟΤΑΣΗ Α	ΠΡΟΤΑΣΗ Β	Α Ή Β	Α ΚΑΙ Β	ΟΧΙ Α
ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ
ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ
ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ

ΔΟΜΗ ΕΠΙΛΟΓΗΣ

Με τις δομές επιλογής γίνονται κάποιοι έλεγχοι από το πρόγραμμα και ανάλογα με το αποτέλεσμα των ελέγχων εκτελούνται οι αντίστοιχες εντολές.

ΣΥΝΤΑΞΗ	**ΛΕΙΤΟΥΡΓΙΑ**
1. ΑΠΛΗ ΕΠΙΛΟΓΗ	ελέγχεται η συνθήκη (λογική έκφραση), αν είναι αληθής
ΑΝ ΣΥΝΘΗΚΗ	τότε εκτελούνται οι εντολές μετά το "τότε"
ΤΟΤΕ ΕΝΤΟΛΗ/ΕΣ	αν δεν είναι αληθής τότε εκτελείται η εντολή που υπάρχει μετά
ΤΕΛΟΣ_ΑΝ	το "τέλος_αν"

2. ΣΥΝΘΕΤΗ ΕΠΙΛΟΓΗ	ελέγχεται η συνθήκη, αν είναι αληθής
ΑΝ ΣΥΝΘΗΚΗ	τότε εκτελούνται οι εντολές μετά το "τότε"
ΤΟΤΕ ΕΝΤΟΛΗ/ΕΣ	αν δεν είναι αληθής τότε εκτελείται η εντολή που
ΑΛΛΙΩΣ ΕΝΤΟΛΗ/ΕΣ	υπάρχει μετά το "αλλιώς"
ΤΕΛΟΣ_ΑΝ	

3. ΠΟΛΛΑΠΛΕΣ ΕΠΙΛΟΓΕΣ	τη χρησιμοποιούμε στα
ΑΝ ΣΥΝΘΗΚΗ ***ΤΟΤΕ*** ΕΝΤΟΛΗ/ΕΣ	προβλήματα που μπορεί να
ΑΛΛΙΩΣ_ΑΝ ΣΥΝΘΗΚΗ ***ΤΟΤΕ*** ΕΝΤΟΛΗ/ΕΣ	ληφθούν περισσότερες από 2
ΑΛΛΙΩΣ ΕΝΤΟΛΗ/ΕΣ	αποφάσεις ανάλογα με την τιμή
ΤΕΛΟΣ_ΑΝ	της μεταβλητής ή της έκφρασης

4. ΕΜΦΩΛΕΥΜΕΝΕΣ ΔΙΑΔΙΚΑΣΙΕΣ

ΑΝ ΣΥΝΘΗΚΗ1 **ΤΟΤΕ**

ΑΝ ΣΥΝΘΗΚΗ2 *ΤΟΤΕ* ΕΝΤΟΛΗ /ΕΣ

ΑΛΛΙΩΣ ΕΝΤΟΛΗ/ΕΣ

ΤΕΛΟΣ_ΑΝ

ΑΛΛΙΩΣ

ΑΝ ΣΥΝΘΗΚΗ3 *ΤΟΤΕ* ΕΝΤΟΛΗ/ΕΣ

ΑΛΛΙΩΣ ΕΝΤΟΛΗ/ΕΣ

ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΑΝ

τις εμφωλευμένες τις χρησιμοποιούμε όταν εξετάζονται περισσότερες από μία μεταβλητές ή εκφράσεις

5. ΕΠΙΛΟΓΗ ΟΡΙΩΝ

ΔΙΑΒΑΣΕ ΜΕΤΑΒΛΗΤΗ

ΕΠΙΛΕΞΕ ΜΕΤΑΒΛΗΤΗ

ΠΕΡΙΠΤΩΣΗ ΣΥΝΘΗΚΗ1

ΕΝΤΟΛΗ/ΕΣ

ΠΕΡΙΠΤΩΣΗ ΣΥΝΘΗΚΗ2

ΕΝΤΟΛΗ/ΕΣ

.

.

.

ΠΕΡΙΠΤΩΣΗ ΣΥΝΘΗΚΗ Ν

ΕΝΤΟΛΗ/ΕΣ

ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ

ΕΝΤΟΛΗ/ΕΣ

ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

η επίλεξε χρησιμοποιείται όπως η πολλαπλή επιλογή

ΠΑΡΑΤΗΡΗΣΕΙΣ:

- Μπορούμε να μετατρέψουμε τη μία μορφή στην άλλη
- Στην πολλαπλή τα **αλλιώς *αν*** *δεν κλείνουν και απαιτείται μόνο ένα* ***τέλος*****_αν** για να κλείσει το αρχικό **αν**
- Αντίθετα στην εμφωλευμένη κάθε **αν** κλείνει με ένα αντίστοιχο **τέλος_αν**
- Γενικά μια πολλαπλή επιλογή στην οποία υπάρχουν ν πιθανές περιπτώσεις μπορεί να αντικατασταθεί από ν απλές επιλογές και ν-1 εμφωλευμένες επιλογές
- Όταν χρησιμοποιούμε την εμφωλευμένη καλό θα είναι να στοιχίζουμε τις εντολές προς τα δεξιά κάθε φορά που εμφανίζεται μια νέα εμφωλευμένη επιλογή
- Μπορεί να έχουμε και συνδυασμό των παραπάνω και π.χ. μια πολλαπλή να εμφωλεύεται
- Γενικά η εμφωλευμένη διαδικασία δεν θεωρείται καλή προγραμματιστική τεχνική

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ

Τις δομές επανάληψης τις χρησιμοποιούμε όταν θέλουμε να εκτελεστούν κάποιες εντολές περισσότερες από μία φορά με βάση την τιμή μιας συνθήκης.

ΟΣΟ ΣΥΝΘΗΚΗ **ΕΠΑΝΕΛΑΒΕ**
    ΕΝΤΟΛΗ /ΕΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Όσο η συνθήκη είναι *αληθής* εκτελούνται οι εντολές μόλις η συνθήκη γίνει ψευδής η επανάληψη τελειώνει.
Γι' αυτό πρέπει να φροντίσουμε μέσα στο "σώμα" της επανάληψης η μεταβλητή που υπάρχει στην συνθήκη να αλλάζει τιμή είτε με εντολή διάβασε είτε με εντολή εκχώρησης, αλλιώς αν δεν αλλάζει θα έχουμε άπειρη ανακύκλωση.
Επειδή η συνθήκη ελέγχεται στην αρχή αν είναι εξαρχής ψευδής δεν θα εκτελεστεί ποτέ.

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΕΝΤΟΛΗ/ΕΣ
ΜΕΧΡΙΣ_ΟΤΟΥ ΣΥΝΘΗΚΗ

Εκτελείται όσο η συνθήκη είναι ψευδής, μόλις η συνθήκη γίνει αληθής σταματάει την εκτέλεσή της. Και στην **αρχή_επανάληψης** θα πρέπει να φροντίζουμε να αλλάζει η τιμή της μεταβλητής στη συνθήκη ώστε να σταματάει. Εδώ επειδή η συνθήκη ελέγχεται στο τέλος η επανάληψη θα εκτελεστεί τουλάχιστον μία φορά ακόμα και αν η συνθήκη είναι εξαρχής αληθής.

Οι τιμές που χρησιμοποιούμε για τον τερματισμό των συνθηκών στις παραπάνω δομές λέγονται **"τιμές φρουροί".**

ΓΙΑ ΜΕΤΑΒΛΗΤΗ **ΑΠΟ** ΑΡΧΙΚΗ_ΤΙΜΗ **ΜΕΧΡΙ** ΤΕΛΙΚΗ_ΤΙΜΗ
    ΕΝΤΟΛΗ/ΕΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Αν γνωρίζουμε εκ των προτέρων των αριθμό των επαναλήψεων και δε μας ενδιαφέρει να ελέγχεται κάποια συνθήκη, αλλά απλά να "τρέξει" η μεταβλητή μας τις τιμές ΑΡΧΙΚΗ_ΤΙΜΗ μέχρι ΤΕΛΙΚΗ_ΤΙΜΗ τότε χρησιμοποιούμε τη **ΓΙΑ.**

Επίσης σε περίπτωση που οι τιμές δεν μεταβάλλονται κατά 1 αλλά έχουμε διαφορετικό βήμα (μεταβολή) τότε η δομή γράφεται ως εξής:

ΓΙΑ ΜΕΤΑΒΛΗΤΗ **ΑΠΟ** ΑΡΧΙΚΗ_ΤΙΜΗ **ΜΕΧΡΙ** ΤΕΛΙΚΗ_ΤΙΜΗ **ΜΕ_ΒΗΜΑ** Ω
    ΕΝΤΟΛΗ/ΕΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Υποχρεωτικά στην περίπτωση που ΑΡΧΙΚΗ_ΤΙΜΗ > ΤΕΛΙΚΗ_ΤΙΜΗ το βήμα θα έχει αρνητική τιμή.

Και στις δομές επανάληψης μπορούμε να μετατρέπουμε την μία στην άλλη. Η δομή **για** μετατρέπεται σε οποιαδήποτε άλλη μορφή ενώ η **όσο** και η **αρχή_επανάληψης** δεν μετατρέπονται πάντα σε **για,** καθώς δεν είναι πάντα γνωστός ο αριθμός των επαναλήψεων.
Παραδείγματα:
1. από **για** σε **όσο** και **αρχή_επανάληψης**

```
για ι από 1 μέχρι 100
        εντολή/ες
τέλος_επανάληψης

ι←1                          ! εκχωρώ στο ι την αρχική τιμή
όσο ι <= 100 επανάλαβε       ! η συνθήκη διαμορφώνεται με χρήση του συγκριτικού
                             ! τελεστή <= και την τελική τιμή
        εντολή/ες
        ι←ι+1                ! ενσωματώνουμε το βήμα σε μία εντολή εκχώρησης
                             !για να αλλάζει τιμές το ι και να μην έχουμε άπειρη
                             !ανακύκλωση
τέλος_επανάληψης

ι←1                          ! εδώ βλέπουμε ότι δεν αλλάζει τίποτα εκτός
αρχή_επανάληψης              ! από τη συνθήκη, όπου χρησιμοποιείται ο αντίθετος
        εντολή/ές            ! συγκριτικός τελεστής -γιατί;
        ι←ι+1
μέχρις_ότου ι>100
```

2. από **όσο** ή **αρχή_επανάληψης** σε **για**

ΠΡΟΣΟΧΗ! η μετατροπή αυτή γίνεται μόνο αν είναι γνωστός ο αριθμός των επαναλήψεων και το βήμα σταθερό

```
Ι←2                          Ι←2
Όσο ι<=200 επανάλαβε         αρχή_επανάληψης
        Εντολή/ές                    Εντολή/ές
        ι←ι+4                        ι←ι+4
τέλος_επανάληψης             μέχρις_ότου ι>200
```

```
για ι από 2 μέχρι 200 με_βήμα 4     ! παρατηρούμε ότι το 2 είναι η αρχική τιμή της
        εντολή/ές                    ! μεταβλητής ι και 200 η τελική και η εντολή
τέλος_επανάληψης                    ! ι←ι+4 είναι το βήμα
```

3. από **όσο** σε **αρχή_επανάληψης** και ανάποδα

η μόνο αλλαγή είναι στον τελεστή σύγκρισης της συνθήκης, ο οποίος γίνεται το αντίθετο. Αν δηλ. έχω > γίνεται <=, αν έχω = γίνεται <>, αν έχω >= γίνεται < κοκ

ΣΗΜΕΙΩΣΗ: μπορούμε να έχουμε εμφωλευμένες επαναλήψεις ακολουθώντας τους εξής κανόνες:

- ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό και ο βρόχος που ξεκινάει τελευταίος τελειώνει πρώτος
- η είσοδος σε κάθε βρόχο γίνεται υποχρεωτικά από την αρχή του
- δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής εμφωλευμένων βρόχων

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΑΛΑ ΡΩΣΙΚΑ

Ο υπολογιστής εκτελεί τον πολλαπλασιασμό με διαφορετικό τρόπο από ότι εμείς γνωρίζουμε.

Ο λόγος είναι ότι στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή δηλ. ακολουθίες από 0 και 1. Μετακινώντας μια τέτοια ακολουθία προς τα αριστερά (αριστερή ολίσθηση) ουσιαστικά πολλαπλασιάζουμε με το 2 ενώ μετακινώντας την προς τα δεξιά (δεξιά ολίσθηση) ουσιαστικά διαιρούμε με 2 (ακέραιο μέρος διαίρεσης). Επειδή η ολίσθηση μπορεί να υλοποιηθεί πολύ εύκολα, έπρεπε να βρούμε ένα τρόπο ώστε να εκτελείται ο πολλαπλασιασμός με αυτό τον τρόπο.

Παράδειγμα:

45*19

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

Βάζουμε τους αριθμούς τον ένα δίπλα στον άλλο τον πρώτο τον πολλαπλασιάζουμε διαδοχικά με το 2 και στον δεύτερο κάνουμε div με 2 μέχρις ότου γίνει 1 οπότε σταματάει ο πολλαπλασιασμός. Στη συνέχεια στη τρίτη στήλη μεταφέρουμε το περιεχόμενο της *πρώτης* για τις περιπτώσεις που η δεύτερη στήλη έχει περιττό αριθμό. Το τελικό αποτέλεσμα προκύπτει από το άθροισμα της τρίτης στήλης.

Έτσι ο υπολογιστής στην προκειμένη περίπτωση θα κάνει 4 ολισθήσεις αριστερά, 4 ολισθήσεις δεξιά, έναν έλεγχο για τους περιττούς και μια πρόσθεση.

Ο αλγόριθμος του πολλαπλασιασμού αλά ρωσικά είναι ο εξής:

```
Αλγόριθμος πολλαπλασιασμός_αλά_ρωσικά
Δεδομένα //  M1,M2 //
P←0
Όσο M2 > 0 επανέλαβε ! θα σταματήσει όταν M2=1 αφού είναι ακέραιοι
        Αν M2 mod 2 =1 τότε P←P+M1
        M1←M1*2
        M2←M2 div 2
Τέλος_επανάληψης
Αποτελέσματα // P, το γινόμενο των ακεραίων M1,M2//
Τέλος_πολλαπλασιασμός_αλά_ρωσικά
```