ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΑ

Γ’ ΓΥΜΝΑΣΙΟΥ

Καθηγητής Πέντσας Παναγιώτης

ΠΕΡΙΕΧΟΜΕΝΑ

Τι είναι αλγόριθμος;

Ορισμός: «Αλγόριθμος είναι ένας πεπερασμένος αριθμός συγκεκριμένων βημάτων (εντολών) που απαιτούνται για την επίλυση ενός προβλήματος».

Αλγορίθμους συναντάμε άλλα και δημιουργούμε καθημερινά στη ζωή μας. Από το πιο απλό πράγμα που μπορεί να είναι το βράσιμο ενός αυγού, μέχρι και τα πιο δύσκολα επιστημονικά προβλήματα απαιτούνται αλγόριθμοι.

Οι αλγόριθμοι για να είναι αποτελεσματικοί θα πρέπει να χαρακτηρίζονται από κάποιες ιδιότητες:

1. Το σύνολο των εντολών ενός αλγορίθμου θα πρέπει να είναι πεπερασμένο.(Η εκτέλεση των εντολών του να ολοκληρώνεται σε εύλογο χρονικό διάστημα)
2. Κάθε εντολή θα πρέπει να εκτελείται σε πεπερασμένο χρόνο.
3. Οι εντολές ενός αλγόριθμου θα πρέπει να είναι διατυπωμένες με ακρίβεια και σαφήνεια.
4. Οι εντολές ενός αλγόριθμου θα πρέπει να είναι διατυπωμένες με απλά «λόγια».

Μερικά παραδείγματα αλγορίθμων στη καθημερινή ζωή είναι οι λύσεις στα παρακάτω προβλήματα:

1. Το ψήσιμο ενός ελληνικού καφέ
2. Το πλύσιμο των πιάτων
3. Η μελέτη κάποιου μαθήματος για την επόμενη σχολική μέρα

Τρόποι παρουσίασης αλγορίθμων

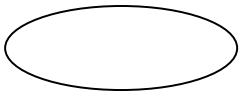
Σκοπός λοιπόν ενός αλγορίθμου είναι να εκφράσει τη διαδικασία επίλυσης ενός προβλήματος με τρόπο κατανοητό. Έτσι λοιπόν υπάρχουν κάποιοι τρόποι έκφρασης(παρουσίασης) αλγορίθμων:

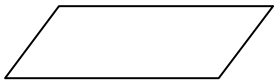
1. Ελεύθερο κείμενο
2. Φυσική γλώσσα με βήματα
3. Ψευδοκώδικας
4. Λογικό διάγραμμα ή διάγραμμα ροής

Εμείς δεν θα ασχοληθούμε με τους τρεις πρώτους αλλά μόνο με το λογικό διάγραμμα παρακάτω.

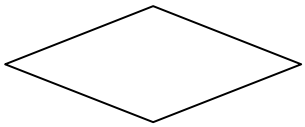
Το Λογικό Διάγραμμα.

Στο Λογικό Διάγραμμα (ΛΔ) η αναπαράσταση του αλγορίθμου γίνεται με την βοήθεια διαφόρων συγκεκριμένων σχημάτων. Τα σχήματα που συνήθως χρησιμοποιούνται φαίνονται παρακάτω μαζί με την λειτουργία που αντιπροσωπεύουν:

Αρχή, τέλος 

Είσοδος δεδομένων προβλήματος 

Επεξεργασία
(π.χ. υπολογισμοί αριθμητικών παραστάσεων) 

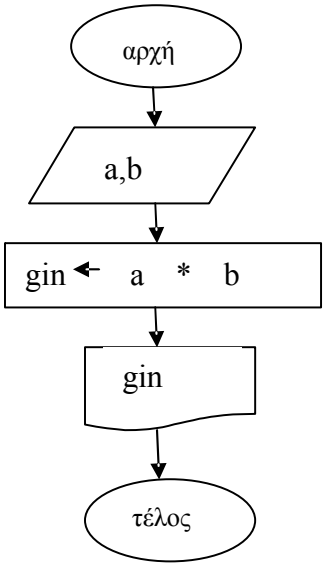
Σύγκριση (π.χ. $\alpha > \beta$) 

Εκτύπωση 

Ροή 

Σύνδεση 

Παράδειγμα χρήσης σχημάτων ΛΔ για τη δημιουργία αλγορίθμου: Ο παρακάτω αλγόριθμος επιλύει το πρόβλημα της εύρεσης του γινομένου δύο αριθμών (a, b).



Σχήμα 1

Στα προβλήματα που θα αντιμετωπισθούν παρακάτω θα δούμε παραδείγματα λογικών διαγραμμάτων για την αναπαράσταση της επίλυσης των προβλημάτων αυτών.

Αλγόριθμοι και υπολογιστές.

Για την επιστήμη της πληροφορικής οι αλγόριθμοι συνδέονται άμεσα με τα συστήματα των ηλεκτρονικών υπολογιστών.

Για να μπορέσουν λοιπόν οι αλγόριθμοι να χρησιμοποιηθούν από τους ηλεκτρονικούς υπολογιστές για την επίλυση διαφόρων προβλημάτων είτε αυτά είναι επιστημονικά είτε είναι εμπορικές εφαρμογές κ.λ.π απαιτείται η μετατροπή του αλγορίθμου –με την βοήθεια μιας γλώσσας προγραμματισμού – σε πρόγραμμα και στη συνέχεια η μετάφραση και εκτέλεση του προγράμματος (εντολή προς εντολή) από κάποιον Η/Υ.

Τι είναι όμως οι γλώσσες προγραμματισμού;

Όπως και η γλώσσα που μιλάμε διαθέτει γραμματικούς και συντακτικούς κανόνες για να μπορούμε να επικοινωνούμε συντάσσοντας προτάσεις, έτσι και οι γλώσσες προγραμματισμού διαθέτουν συντακτικούς και γραμματικούς κανόνες (οι οποίοι είναι πολύ κοντά σε μια γλώσσα που μιλάμε -αγγλικά).

Βέβαια για τη σύνταξη ενός προγράμματος σε κάποια γλώσσα προγραμματισμού, θα πρέπει οι συντακτικοί και γραμματικοί κανόνες της γλώσσας αυτής να τηρούνται με απόλυτη ακρίβεια σε αντίθεση με τις κοινές γλώσσες (αγγλικά,ελληνικά κλπ) όπου μπορούν να γίνουν κατανοητές και προτάσεις που δεν ακολουθούν απόλυτα τους κανόνες του συντακτικού και της γραμματικής της γλώσσας αυτής.

Έτσι μπορούμε να δίνουμε στους υπολογιστές, τους αλγόριθμους(με τη μορφή προγραμμάτων) και να παίρνουμε τις λύσεις για τα προβλήματα μας.

Μια τέτοια γλώσσα προγραμματισμού είναι και η LOGO.

Τι είναι μεταβλητές και τι είναι σταθερές;

Επειδή θα μιλήσουμε παρακάτω για μεταβλητές καλό είναι στο σημείο αυτό να πούμε 2 λόγια. Οι μεταβλητές που χρησιμοποιούμε στους αλγορίθμους αλλά και στα προγράμματα είναι περίπου* το ίδιο με τις μεταβλητές που μάθατε στα μαθηματικά.

*(ουσιαστική διαφορά είναι ότι μια μεταβλητή στα μαθηματικά μπορεί να χωρέσει έναν αριθμό όσο μεγάλος και να είναι – να προσεγγίζει το άπειρο – , ενώ στον προγραμματισμό, επειδή οι μεταβλητές συνδέονται – όπως θα δούμε στην επόμενη σελίδα – με την κύρια μνήμη του υπολογιστή [RAM], υπάρχει ο περιορισμός της χωρητικότητας της μνήμης RAM – η μνήμη RAM δεν είναι άπειρη, έχει συγκεκριμένη χωρητικότητα)

Μια μεταβλητή δεν είναι τίποτε άλλο παρά μια θέση μνήμης στην κύρια μνήμη ενός Η/Υ(RAM). Κάθε αλγόριθμος(πρόγραμμα) για να δώσει τη λύση στο πρόβλημα που αντιμετωπίζει θα πρέπει να επεξεργαστεί δεδομένα και να παράξει αποτελέσματα Τα δεδομένα που δίνονται κατά τη διάρκεια εκτέλεσης του αλγορίθμου(προγράμματος) άλλα και τα αποτελέσματα που παράγονται φυλάσσονται στην κύρια μνήμη του Η/Υ(RAM).

Μια μεταβλητή έχει ένα όνομα και μπορεί να πάρει ένα μόνο περιεχόμενο κάθε φορά το οποίο βεβαίως μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης ενός προγράμματος, γι'αυτό άλλωστε ονομάζεται και μεταβλητή. Το όνομα για μια μεταβλητή θα πρέπει να επιλέγεται με προσοχή και να είναι σχετικό με τη χρήση της μεταβλητής. Π.χ. Εάν χρειαζόμαστε μια μεταβλητή για να της δώσουμε ως περιεχόμενο την μεγαλύτερη τιμή από ένα πλήθος αριθμών ένα κατάλληλο όνομα θα ήταν το: MAX και όχι το: ΠΑΝΟΣ.

Από την άλλη μεριά έχουμε τις σταθερές. Όπως αναφέρει και το όνομα τους δεν μπορούν να αλλάζουν περιεχόμενο. Είναι για παράδειγμα όπως το επίθετο μας το οποίο δεν αλλάζει αλλά παραμένει σταθερό για όλα τα χρόνια της ζωής μας. Π.χ ο τριγωνομετρικός αριθμός π = 3,14.

Με ποιόν τρόπο όμως ορίζονται οι μεταβλητές έτσι ώστε να μπορούν να χρησιμοποιηθούν σε έναν αλγόριθμο(πρόγραμμα); Αυτό γίνεται στη LOGO με την εντολή make όπως θα δείτε παρακάτω. (χρήση 1 της **make** παρακάτω)

Με ποιους τρόπους όμως μπορούμε να δώσουμε περιεχόμενο ή να αλλάξουμε το προηγούμενο περιεχόμενο μιας μεταβλητής; Αυτό γίνεται στους αλγόριθμους(προγράμματα) με τη χρήση της εντολής εκχώρησης **τιμής σε μεταβλητή.** Η εντολή αυτή αποτελείται από 2 μέρη, το αριστερό και το δεξί. Διαχωριστικό μεταξύ των 2 μερών είναι το βελάκι ← . Το αριστερό μέρος είναι πάντοτε μια μεταβλητή. Το δεξί μέρος μπορεί να είναι διάφορα πράγματα π.χ. μια σταθερά(αριθμός ή κείμενο), μια μεταβλητή , μια παράσταση που περιέχει ή όχι μεταβλητές. Σε κάθε περίπτωση υπολογίζεται η τιμή του δεξιού μέρους και δίνεται ως περιεχόμενο στη μεταβλητή που βρίσκεται στο αριστερό μέρος. Π.χ.

1. **α← 0** (ορίζεται η μεταβλητή α και της δίνεται το περιεχόμενο 0)

2. **α ← β** (αν η μεταβλητή β έχει περιεχόμενο τον αριθμό 5, τότε το 5 γίνεται περιεχόμενο της μεταβλητής α)

3. **α ← β + 2** (αν η μεταβλητή β έχει περιεχόμενο τον αριθμό 5, τότε υπολογίζεται η τιμή της παράστασης 5+2 δηλαδή 7 και το 7 γίνεται περιεχόμενο της μεταβλητής α)

(χρήσεις 1,2,3 της εντολής **make** παρακάτω)

Βασικές εντολές (και στοιχεία συντακτικού)της γλώσσας προγραμματισμού LOGO.

Η εντολή make έχει διάφορες χρήσεις στη LOGO:

1. Ορίζει μεταβλητές και τους δίνει αρχική τιμή.
   Π.χ. **make "MAX 0** (ορίζει τη μεταβλητή MAX και της δίνει το περιεχόμενο 0)

2. Υπολογίζει την τιμή μιας αριθμητικής παράστασης και την δίνει σαν περιεχόμενο σε μια μεταβλητή.
   Π.χ. **make "sum 2 + 2** (υπολογίζει την τιμή της αριθμητικής παράστασης 2+2 και το αποτέλεσμα (4) το βάζει μέσα στην μεταβλητή **sum**).

3. Υπολογίζει την τιμή μιας παράστασης που περιέχει και μεταβλητή(ές) και την δίνει περιεχόμενο σε μια μεταβλητή.
   Π.χ. **make "sum :a + 2** (εάν η μεταβλητή **:a** έχει την τιμή 5 τότε υπολογίζει την τιμή της αριθμητικής παράστασης 5+2 και το αποτέλεσμα (7) το βάζει μέσα στην μεταβλητή **sum**).

4. Χρησιμοποιείται επίσης και για την εισαγωγή δεδομένων σε μεταβλητές από το πληκτρολόγιο και κατά τη διάρκεια εκτέλεσης ενός προγράμματος.
   Π.χ. Όταν τα δεδομένα είναι αριθμοί: **make "number readword.** Διαβάζει από το πληκτρολόγιο τον αριθμό που πληκτρολόγησε ο χρήστης (έστω 5) και το δίνει περιεχόμενο στη μεταβλητή **number**.
   Π.χ. Όταν τα δεδομένα είναι κείμενο: **make "word readlist**.
   Διαβάζει από το πληκτρολόγιο τη λέξη που πληκτρολόγησε ο χρήστης (έστω «καλημέρα») και τη δίνει περιεχόμενο στη μεταβλητή **word**.

Η εντολή print(pr) στη LOGO.

Η εντολή **print** στη **LOGO** είναι μια απλή εντολή η οποία μας βοηθάει να εμφανίζουμε τα αποτελέσματα της επίλυσης ενός προβλήματος στην οθόνη του ηλεκτρονικού υπολογιστή.

Παραδείγματα χρήσης της εντολής pr:

1. **pr "goodmorning** (εμφανίζει στην οθόνη: goodmorning).
2. **pr :x** (εάν η μεταβλητή x έχει σαν περιεχόμενο την τιμή 7, τότε θα εμφανιστεί στην οθόνη το περιεχόμενο αυτό, δηλ: 7)
3. **pr [very good day]** (όταν θέλουμε να εμφανίσουμε στην οθόνη μια ολόκληρη πρόταση με περισσότερες από μια λέξεις τότε την κλείνουμε σε: [ ])
4. **(print [Δώσε ποσό Α τριμήνου για τον] :trexon_math "μαθητή)** (όταν θέλουμε να εκτυπώσουμε συνδυασμούς των παραπάνω με μια εντολή, τότε βάζουμε ολόκληρη την εντολή μέσα σε παρενθέσεις)

Η δομή ενός προγράμματος στη LOGO(για τις ανάγκες του μαθήματος).

Κάθε πρόγραμμα στη LOGO αποτελείται από την επικεφαλίδα, τον κορμό και το τέλος.

Επικεφαλίδα
Κορμός
Τέλος

Στην επικεφαλίδα δίνουμε το όνομα του προγράμματος .

Στον κορμό :

1. Γίνεται η εισαγωγή των δεδομένων με τη βοήθεια των μεταβλητών(μεταβλητές εισόδου).
2. Δίνουμε όλες τις εντολές που είναι απαραίτητες για την επίλυση του συγκεκριμένου προβλήματος(την επεξεργασία δηλαδή των δεδομένων που εισήχθηκαν παραπάνω).
3. Κατά την επεξεργασία των δεδομένων παράγονται αποτελέσματα(ενδιάμεσα ή/και τελικά). Τα αποτελέσματα αυτά συνήθως δίνονται σε μεταβλητές(μεταβλητές εξόδου) και στη συνέχεια εκτυπώνονται συνήθως στην οθόνη του Η/Υ.

Στο τέλος δίνουμε τη δεσμευμένη(δεσμευμένες είναι οι λέξεις που ανήκουν στο λεξιλόγιο της γλώσσας προγραμματισμού LOGO και δεν μπορούν να χρησιμοποιηθούν για ονόματα μεταβλητών) λέξη **end**.

Παράδειγμα προγράμματος στη LOGO που αντιστοιχεί στο λογικό διάγραμμα του Σχήματος 1:

to ginomeno (Επικεφαλίδα)

make "a readword	**(Κορμός –** εισαγωγή δεδομένων στις
make "b readword	μεταβλητές εισόδου **a, b.**
make "gin :a * :b	**Κορμός** – επεξεργασία δεδομένων και παραγωγή αποτελεσμάτων**.**
pr :gin	**Κορμός –** εκτύπωση αποτελέσματος που βρίσκεται μέσα στη μεταβλητή εξόδου gin)
end (	**Τέλος)**

Μέθοδος επίλυσης (απλών)προβλημάτων στον ηλεκτρονικό υπολογιστή

Υπάρχουν περισσότερες από μία μέθοδοι για την επίλυση κάποιου προβλήματος με την βοήθεια του ηλεκτρονικού υπολογιστή. Παρακάτω θα παρουσιαστεί μία μέθοδος που θα ακολουθείται και στις εργαστηριακές ασκήσεις:

1. Διατύπωση του προβλήματος (εάν αυτό δεν έχει διατυπωθεί)
2. Ανάλυση του προβλήματος
   a. Κατανόηση του προβλήματος (προσπάθεια κατανόησης των ζητουμένων και των δεδομένων του προβλήματος αλλά και του τρόπου επεξεργασίας των δεδομένων του προβλήματος)
   b. Καθορισμός μεταβλητών εισόδου (μεταβλητές για τα δεδομένα εισόδου του προβλήματος) και εξόδου (μεταβλητές μέσα στις οποίες παίρνουμε τα ζητούμενα του προβλήματος)
   c. Τρόποι επεξεργασίας των δεδομένων για την παραγωγή των αποτελεσμάτων(ζητούμενων).
3. Σχεδίαση του λογικού διαγράμματος(αλγόριθμου).
4. Μετατροπή του αλγόριθμου (από το λογικό διάγραμμα) σε πρόγραμμα στη γλώσσα προγραμματισμού (LOGO).
5. Εκτέλεση του προγράμματος στον Η/Υ και έλεγχος αποτελεσμάτων.

Βέβαια όταν βλέπουμε ότι κάτι δεν πάει καλά σε κάποιο από τα στάδια της μεθόδου, είμαστε υποχρεωμένοι να τα ξαναδούμε όλα από την αρχή.

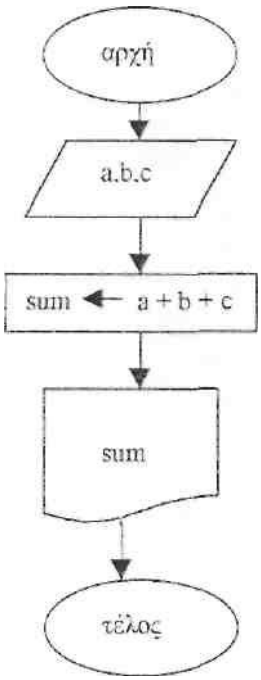
Εδώ θα δούμε ένα ολοκληρωμένο παράδειγμα επίλυσης προβλήματος με τη χρήση της παραπάνω μεθόδου:

1. Διατύπωση του προβλήματος:
   Δίνονται τρεις αριθμοί. Να βρεθεί το άθροισμα τους.

2. Ανάλυση του προβλήματος.

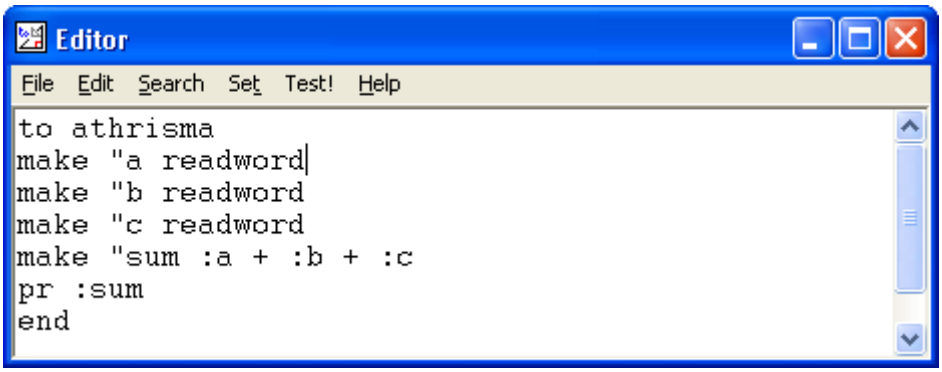
   a. Κατανόηση του προβλήματος:
      Διαβάζω και ξαναδιαβάζω το πρόβλημα για να καταλάβω τι είναι αυτό που μου ζητάει (ζητούμενα) και τι είναι αυτό που μου δίνει (δεδομένα) για να με βοηθήσει να βρω τη λύση του προβλήματος, αλλά και τι πράξεις πρέπει να γίνουν για την επίλυση του προβλήματος.
   b. Καθορισμός μεταβλητών εισόδου και εξόδου του προβλήματος:
      Μεταβλητές εισόδου: **a, b, c** (οι τρεις αριθμοί του προβλήματος)
      Μεταβλητές εξόδου: **sum** (το άθροισμα)
   c. Τρόποι επεξεργασίας των δεδομένων για την παραγωγή των αποτελεσμάτων:
      Αυτό που πρέπει να γίνει είναι να προσθέσουμε τα δεδομένα που είναι μέσα στις μεταβλητές εισόδου **a, b, c** και να βάλουμε το αποτέλεσμα μέσα στη μεταβλητή εξόδου **sum.**

3. Σχεδίαση του λογικού διαγράμματος:



Σχήμα 2

4. Μετατροπή του αλγόριθμου (από το λογικό διάγραμμα) σε πρόγραμμα στη γλώσσα προγραμματισμού (LOGO).



Σχήμα 3

5. Εκτέλεση του προγράμματος στον Η/Υ και έλεγχος αποτελεσμάτων.

Αν υποθέσουμε ότι διαβάστηκαν από το πληκτρολόγιο τα παρακάτω δεδομένα: 5, 10, 15 και έγιναν περιεχόμενα στις αντίστοιχες μεταβλητές a, b, c, μετά την εκτέλε εντολής **make "sum :a + :b + :c** η μεταβλητή sum θα έχει ως περιεχόμενο τον αριθμό 30. Μετά την εκτέλεση και της εντολής **pr :sum,** στην οθόνη του Η/Υ θα εμφανιστεί το περιεχόμενο της μεταβλητής sum (δηλ 30). Από τον έλεγχο που έγινε διαπιστώθηκε ότι το αποτέλεσμα είναι το αναμενόμενο

Βασικές δομές αλγορίθμων

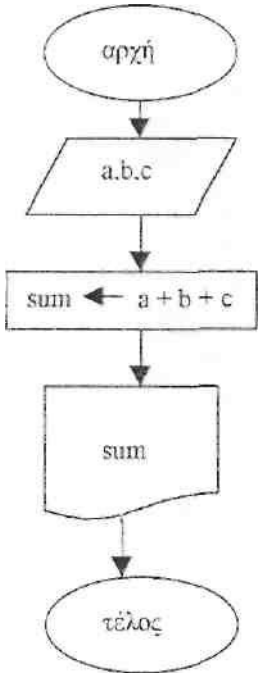
Για να γίνεται ευκολότερη η αντιμετώπιση της επίλυσης των προβλημάτων με αλγορίθμους και στη συνέχεια με προγράμματα σε κάποια γλώσσα προγραμματισμού, υπάρχουν κάποιες βασικές λογικές δομές αλγορίθμων με τον συνδυασμό των *οποίων* μπορούμε να επιλύσουμε οποιοδήποτε πρόβλημα (που έχει λύση φυσικά):

1. Η διαδοχή
2. Η επιλογή βάση του ελέγχου κάποιας συνθήκης
3. Η επαναληπτική δομή.

1. Η διαδοχή

Στη δομή της διαδοχής τα βήματα του αλγορίθμου και οι εντολές του προγράμματος εκτελούνται η μία μετά την άλλη χωρίς διακλαδώσεις και επαναλήψεις. Είναι η πιο απλή δομή.

Π.χ.

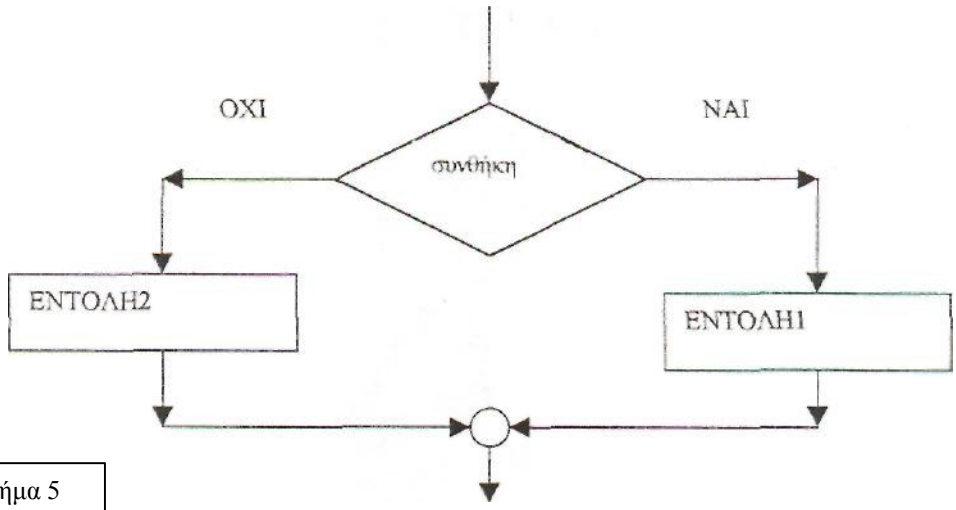


Σχήμα 4

2. Η επιλογή βάση του ελέγχου κάποιας συνθήκης

Επιλογή (Απλή επιλογή).

Στην περίπτωση αυτή έχουμε τον έλεγχο μιας συνθήκης, αν η συνθήκη ισχύει (ΝΑΙ) τότε εκτελείται η εντολή(ες)1 αν δεν ισχύει (ΟΧΙ) εκτελείται η εντολή(ες)2



Σχήμα 5

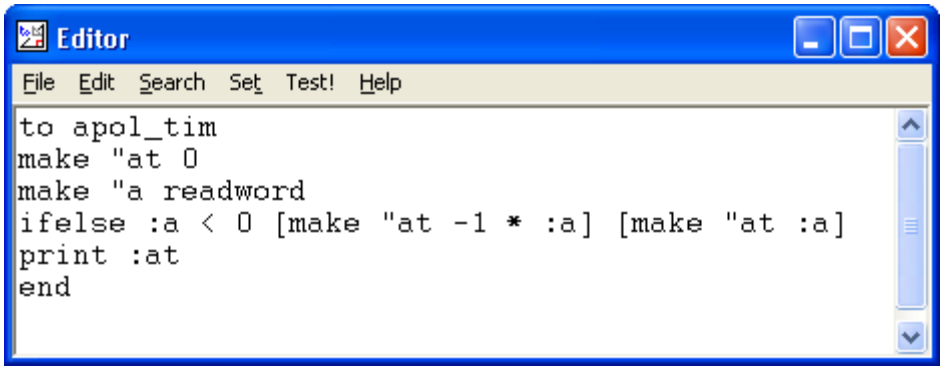
Η εντολή στη LOGO για την δομή της απλής επιλογής είναι:

ifelse συνθήκη [εντολή (ες)1] [εντολή(ες)2]

Στην περίπτωση του ΝΑΙ (όταν ισχύει η συνθήκη) εκτελείται η **εντολή (ες)1**που βρίσκεται μέσα στο πρώτο ζευγάρι αγκύλες.

Στην περίπτωση του ΟΧΙ(όταν δεν ισχύει η συνθήκη) εκτελείται η **εντολή(ες)2** που βρίσκεται μέσα στο δεύτερο ζευγάρι αγκύλες.

Παράδειγμα: Να γίνει πρόγραμμα που να υπολογίζει την απόλυτη τιμή ενός αριθμού.

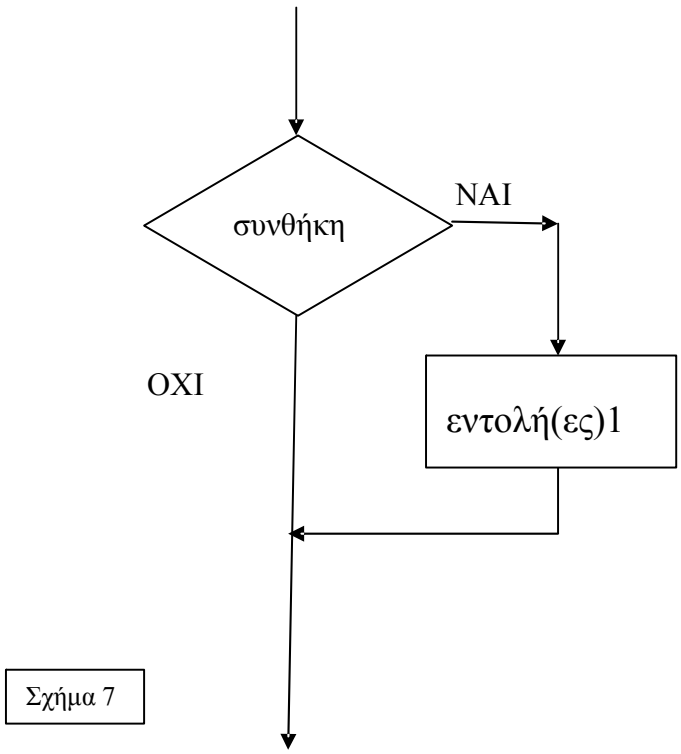


Σχήμα 6

Δίνουμε με το πληκτρολόγιο μια τιμή στη μεταβλητή a. Αν η τιμή είναι αρνητικός αριθμός τότε εκτελείται η εντολή στο πρώτο ζευγάρι αγκυλών «make “at -1 * :a»(αρνητικός[-1] * αρνητικό μας δίνει θετικό, στους θετικούς όμως δεν μπαίνει το πρόσημο + μπροστά, έτσι ο αριθμός που θα εμφανιστεί μετά στην οθόνη δεν θα έχει πρόσημο), αλλιώς εκτελείται η εντολή στο δεύτερο ζευγάρι αγκυλών «make “at :a». Τέλος εκτυπώνεται το περιεχόμενο της μεταβλητής at.

Περιορισμένη επιλογή

Στην περίπτωση αυτή έχουμε τον έλεγχο μιας συνθήκης, αν η συνθήκη ισχύει (ΝΑΙ) τότε εκτελείται η εντολή(ες)1, αλλιώς απλά αγνοείται η εντολή(ες)1 και συνεχίζεται η εκτέλεση του αλγορίθμου με την επόμενη εντολή.



Σχήμα 7

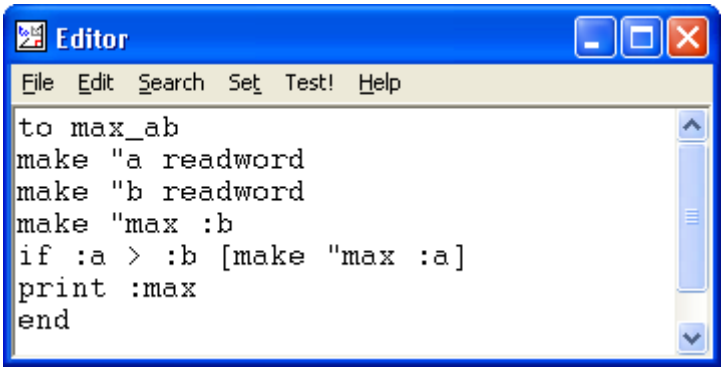
Η εντολή στη LOGO για την δομή της απλής επιλογής είναι:

if συνθήκη [εντολή (ες)1]

Στην περίπτωση του ΝΑΙ (όταν ισχύει η συνθήκη) εκτελείται η εντολή (ες)1 που βρίσκεται μέσα στις αγκύλες.

Στην περίπτωση του ΟΧΙ (όταν δεν ισχύει η συνθήκη) απλά αγνοείται η εντολή(ες)1 και συνεχίζεται η εκτέλεση του αλγορίθμου με την επόμενη εντολή.

Παράδειγμα: Να βρεθεί ο μεγαλύτερος από 2 αριθμούς.



Σχήμα 8

Δίνουμε με το πληκτρολόγιο κατά τη διάρκεια εκτέλεσης του προγράμματος max_ab, την τιμή 8 για τη μεταβλητή a και την τιμή 6 για την μεταβλητή b.

Στη συνέχεια βάζουμε στη μεταβλητή max την τιμή της μεταβλητής b, δηλαδή 6.

Στη συνέχεια ελέγχουμε με την εντολή if αν το περιεχόμενο της μεταβλητής a είναι μεγαλύτερο από αυτό της μεταβλητής b. Επειδή το 8 είναι μεγαλύτερο από το 6 εκτελείται η εντολή μέσα στις αγκύλες, δηλ η μεταβλητή max παίρνει την τιμή της μεταβλητής a (δηλ 8).

Στη συνέχεια εκτυπώνεται το περιεχόμενο της μεταβλητής max(που είναι το 8) με την εντολή print.

3. Επαναληπτική δομή.

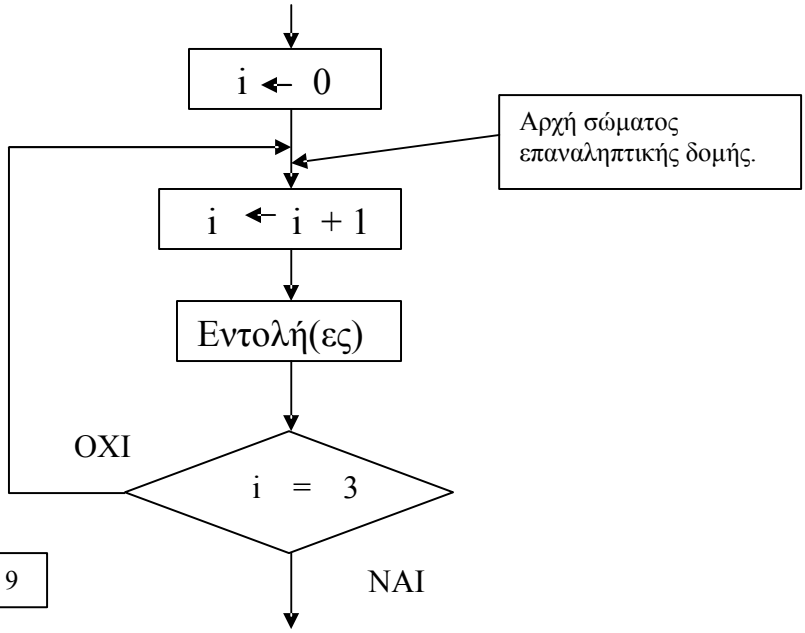
Πολλές φορές σε κάποια προβλήματα χρειάζεται να επαναληφθεί η εκτέλεση του ίδιου συνόλου εντολών πολλές φορές ώστε να δοθεί λύση στα προβλήματα αυτά με πιο αποτελεσματικό τρόπο. (θα δούμε παράδειγμα παρακάτω)

Σε αυτές τις περιπτώσεις χρησιμοποιείται η επαναληπτική δομή.

Θα γνωρίσουμε 2 επαναληπτικές δομές. Την δομή REPEAT-UNTIL και την δομή WHILE-DO.

Πριν όμως δούμε τις 2 επαναληπτικές δομές αναλυτικά, καλό είναι να μάθουμε για μια μεταβλητή που την χρησιμοποιούμε στις επαναληπτικές δομές για να μετράμε τις επαναλήψεις. Η μεταβλητή αυτή παίζει λοιπόν το ρόλο του μετρητή. Συνηθίζεται να την ονομάζουμε i ή j ή k. Γιατί όμως χρειάζεται να μετράμε τις επαναλήψεις θα το δούμε παρακάτω.

Θα δούμε λοιπόν την πρώτη δομή (REPEAT-UNTIL) και μέσα από κει θα μάθουμε πως «λειτουργεί» ο μετρητής.



Σχήμα 9

Για να δούμε λοιπόν τι συμβαίνει στο παραπάνω τμήμα λογικού διαγράμματος.

Η/Οι «Εντολή(ες)» θα εκτελείται(ούνται) όσο η συνθήκη «i = 3» δεν ισχύει (ΟΧΙ). Κάποια στιγμή η συνθήκη θα ισχύσει (ΝΑΙ) (θα δούμε αμέσως μετά πως γίνεται αυτό). Στην περίπτωση αυτή σταματά η επανάληψη της εκτέλεσης της/των «Εντολή(ες)» και συνεχίζεται η εκτέλεση του αλγόριθμου με τις τυχών υπόλοιπες εντολές μετά την επαναληπτική δομή.

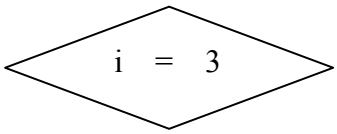
Πότε λοιπόν και πως θα ισχύσει η συνθήκη «i = 3»; Παραπάνω μιλήσαμε για μια μεταβλητή που παίζει το ρόλο του μετρητή στις επαναληπτικές δομές. Στο παραπάνω τμήμα λογικού διαγράμματος ο μετρητής είναι η μεταβλητή i. Προσέξτε λοιπόν πως χρησιμοποιείται ο μετρητής στην επαναληπτική δομή:

1) Πριν μπούμε στο σώμα της επαναληπτικής δομής δίνουμε μια αρχική τιμή στον μετρητή μας. (συνήθως η τιμή αυτή είναι το 0). Αυτό το κάνουμε για να ξέρουμε από πού θα αρχίσει το μέτρημα ο μετρητής μας. $i \leftarrow 0$

2) Η πρώτη εντολή μέσα στο σώμα της επαναληπτικής δομής είναι μια εντολή **εκχώρησης τιμής σε μεταβλητή**(γι' αυτές μιλήσαμε στη σελίδα 7) στην οποία και στα 2 μέλη υπάρχει η μεταβλητή μετρητής (i). $i \leftarrow i + 1$ Κάθε φορά που εκτελείται αυτή η εντολή στην τιμή της μεταβλητής i προστίθεται το 1 και το αποτέλεσμα γίνεται η καινούρια τιμή της μεταβλητής i. Εάν για παράδειγμα η i έχει την τιμή 0, τότε μετά την εκτέλεση της εντολής $i \leftarrow i + 1$ στην τιμή 0 προστίθεται το 1. Έτσι λοιπόν καινούριο περιεχόμενο της μεταβλητής i είναι το 1.

3) Ο μετρητής μας βρίσκεται μέσα στη συνθήκη τερματισμού της επαναληπτικής δομής «i = 3». Το 3 δείχνει πόσες φορές θα εκτελεστούν οι εντολές της επαναληπτικής δομής. Και γιατί συμβαίνει αυτό; Προσέξτε, την πρώτη φορά που θα εκτελεστούν οι εντολές της επαναληπτικής δομής ο μετρητής i θα πάρει την τιμή 1. Στο τέλος της επαναληπτικής δομής θα γίνει το ερώτημα «i = 3»; Βέβαια η απάντηση θα είναι ΟΧΙ γιατί ο μετρητής i έχει την τιμή 1. Επομένως όπως φαίνεται και στο σχήμα 9 με το βέλος που ξεκινά από το ΟΧΙ, θα ξεκινήσει ένας καινούριος κύκλος εκτέλεσης των εντολών της επαναληπτικής δομής. Η πρώτη εντολή όμως που εκτελείται σε κάθε καινούριο κύκλο είναι η $i \leftarrow i + 1$. Αυτό σημαίνει ότι ο μετρητής i τώρα θα πάρει την τιμή 2 (γιατί 1 η τιμή του μετρητή από τον προηγούμενο κύκλο + 1 μας κάνει 2). Στο τέλος αυτού του κύκλου θα γίνει πάλι το ερώτημα «i = 3»; Βέβαια η απάντηση θα είναι ΟΧΙ γιατί ο μετρητής i έχει την τιμή 2. Έτσι λοιπόν ξεκινά ένας καινούριος κύκλος. Ο μετρητής i παίρνει την τιμή 3. Στο τέλος και αυτού του κύκλου θα γίνει πάλι το ερώτημα «i = 3»; Και να που τώρα η απάντηση είναι ΝΑΙ γιατί πράγματι ο μετρητής μας έχει την τιμή 3. Έτσι λοιπόν τέρμα οι επαναλήψεις. Άντε για επαναληπτική δομή ,όπως φαίνεται και στο σχήμα 9 με το βέλος που ξεκινά από το ΝΑΙ.

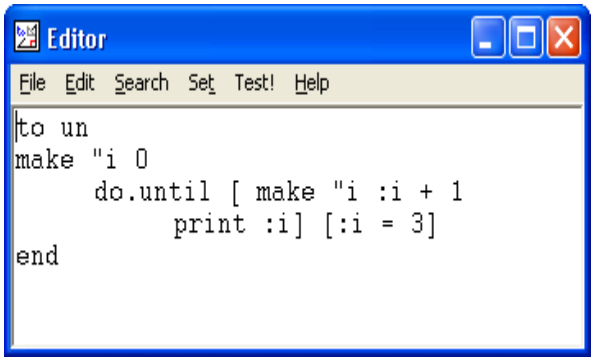
Συμπερασματικά:
-Πόσες φορές εκτελέστηκαν οι εντολές της επαναληπτικής δομής;
-Μα 3 φυσικά.
-Ποιος βοήθησε για να γίνει αυτό;
-Μα ο μετρητής i φυσικά.
-Ποια ήταν η συνθήκη ελέγχου;

- 

φυσικά.

Η εντολή στη LOGO για την επαναληπτική δομή REPEAT-UNTIL είναι:
make “i 0 (αρχική τιμή μετρητή)
do.until [make “i :i + 1 Εντολή(ες)] [:i = n] (όπου **n** είναι το πλήθος των επαναλήψεων).

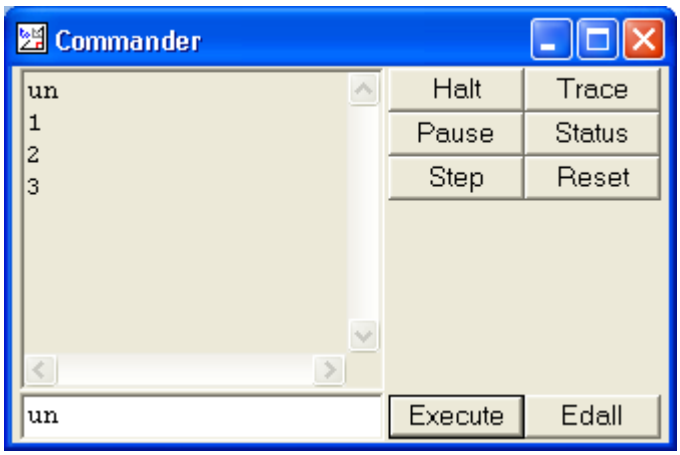
Η εντολή στη LOGO για την επαναληπτική δομή REPEAT-UNTIL εφαρμοσμένη στο σχήμα 9, είναι:



Σχήμα 10

Όπου η/οι «Εντολή(ες)» του σχήματος 9, στο παραπάνω πρόγραμμα είναι η εντολή “print :i”.

Μετά την εκτέλεση του παραπάνω προγράμματος το αποτέλεσμα θα είναι:



Σχήμα 11

Στο παρακάτω παράδειγμα θα φανεί η χρήση στις επαναληπτικές δομές και μιας άλλης κατηγορίας μεταβλητών εκτός από τους μετρητές. Αυτή είναι η κατηγορία των αθροιστών. Η λειτουργία ενός αθροιστή όπως θα δούμε στο παράδειγμα παρακάτω, είναι να προσθέτει κάθε καινούριο αριθμό στο άθροισμα των προηγούμενων.

Παράδειγμα: Κάθε μαθητής της Γ τάξης του Γυμνασίου Σορωνής δίνει ένα ποσό για το ταμείο της τάξης κάθε τρίμηνο. Το ποσό μπορεί να είναι διαφορετικό για κάθε τρίμηνο και για κάθε μαθητή. Το σύνολο των μαθητών είναι 50. Το έτος αποτελείται από 3 τρίμηνα.

Ζητείται: α) Το συνολικό ποσό που έδωσε ο κάθε μαθητής για το έτος και β) Το συνολικό ποσό που έχει το ταμείο στο τέλος του έτους.

Επειδή το παραπάνω πρόβλημα είναι πιο σύνθετο από τα προηγούμενα γιαυτό θα εφαρμόσουμε τη **μέθοδο επίλυσης προβλημάτων στον ηλεκτρονικό υπολογιστή** που γνωρίσαμε παραπάνω.

1. Διατύπωση του προβλήματος (εάν αυτό δεν έχει διατυπωθεί)

Κάθε μαθητής της Γ τάξης του Γυμνασίου Σορωνής δίνει ένα ποσό για το ταμείο της τάξης κάθε τρίμηνο. Το ποσό μπορεί να είναι διαφορετικό για κάθε τρίμηνο και για κάθε μαθητή. Το σύνολο των μαθητών είναι 50. Το έτος αποτελείται από 3 τρίμηνα.

Ζητείται: α) Το συνολικό ποσό που έδωσε ο κάθε μαθητής για το έτος και β) Το συνολικό ποσό που έχει το ταμείο στο τέλος του έτους.

2. Ανάλυση του προβλήματος

Α) Κατανόηση του προβλήματος (προσπάθεια κατανόησης των ζητουμένων και των δεδομένων του προβλήματος αλλά και του τρόπου επεξεργασίας των δεδομένων του προβλήματος)

Β) Καθορισμός μεταβλητών εισόδου (μεταβλητές για τα δεδομένα εισόδου του προβλήματος) και εξόδου (μεταβλητές μέσα στις οποίες παίρνουμε τα ζητούμενα του προβλήματος)

-Μεταβλητές εισόδου : **poso_math_tra** (το ποσό που δίνει ο κάθε μαθητής το τρίμηνο a), **poso_math_trb** (το ποσό που δίνει ο κάθε μαθητής το τρίμηνο b), **poso_math_trc** (το ποσό που δίνει ο κάθε μαθητής το τρίμηνο c).

-Μεταβλητές εξόδου: **syn_poso_math_et** (το συνολικό ποσό που έδωσε στο ταμείο ο κάθε μαθητής το έτος), **syn_poso_tam_et** (το συνολικό ποσό που έχει το ταμείο στο τέλος του έτους)

Γ) Τρόποι επεξεργασίας των δεδομένων για την παραγωγή των αποτελεσμάτων(ζητούμενων).

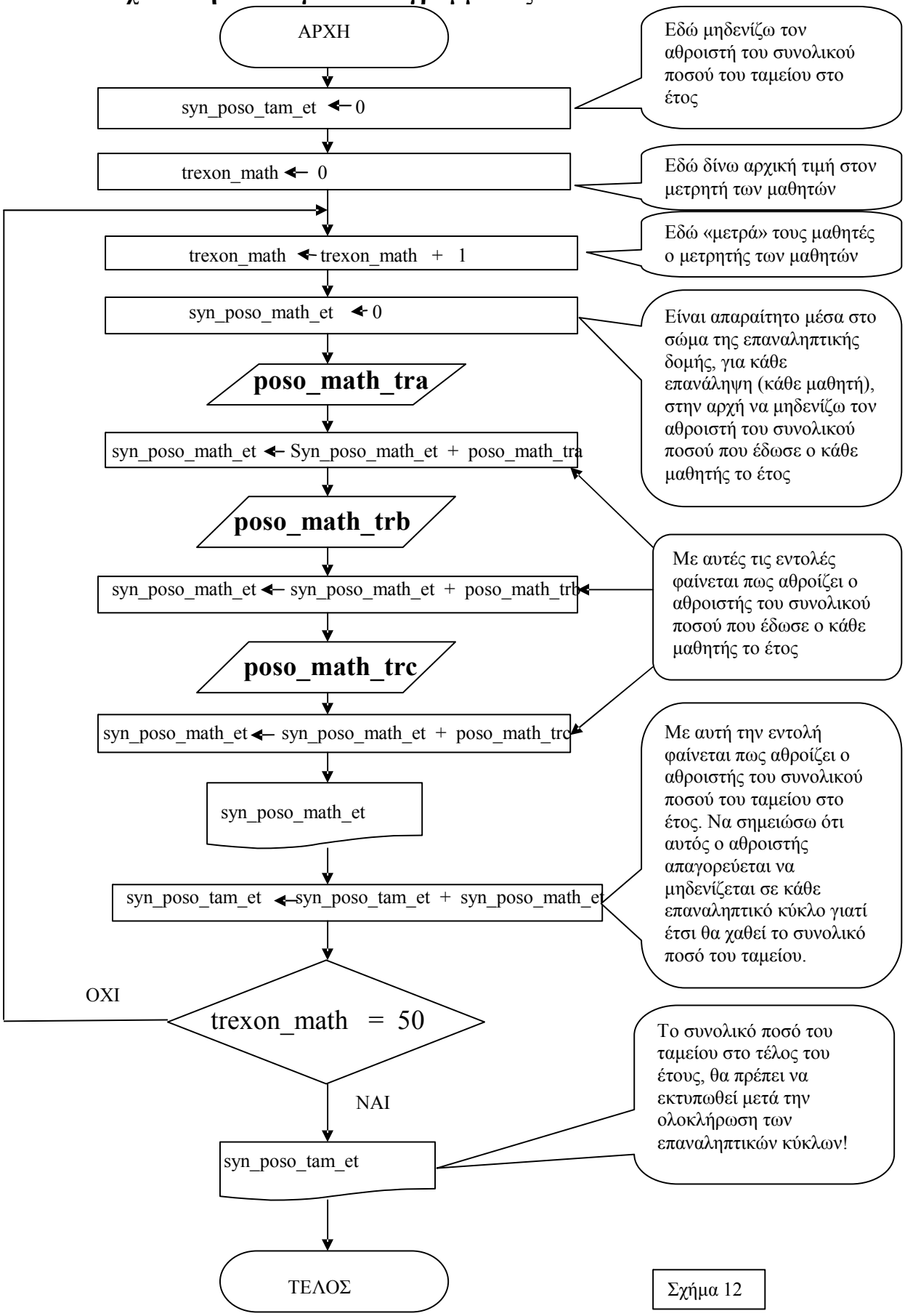
Για την επίλυση του παραπάνω προβλήματος θα χρησιμοποιήσουμε την επαναληπτική δομή. Αυτό διότι από την διατύπωση του προβλήματος προκύπτει ότι οι ίδιες ενέργειες πρέπει να εκτελεστούν και για τους 50 μαθητές.

Ετσι λοιπόν θα πρέπει να χρησιμοποιήσω έναν μετρητή για τους μαθητές. Ας του δώσω το όνομα : **trexon_math.** Δεν θα πρέπει να ξεχάσω να δώσω αρχική τιμή στον μετρητή πριν από την αρχή του σώματος της επαναληπτικής δομής **make "trexon_math 0.**

Θα χρησιμοποιήσω την επαναληπτική δομή REPEAT-UNTIL που έμαθα παραπάνω. Σε κάθε επαναληπτική δομή όμως πρέπει να υπάρχει και μια συνθήκη τερματισμού. Σ'αυτή τη συνθήκη υπάρχει και ο μετρητής. Η συνθήκη είναι: «**trexon_math = 50**»; (για 50 μαθητές)

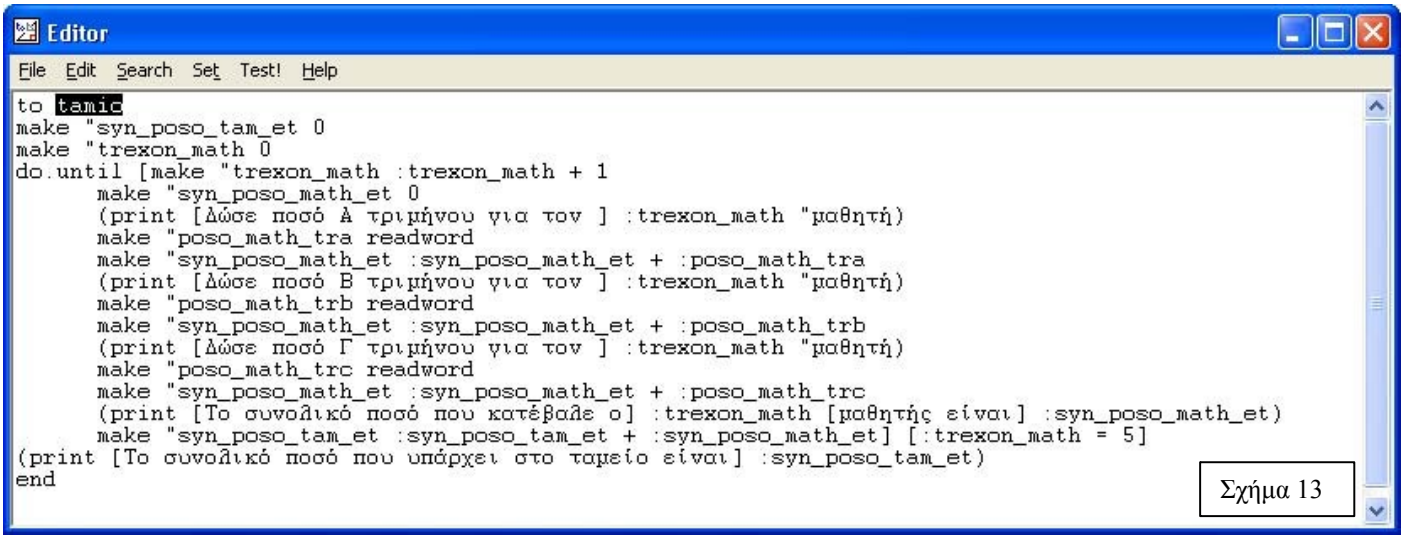
Από τα ζητούμενα του προβλήματος προκύπτει ότι θα πρέπει να χρησιμοποιήσουμε αθροιστές. Αυτό διότι μου ζητάνε συνολικά ποσά που σημαίνει επαναλαμβανόμενη αθροιστική διαδικασία. Θέλω 2 αθροιστές. Έναν για κάθε ζητούμενο. Για το συνολικό ποσό που έδωσε στο ταμείο ο κάθε μαθητής το έτος θα χρησιμοποιήσω ως αθροιστή την μεταβλητή εξόδου **syn_poso_math_et.**
Για το συνολικό ποσό που έχει το ταμείο στο τέλος του έτους θα χρησιμοποιήσω ως αθροιστή την μεταβλητή εξόδου **syn_poso_tam_et.** (για τον τρόπο λειτουργίας των αθροιστών θα δούμε παρακάτω στην πράξη). Επίσης όπως και με τους μετρητές έτσι και με τους αθροιστές θα πρέπει να μη ξεχνάμε, εάν και όποτε αυτό χρειάζεται ,να τους δίνουμε αρχική τιμή ή να τους μηδενίζουμε για καινούρια χρήση. Για παράδειγμα ο αθροιστής **syn_poso_math_et** θα πρέπει να μηδενίζεται πριν από τις ενέργειες που πρέπει να γίνουν για κάθε μαθητή (δηλαδή στην αρχή του σώματος της επαναληπτικής δομής) για να περιέχει το συνολικό ποσό για τον τρέχον μαθητή και όχι το συνολικό ποσό για τον τρέχον και όλους τους προηγούμενους.

3. Σχεδίαση του λογικού διαγράμματος:



Σχήμα 12

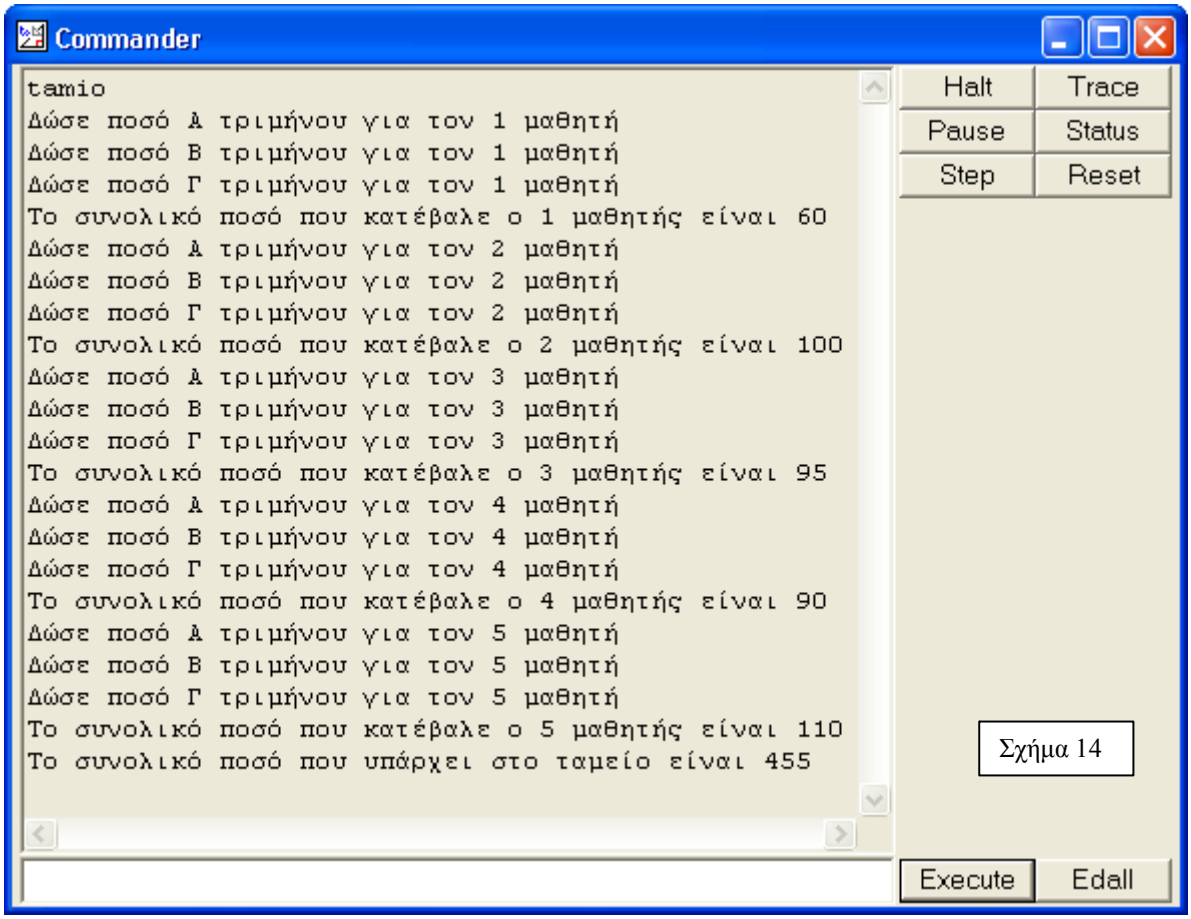
4 Μετατροπή του αλγόριθμου (από το λογικό διάγραμμα) σε πρόγραμμα στη γλώσσα προγραμματισμού (LOGO).



Σχήμα 13

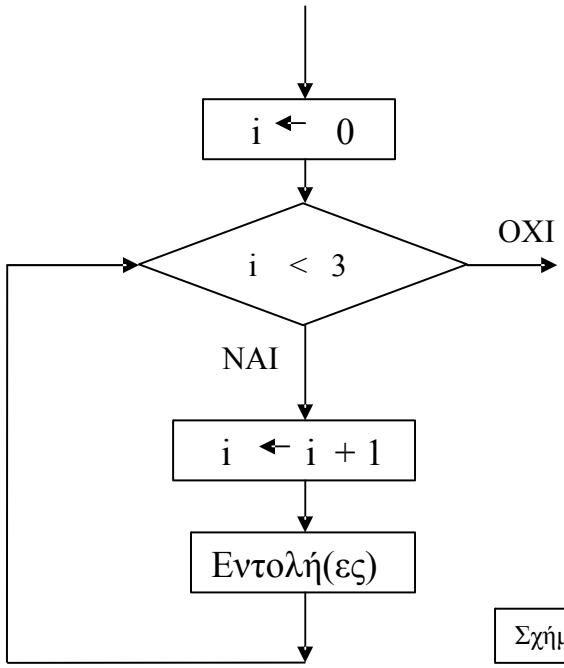
Το παραπάνω πρόγραμμα είναι για 5 μαθητές και όχι για 50. Αυτό για να μπορέσει να δοκιμαστεί στην πράξη.

5 Εκτέλεση του προγράμματος στον Η/Υ και έλεγχος αποτελεσμάτων.



Σχήμα 14

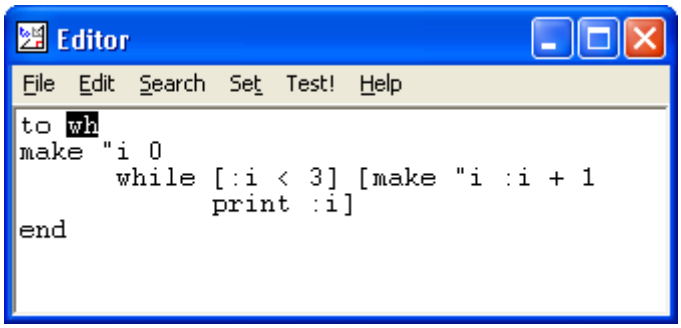
Θα δούμε τώρα τη δεύτερη επαναληπτική δομή (WHILE-DO). Ότι είπαμε παραπάνω για του μετρητές και του αθροιστές ισχύουν και σ'αυτή την περίπτωση. Η διαφορά με την προηγούμενη επαναληπτική δομή (REPEAT-UNTIL) είναι ότι η δομή αυτή επιτρέπει την επαναληπτική εκτέλεση ενός συνόλου εντολών (Εντολή(ες)), όσο ισχύει (ΝΑΙ) κάποια συνθήκη (στην περίπτωση του σχήματος 15 η συνθήκη είναι «i < 3»;). Όταν η συνθήκη πάψει να ισχύει (ΟΧΙ), τότε σταματά η επανάληψη της εκτέλεσης της/των «Εντολή(ες)» και συνεχίζεται η εκτέλεση του αλγόριθμου με τις τυχών υπόλοιπες εντολές μετά την επαναληπτική δομή



Σχήμα 15

Η εντολή στη LOGO για την επαναληπτική δομή WHILE-DO είναι:
make "i 0 (αρχική τιμή μετρητή)
while [:i < n] [make "i :i + 1 Εντολή(ες)] (όπου **n** είναι το πλήθος των επαναλήψεων).

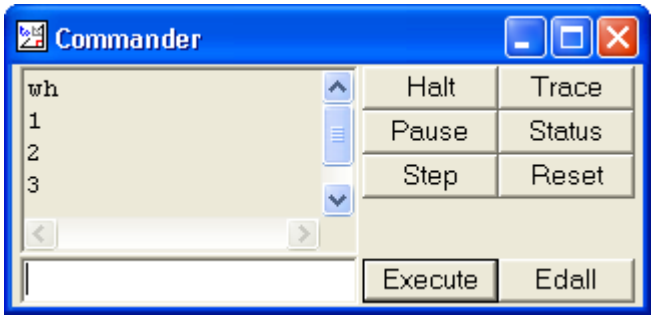
Η εντολή στη LOGO για την επαναληπτική δομή WHILE-DO εφαρμοσμένη στο σχήμα 15, είναι:



Σχήμα 16

Όπου η/οι «Εντολή(ες)» του σχήματος 15, στο παραπάνω πρόγραμμα είναι η εντολή "print :i".

Μετά την εκτέλεση του παραπάνω προγράμματος το αποτέλεσμα θα είναι:



Σχήμα 17