Φώτης Φωτόπουλος – Αριστοτέλης Χαραλαμπάκης

BASIC

ΒΑΣΙΚΕΣ ΑΡΧΕΣ
ΕΝΤΟΛΕΣ
ΑΝΑΛΥΤΙΚΑ ΛΥΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΑ
ΛΥΜΕΝΑ ΘΕΜΑΤΑ ΕΞΕΤΑΣΕΩΝ

ΑΘΗΝΑ 1996

Πρόλογος

Οι σημειώσεις αυτές γράφτηκαν για τους φοιτητές του Εθνικού Μετσοβίου Πολυτεχνείου και καλύπτουν πλήρως το μάθημα της χρήσης Ηλεκτρονικών Υπολογιστών που περιλαμβάνει τη γλώσσα προγραμματισμού BASIC.Η σειρά για τους Η/Υ περιλαμβάνει άλλα δυο βοηθήματα , τη γλώσσα προγραμματισμού Fortran 77 καθώς και την Αριθμητική Ανάλυση. Σκοπός των σημειώσεων αυτών είναι να δοθούν με σαφήνεια και απλότητα όλες οι έννοιες και οι εφαρμογές που περιέχονται στη γλώσσα BASIC διατηρώντας όμως παράλληλα την επιστημονική αυστηρότητα και ευκρίνεια που πρέπει να διέπει τέτοιες προσπάθειες.

Ο καλύτερος τρόπος για την εκμάθηση της γλώσσας αυτής είναι η ταυτόχρονη επεξεργασία των προγραμμάτων σε Η/Υ. Αν αυτό καθίσταται αδύνατο, προτείνομε να αρχίσει η εκμάθηση καταρχήν από το 1ο κεφάλαιο , το οποίο διαπραγματεύεται γενικές γνώσεις πάνω στη BASIC.Οι έννοιες που περιγράφονται είναι απαραίτητες για την ορθή κατανόηση των υπολοίπων κεφαλαίων. Κατόπιν το 2ο κεφάλαιο μπορεί να το διαβάσει κανείς συντάσσοντας ταυτόχρονα τα προγράμματα του 3ου κεφαλαίου. Το 3ο κεφάλαιο περιέχει αναλυτικότατα λυμένα παραδείγματα με τον πιο απλό και κατανοητό τρόπο. Επίσης περιέχονται ορισμένα συμπληρωματικά στοιχεία της θεωρίας. Τέλος υπάρχει και το κεφάλαιο 4, στο οποίο επεκτείνεται η χρήση της BASIC σε πολλές εφαρμογές. Το κεφάλαιο 4 χωρίστηκε σε δυο ενότητες. Στην πρώτη έχομε κατατάξει προγράμματα για όλους , πολλά από αυτά ήταν και θέματα εξετάσεων, ενώ στη δεύτερη υπάρχουν δυσκολότερα προγράμματα για όσους δεν αρκούνται στη διεθνή φοιτητική σταθερά και θέλουν το "κάτι παραπάνω". Όλα τα θέματα (πλην φυσικά αυτά των εξετάσεων) είναι πρωτότυπα και φροντίστηκε ώστε να υπάρχει ομοιογένεια στην έκφραση καθώς και στην διατύπωση για να μην δημιουργούνται προβλήματα στους αναγνώστες.

Φ. Φωτόπουλος

Α. Χαραλαμπάκης

ΠΕΡΙΕΧΟΜΕΝΑ

Συμβατότητα Προγραμμάτων

Τα προγράμματα που παρατίθενται στο φυλλάδιο αυτό είναι ελεγμένα και συνεπώς δεν παρουσιάζουν λάθη στην έκδοση της basic την GW-basic 3.02 της Microsoft .Παρόλα αυτά μπορεί σε ορισμένες εκδόσεις της basic λόγω των μικροδιαφορών των κατασκευαστριών εταιριών να υπάρχουν προβλήματα συμβατότητας των προγραμμάτων. Μπορείτε να συμβουλευτείτε τα εγχειρίδια λειτουργίας των εκδόσεων αυτών για περισσότερες πληροφορίες. Η έκδοση της basic που χρησιμοποιήθηκε είναι συμβατή με αυτή της standard basic που χρησιμοποιείται κατά τις παραδόσεις του μαθήματος στο ΕΜΠ.

Τι θα πρέπει να γνωρίζετε για τις εξετάσεις

Για επιτυχία στις εξετάσεις του μαθήματος αυτού θα πρέπει να σας είναι γνωστά τα παρακάτω:

- Διαχείριση αρχείων τυχαίας κυρίως προσπέλασης , εγγραφή και ανάγνωση.
- Εκτύπωση αποτελεσμάτων με κομψή μορφή.
- Αριθμητικές πράξεις και εντολές.
- Γενική άποψη των εντολών της basic (ελέγχου, υπορουτίνες) όπως παρουσιάζονται μέσα στο φυλλάδιο αυτό.

ΚΕΦΑΛΑΙΟ 1:ΤΑ ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ BASIC

1.1 ΤΟ ΑΛΦΑΒΗΤΟ ΤΗΣ BASIC

Η BASIC όπως και κάθε άλλη γλώσσα προγραμματισμού, διαθέτει ένα δικό της αλφάβητο για την εγγραφή των πληροφοριών(του προγράμματος, δηλ. Των οδηγιών - εντολών αλλά και των δεδομένων στον Η/Υ. Το αλφάβητο αυτό περιλαμβάνει:

- Αριθμητικούς χαρακτήρες (είναι τα δέκα ψηφία του δεκαδικού συστήματος)
- Αλφαβητικούς χαρακτήρες (είναι τα γράμματα του Αγγλικού αλφάβητου, κεφαλαία και μικρά)
- Ειδικούς χαρακτήρες (είναι τα υπόλοιπα σύμβολα που υπάρχουν στο πληκτρολόγιο π.χ. ! # $ % ^ & * ( ) > < , . ? / { } [ ] κλπ.)

Η BASIC εκτός από τα παραπάνω δέχεται και ορισμένους χαρακτήρες ελέγχου (είναι συνδυασμοί γραμμάτων του Αγγλικού αλφάβητου με το Control (Ctrl) .

Τέλος, μπορούν να χρησιμοποιηθούν και ελληνικοί χαρακτήρες (κεφαλαία και μικρά γράμματα) αλλά **μόνο** σε δεδομένα, αποτελέσματα και σχόλια (Remarks).

1.2 ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ ΣΤΗ BASIC

Για την επίλυση ενός προβλήματος, το σύνολο των οδηγιών - εντολών (δηλ. το πρόγραμμα) περιγράφει τον **Αλγόριθμο λύσης** του προβλήματος (διαδικασία επίλυσης).Για να εκτελεστεί το πρόγραμμα χρειάζεται να δώσουμε ένα σύνολο πληροφοριών, δεδομένων (data), έτσι ώστε να πάρουμε τα αποτελέσματα

από την λύση του προβλήματος. Τα δεδομένα χωρίζονται σε δύο βασικές κατηγορίες:

- Στους αριθμούς ή αριθμητικά δεδομένα (Numbers).
- Στις σειρές χαρακτήρων ή αλφαριθμητικά δεδομένα (Strings).

1.2.1 Αριθμητικά δεδομένα (Numbers)

Οι αριθμοί που αναγνωρίζει η BASIC είναι οι εξής:

i) Ακέραιοι αριθμοί του δεκαδικού συστήματος

ii) Πραγματικοί αριθμοί του δεκαδικού συστήματος

iii) Αριθμοί του δεκαεξαδικού συστήματος

iv) Αριθμοί του οκταδικού συστήματος

Οι αριθμοί μπορεί να είναι προσημασμένοι (δηλ. μπορεί να είναι θετικοί ,αρνητικοί ή μηδέν).Η υποδιαστολή των αριθμών στη BASIC δηλώνεται με μια τελεία (.) και όχι με κόμμα.

Στη BASIC δεν είναι απαραίτητο να γίνεται διάκριση μεταξύ ακεραίων και πραγματικών (όπως γίνεται π.χ. στη FORTRAN), πολλές φορές όμως θέλουμε να καθορίσουμε επακριβώς τον τύπο των αριθμών στα προγράμματα. Αυτό γίνεται όπως θα δούμε παρακάτω με χρήση συμβόλων όπως %,! και #.

Για την αποθήκευση ενός αριθμού στην μνήμη του Η/Υ κατακρατείται ένας συγκεκριμένος χώρος ,ανάλογα και με τον τύπο του αριθμού. Πρακτικά, αυτό σημαίνει ότι δεν μπορούμε να χρησιμοποιούμε αριθμούς οσοδήποτε μικρούς ή οσοδήποτε μεγάλους.

Για τους **ακέραιους** τα όρια είναι από -32768 έως +32767.Εξω από αυτά τα όρια οι αριθμοί θεωρούνται πραγματικοί απλής ακρίβειας (αρκεί να βρίσκεται μέσα στα αντίστοιχα όρια).Για τους θετικούς αριθμούς το πρόσημο (+) μπορεί να παραληφθεί, κάτι που δεν συμβαίνει με τους αρνητικούς.

Για τους **πραγματικούς αριθμούς** τα όρια είναι:

από + 2.9 x 10^{-39} έως + 1.7 x 10^{+38} για τους θετικούς και

από - 1.7 x 10^{+38} έως - 2.9 x 10^{-39} για τους αρνητικούς.

Ένας πραγματικός αριθμός μπορεί να είναι απλής ή διπλής ακρίβειας, οπότε αποθηκεύεται με 7 και 15 ψηφία ακριβείας αντίστοιχα. Όπως συμβαίνει και με τους ακεραίους, το πρόσημο (+) μπορεί να παραληφθεί.

1.2.2 Αλφαριθμητικά δεδομένα ή συμβολοσειρές (Strings)

Μια συμβολοσειρά ή string είναι μια ακολουθία χαρακτήρων που περικλείονται σε διπλά εισαγωγικά (").Οι χαρακτήρες που περιέχονται σε ένα αλφαριθμητικό μπορεί να είναι γενικά όλοι οι χαρακτήρες του κώδικα ASCII. Μέσα σε μία συμβολοσειρά όμως που περικλείεται από διπλά εισαγωγικά δεν επιτρέπεται να υπάρχουν άλλα διπλά εισαγωγικά.

Π.χ.

"Ανάληψη : 5000 δραχμές"

"4565"

"$%DFH/AZ$" κλπ.

Το μήκος μιας συμβολοσειράς είναι το πλήθος των χαρακτήρων που περιέχει, συμπεριλαμβανομένων και των κενών. Το μέγιστο μήκος μιας συμβολοσειράς είναι 255 χαρακτήρες. Επίσης υπάρχει και η μηδενική συμβολοσειρά (null string) που έχει μηδενικό μήκος και συμβολίζεται ("") (δηλ. δεν υπάρχει τίποτα μεταξύ των διπλών εισαγωγικών).

Τα αλφαριθμητικά δεδομένα γενικά δεν μπορούν να χρησιμοποιηθούν σε πράξεις όπως τα αριθμητικά. Για παράδειγμα, το "4565" δεν καταχωρείται ως αριθμός, αλλά ως ψηφία 4,5,6,5 και έτσι δεν μπορεί να χρησιμοποιηθεί σε

αριθμητικές πράξεις.(Υπάρχουν τρόποι μετατροπής "αριθμητικού" αλφαριθμητικού σε αριθμό και αντίστροφα ,όπως περιγράφεται στην συνέχεια του φυλλαδίου).Η μόνη πράξη που επιτρέπεται σε καθαυτού αλφαριθμητικά είναι η ένωσή τους, όπως περιγράφεται στην παράγραφο 1.4.2.

1.3 ΣΤΑΘΕΡΕΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ

Κάθε πρόγραμμα καθώς εκτελείται διαχειρίζεται ορισμένες ποσότητες που είτε αλλάζουν είτε παραμένουν αμετάβλητες καθ' όλη την διάρκεια εκτέλεσης. Συνεπώς έχουμε δύο είδη ποσοτήτων: τις **Σταθερές (constants)** και τις **Μεταβλητές (variables).**

1.3.1. Σταθερές (constants)

Σταθερές είναι εκείνες οι ποσότητες που παραμένουν αμετάβλητες καθ' όλη την διάρκεια εκτέλεσης του προγράμματός μας. Η BASIC αναγνωρίζει δύο τύπους σταθερών:

i) Τις αριθμητικές σταθερές

ii) Τις αλφαριθμητικές σταθερές

π.χ. στην παράσταση που υπολογίζει το εμβαδόν ενός ορθογωνίου τριγώνου με πλευρές a,b:

(a x b) / 2

είναι προφανές ότι η ποσότητα 2 είναι σταθερή ενώ οι a,b, είναι, όπως θα δούμε , μεταβλητές (κατά κάποιο τρόπο μπορούν να πάρουν οποιαδήποτε τιμή).Επίσης η παράσταση:

δεν αποτελεί παρά μια σταθερή συμβολοσειρά.

1.3.2 Μεταβλητές (Variables)

Μεταβλητές είναι εκείνες οι ποσότητες που η τιμή τους μπορεί να αλλάζει κατά την διάρκεια εκτέλεσης του. Γενικά παρίστανται με συμβολικά ονόματα που καθορίζονται από τον προγραμματιστή, έτσι ώστε να μπορεί ανά πάσα στιγμή να αναγνωρίζει, μόνο από το όνομα, τι παριστάνει κάθε μεταβλητή.

Όταν μια ποσότητα δηλωθεί ως μεταβλητή , κατακρατείται από τον Η/Υ μια συγκεκριμένη θέση στην κεντρική μνήμη. Στη θέση αυτή θα κρατείται η **τιμή** της μεταβλητής κάθε στιγμή εκτέλεσης του προγράμματος.

Είναι προφανές ότι η τιμή κάθε μεταβλητής μπορεί να αλλάζει κατά την διάρκεια εκτέλεσης, με διαδικασίες που περιγράφονται παρακάτω. Συγκεκριμένα, καταχωρώντας μια νέα τιμή σε κάποια μεταβλητή, **σβήνουμε την παλιά τιμή** αφού στην ίδια θέση της κεντρικής μνήμης του Η/Υ κατακρατείται πλέον η νέα τιμή της μεταβλητής.

Κάθε μεταβλητή ανάλογα με την τιμή την οποία δέχεται μπορεί να είναι **αριθμητική** ή **αλφαριθμητική** μεταβλητή.

Όταν μια δεδομένη χρονική στιγμή μια μεταβλητή περιέχει μόνο μία τιμή ονομάζεται **απλή μεταβλητή**. Στη BASIC όπως και σε άλλες γλώσσες προγραμματισμού υπάρχουν και μεταβλητές με δείκτες (έναν ή περισσότερους).Σ' αυτή τη περίπτωση οι μεταβλητές ονομάζονται **πίνακες** και το όνομα αυτής της

μεταβλητής παριστάνει (μεταβαλλόμενου του / των δεικτών) πολλές τιμές. Στην συνέχεια θα ασχοληθούμε με τις απλές μεταβλητές.

1.3.3 Αριθμητικές μεταβλητές

Είναι προφανές ότι μεταβλητές αυτού του είδους αντιπροσωπεύουν αριθμούς, ακέραιους ή πραγματικούς απλής ή διπλής ακριβείας. Για τον καθορισμό των ονομάτων των μεταβλητών αυτών πρέπει να τηρούνται οι παρακάτω κανόνες:

- Το όνομα μιας μεταβλητής αρχίζει με ένα από τα γράμματα του Αγγλικού αλφάβητου (κεφαλαίο ή μικρό).Στην συνέχεια μπορούμε να προσθέσουμε και αριθμούς στο όνομα.
- Κενά και ειδικοί χαρακτήρες απαγορεύονται. Μόνο ο τελευταίος χαρακτήρας μπορεί να είναι : % , ! ή #.Αυτοί οι ειδικοί χαρακτήρες χρησιμοποιούνται για να δηλώσουν ότι η μεταβλητή παριστάνει ακέραιο , απλής ακρίβειας πραγματικό και διπλής ακρίβειας πραγματικό αντίστοιχα. Σε περίπτωση που δεν χρησιμοποιηθεί κάποιος από τους παραπάνω χαρακτήρες, τότε θεωρείται ότι η πραγματική μεταβλητή είναι πραγματική απλής ακρίβειας.
- Συνήθως το μήκος του ονόματος απαγορεύεται να είναι παραπάνω από 8 χαρακτήρες.
- Επίσης δεν μπορούμε να χρησιμοποιήσουμε ως ονόματα **δεσμευμένες λέξεις** της BASIC,όπως είναι π.χ. εντολές ή ονόματα συναρτήσεων.

Οι **ακέραιες μεταβλητές** αναγνωρίζονται όπως είπαμε από το σύμβολο % στο τέλος του ονόματος. Εάν μια μεταβλητή δηλωθεί ως ακέραια αλλά της δώσουμε τιμή πραγματική τότε γίνεται στρογγυλοποίηση της πραγματικής τιμής στον πλησιέστερο ακέραιο και αποθηκεύεται κανονικά ως ακέραια μεταβλητή. Η στρογγυλοποίηση γίνεται με βάση το πρώτο δεκαδικό ψηφίο: εάν αυτό είναι 5 ή μεγαλύτερο τότε η στρογγυλοποίηση γίνεται προς τον απόλυτα μεγαλύτερο

ακέραιο ενώ εάν αυτό είναι 0,1,2,3,4 τότε απλά απορρίπτονται τα δεκαδικά ψηφία. Εάν ένας αριθμός δεν δηλωθεί ως ακέραιος αλλά του δώσουμε ακέραια τιμή π.χ. A=123 τότε αποθηκεύεται ως πραγματικός 123.0 και χρησιμοποιείται ως πραγματικός.

Οι **πραγματικές μεταβλητές απλής ακρίβειας** αναγνωρίζονται από το σύμβολο ! στο τέλος του ονόματός τους.

Οι **πραγματικές μεταβλητές διπλής ακρίβειας** αναγνωρίζονται από το σύμβολο # στο τέλος του ονόματός τους.

1.3.4 Αλφαριθμητικές μεταβλητές

Για τον καθορισμό του ονόματος μιας αλφαριθμητικής μεταβλητής ισχύουν οι κανόνες που αναφέρθηκαν παραπάνω. Η μόνη διαφορά είναι ότι ο τελευταίος χαρακτήρας μιας αλφαριθμητικής μεταβλητής είναι πάντοτε το ($) π.χ. ΟΝΟΜΑ$=”ΑΡΙΣΤΟΤΕΛΗΣ”

1.4 ΠΡΑΞΕΙΣ ΚΑΙ ΠΑΡΑΣΤΑΣΕΙΣ

1.4.1 Αριθμητικές πράξεις και παραστάσεις

Η BASIC χρησιμοποιεί τις παρακάτω βασικές αριθμητικές πράξεις:

Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/
Ύψωση σε δύναμη	^ ή **

Η σειρά, γενικά, με την οποία εκτελούνται οι πράξεις είναι: ύψωση σε δύναμη - πολλαπλασιασμοί και διαιρέσεις - προσθέσεις και αφαιρέσεις. Οι πράξεις γίνονται κατά την σειρά που υποδεικνύουν οι παρενθέσεις και κατά την γενική κατεύθυνση από αριστερά προς τα δεξιά.

1.4.2 Παραστάσεις συμβολοσειρών

Οι συμβολοσειρές χρησιμοποιούνται σε λογικές πράξεις, όχι όμως και σε αριθμητικές. Η μόνη πράξη που επιτρέπεται είναι η ένωση συμβολοσειρών, η οποία επιτυγχάνεται με το συν (+).Κατά την ένωση συμβολοσειρών έχει σημασία προφανώς η σειρά με την οποία ενώνονται οι συμβολοσειρές.

Π.χ.

```
10 A$ = ”ΕΘΝΙΚΟ”
20 B$ = ”ΜΕΤΣΟΒΙΟ”
30 C$ = “ΠΟΛΥΤΕΧΝΕΙΟ”
40 D$ = A$ + “ “ + B$ + “ “ + C$
50 E$ = C$ + “ ΕΙΝΑΙ ΜΟΝΟ ΕΝΑ!”
```

Η τιμή της D$ είναι “ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ” ενώ η τιμή της E$ είναι “ΠΟΛΥΤΕΧΝΕΙΟ ΕΙΝΑΙ ΜΟΝΟ ΕΝΑ!”

1.5 ΣΥΝΑΡΤΗΣΕΙΣ ΒΙΒΛΙΟΘΗΚΗΣ

Η BASIC έχει μια δική της βιβλιοθήκη μαθηματικών συναρτήσεων, τις οποίες μπορούμε απευθείας να χρησιμοποιούμε στα προγράμματά μας, απλά καλώντας τις με το όνομα με το οποίο τις αναγνωρίζει η BASIC .

Παρακάτω δίνεται ένας συνοπτικός πίνακας με τις κυριότερες συναρτήσεις της βιβλιοθήκης της BASIC:

Τετραγωνική ρίζα	SQR(X)
Λογάριθμος (βάση e)	LOG(X)
Εκθετική συνάρτηση	EXP(X)
Απόλυτη τιμή	ABS(X)
Ημίτονο	SIN(X)
Συνημίτονο	COS(X)
Εφαπτομένη	TAN(X)
Τόξο εφαπτομένης (ακτίνια)	ATN(X)
Ακέραιο μέρος του X	FIX(X)
Ακεραιοποίηση (στρογγύλευση)	INT(X)

ΚΕΦΑΛΑΙΟ 2: ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΤΗΣ BASIC

2.1 ΓΕΝΙΚΑ ΓΙΑ ΤΑ ΠΡΟΓΡΑΜΜΑΤΑ ΣΕ BASIC

Ένα πρόγραμμα σε γλώσσα BASIC αποτελείται από εντολές γραμμένες σε γραμμές. Κάθε γραμμή στο πρόγραμμα αρχίζει με έναν ακέραιο αριθμό (ο οποίος είναι μοναδικός σε όλο το πρόγραμμα και ακολουθεί αύξουσα σειρά). Ο αριθμός αυτός **χαρακτηρίζει** την γραμμή εντολών.

Χάρις σε αυτή την αμφιμονοσήμαντη σχέση, μπορούμε π.χ. πολύ εύκολα να μετακινούμαστε μέσα στο πρόγραμμα με απλές εντολές GOTO όπως θα δούμε. Συνηθίζεται να αριθμούμε τις γραμμές χρησιμοποιώντας πολλαπλάσια του 10.Ο λόγος είναι απλός: εάν εκ των υστέρων χρειαστεί να παρεμβάλλουμε και άλλες γραμμές εντολών στο πρόγραμμά μας, δεν θα χρειαστεί να αλλάξουμε τον αριθμό παρά σε ελάχιστες γραμμές ή και σε καμία ακόμη και θα χρησιμοποιήσουμε τους διαθέσιμους ενδιάμεσους ακεραίους.

Μετά τον αριθμό της εντολής ακολουθεί ένα κενό και στη συνέχεια ακολουθούν οι εντολές. Σε κάθε γραμμή είναι δυνατόν να υπάρχουν περισσότερες από μία εντολές αρκεί να διαχωρίζονται μεταξύ τους με την άνω και κάτω τελεία (:). Τέλος, το συνολικό μήκος της γραμμής δεν πρέπει να ξεπερνά τους 255 χαρακτήρες. Στα παρακάτω παραδείγματα (όπου υπάρχει) το " XX " αντιπροσωπεύει **τον αριθμό της γραμμής της εντολής** π.χ. 120 , 3450 κλπ.

2.2 Η ΕΝΤΟΛΗ ΑΝΤΙΚΑΤΑΣΤΑΣΗΣ LET Η' Η ΙΣΟΤΗΤΑ ΣΤΗΝ BASIC

Η εντολή LET χρησιμοποιείται ως εξής:

ΧΧ LET <ΜΕΤΑΒΛΗΤΗ> = <ΣΤΑΘΕΡΑ ή ΜΕΤΑΒΛΗΤΗ ή ΠΑΡΑΣΤΑΣΗ>

Χρησιμοποιείται για να εκχωρήσει σε μια μεταβλητή (αριθμητική ή αλφαριθμητική) μια τιμή (αριθμό ή string αντίστοιχα). Η εντολή αυτή δεν χρησιμοποιείται συχνά. Αυτό συμβαίνει διότι η παραπάνω έκφραση ισοδυναμεί με την:

ΧΧ <ΜΕΤΑΒΛΗΤΗ> = <ΣΤΑΘΕΡΑ ή ΜΕΤΑΒΛΗΤΗ ή ΠΑΡΑΣΤΑΣΗ>

Το σημείο της ισότητας (=) δεν έχει την έννοια της αλγεβρικής ισότητας (αυτό παρατηρείται σχεδόν σε όλες τις γλώσσες προγραμματισμού) .Το ίσον στην BASIC έχει την έννοια της **αντικατάστασης**. Δηλαδή στην BASIC **έχει** νόημα η - παντελώς λαθεμένη από μαθηματικής άποψης - έκφραση π.χ. Α = Α + 1. Η προηγούμενη έκφραση απλά σημαίνει ότι έχουμε νέα καταχώρηση για την τιμή της μεταβλητής Α (αντικατάσταση της παλιάς) και ότι η νέα τιμή είναι κατά 1 μεγαλύτερη της παλιάς τιμής της μεταβλητής Α. Στις περισσότερες περιπτώσεις όπου υπάρχει πρόβλημα , η παρανόηση λύνεται εάν κατά την έκφρασή μας αντικαταστήσουμε το " ίσον" με το " αντικαθίσταται από ".

Μεγάλη προσοχή χρειάζεται να τοποθετείται η προς εκχώρηση μεταβλητή στο αριστερό μέρος της ισότητας ή της εντολής LET. Μεγάλη προσοχή επίσης χρειάζεται το γεγονός ότι δεν μπορούμε να αναμίξουμε τύπους στις μεταβλητές. Αυτό σημαίνει ότι εάν η προς εκχώρηση μεταβλητή είναι δηλωμένη ως π.χ. ακέραια τότε το προϊόν του δεξιού μέλους πρέπει να είναι απαραίτητα αριθμός και όχι string.Οι παρακάτω εκφράσεις αποτελούν παραδείγματα τόσο της εντολής LET όσο και της απλής ισότητας στην BASIC (με απλή παράλειψη του LET).

ΠΑΡΑΔΕΙΓΜΑ

```
10 LET LSD! = 1456.34
```

```
20 LET PRZ% = LSD!
30 LET ALF$ = "ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ"
ή πιο απλά:
10 LSD = 1456.34
20 PRZ = LSD
30 ALF$ = "ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ"
40 K$ = ALF$
50 KOK = PRZ + 1 + (123 * (34 ^ 2))
60 L$ = K$ + " " + K$
κλπ.
```

Όπως φαίνεται και στα παραδείγματα, το δεξί μέλος μπορεί να είναι σταθερά, μεταβλητή ή και ολόκληρη παράσταση. Είναι προφανές ότι π.χ. στις γραμμές 50 - 60 **εκτελείται η παράσταση στο δεξί μέλος της ισότητας και το αποτέλεσμα αυτής εκχωρείται στην τιμή της μεταβλητής που βρίσκεται αριστερά.** Είναι πρόδηλο ότι δεν πρέπει να συγχέονται οι τύποι μεταβλητών ,αριθμητικών και αλφαριθμητικών. Έτσι π.χ. στην γραμμή 60 το δεξί μέλος βγάζει εξαγόμενο ένα αλφαριθμητικό (συγκεκριμένα το "ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ") και αποθηκεύεται στην **επίσης αλφαριθμητική μεταβλητή L$.** Όμοια στην γραμμή 50 το δεξί μέλος εξάγει αποτέλεσμα αριθμητικό. Εάν η προς εκχώρηση αριθμητική μεταβλητή είναι διαφορετικού τύπου από το εξαγόμενο του δεξιού μέλους τότε λαμβάνει χώρα η διαδικασία που περιγράφηκε στην παράγραφο 1.3.2. (π.χ. στρογγυλοποίηση πραγματικού ώστε να εκχωρηθεί ως ακέραιος κλπ.).

2.3 Η ΕΝΤΟΛΗ ΣΧΟΛΙΩΝ REM

Η γενική μορφή της εντολής REM είναι:

XX REM [ΣΧΟΛΙΑ]

Είναι μη εκτελέσιμη εντολή. Χρησιμοποιείται μόνο για την παρεμβολή σχολίων ενδιάμεσα στο πρόγραμμά μας, έτσι ώστε να γίνει πιο ευανάγνωστο και πιο εύκολο στην διόρθωση, εάν υπάρξουν τυχόν λάθη. Τα σχόλια μπορεί να είναι οτιδήποτε και γενικά ο μόνος περιορισμός είναι ο μέγιστος αριθμός χαρακτήρων ανά γραμμή (255).

Η εντολή REM απαιτεί δική της ξεχωριστή γραμμή εντολών. Εάν θελήσουμε κάποια στιγμή να γράψουμε σχόλια σε γραμμή που περιέχει και εκτελέσιμες εντολές (πράγμα σπάνιο) , γράφουμε μια απόστροφο (‘). Ο interpreter (μεταφραστής) της BASIC θα θεωρήσει οτιδήποτε **μετά** την απόστροφο ως σχόλια και θα τα αγνοήσει.

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 ‘ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ‘ Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
```

2.4 ΟΙ ΕΝΤΟΛΕΣ STOP - END

Η γενική μορφή της εντολής END είναι η εξής:

XX END

Η εντολή END είναι πάντα η τελευταία εντολή κάθε προγράμματος. Εκτέλεση αυτής σημαίνει και το τέλος του προγράμματος.

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 ‘ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
```

```
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
50 END
```

Η γενική μορφή της εντολής STOP είναι:

XX STOP

Μπορεί να υπάρχουν πολλές εντολές STOP σε πολλά σημεία του προγράμματος. Η εκτέλεση της εντολής σημαίνει ότι πρέπει να σταματήσει η εκτέλεση του προγράμματος.

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 'ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
50 STOP
60 END
```

κλπ.

(παρατηρείστε ότι θα μπορούσαμε να παρεμβάλλουμε την εντολή STOP στο προηγούμενο παράδειγμα χρησιμοποιώντας τον αριθμό εντολής π.χ. 45 , χωρίς να αλλάξουμε το 50 END σε 60 END)

2.5 Η ΕΝΤΟΛΗ ΕΛΕΓΧΟΥ RUN

Χρησιμοποιείται για την έναρξη της εκτέλεσης του προγράμματος. **Δεν αποτελεί τμήμα του προγράμματος,(άλλωστε δεν χρησιμοποιούμε αριθμό γραμμής γι' αυτή) απλά δίνει στον μεταφραστή την εντολή να εκτελέσει το πρόγραμμα που προηγουμένως φτιάχτηκε.** Γενικά χρησιμοποιείται χωρίς

ορίσματα (απλά γράφουμε RUN) οπότε εκτελείται ολόκληρο το πρόγραμμα. Εάν γράψουμε RUN XX τότε το πρόγραμμα εκτελείται από την γραμμή XX και μετά.

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 'ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
50 STOP
60 END
RUN
```

Το αποτέλεσμα θα είναι η εκτέλεση του προγράμματος μας.

2.6 ΟΙ ΕΝΤΟΛΕΣ ΕΞΟΔΟΥ PRINT

Γενικά χρησιμοποιούμε τις εντολές εξόδου για να πάρουμε τα αποτελέσματα που προκύπτουν από την εκτέλεση του προγράμματός μας (π.χ. το μήκος της υποτείνουσας του ορθογωνίου τριγώνου) και να τα εμφανίσουμε στην οθόνη. Επιπλέον, μπορούμε να τυπώσουμε οτιδήποτε άλλο θέλουμε από το πρόγραμμα όπως π.χ. ολόκληρο ή μέρος του προγράμματος, σχόλια, κείμενα, επικεφαλίδες, μεταβλητές κλπ.

Η γενική μορφή της εντολής PRINT είναι:

XX PRINT <ΣΤΑΘΕΡΕΣ, ΜΕΤΑΒΛΗΤΕΣ Ή ΚΑΙ ΠΑΡΑΣΤΑΣΕΙΣ>

Μέσα από απλά παραδείγματα θα φανεί ο τρόπος χρήσης αυτής της χρησιμότατης εντολής.

- Εκτύπωση αριθμητικής σταθεράς:

π.χ.

```
10 PRINT 234.56
```

- Εκτύπωση σταθερής συμβολοσειράς:

π.χ.

```
30 PRINT " ΥΠΟΤΕΙΝΟΥΣΑ ΤΡΙΓΩΝΟΥ"
```

- Εκτύπωση μεταβλητής:

π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 'ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
50 PRINT YPOTEIN
60 STOP
70 END
```

- Εκτύπωση παράστασης:

π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 LET A = 3 : B = 4 'ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 PRINT SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ & ΤΥΠΩΘΗΚΕ…
50 STOP
60 END
```

Αν θέλουμε με μία μόνο PRINT να εκτυπώσουμε πολλές σταθερές, πολλές μεταβλητές κλπ. τότε πρέπει να τις χωρίζουμε μεταξύ τους με ένα κόμμα (,) ή με ένα ελληνικό ερωτηματικό (;).

Οι διάφορες εκφράσεις προς εκτύπωση, όταν χωρίζονται με κόμμα εκτυπώνονται σε συγκεκριμένες θέσεις μέσα στη γραμμή , "πάχους" 14 χαρακτήρων η καθεμιά ενώ υπάρχουν 5 τέτοιες θέσεις ανά γραμμή. Αν κάποια από τις εκφράσεις δεν χωράει σε μία θέση, τότε καταλαμβάνει ολόκληρη την διπλανή κοκ. μέχρι να χωρέσει εντελώς, οπότε η επόμενη έκφραση θα τυπωθεί στην αμέσως διπλανή (ολόκληρη) θέση. Εάν δεν επαρκούν οι 5 θέσεις ανά γραμμή η εκτύπωση συνεχίζεται κανονικά στην επόμενη γραμμή. Το πρόσημο (-) ενός αρνητικού αριθμού τυπώνεται ενώ δεν συμβαίνει το ίδιο και με τους θετικούς αριθμούς.

Οι διάφορες εκφράσεις προς εκτύπωση όταν χωρίζονται με ελληνικό ερωτηματικό τότε τυπώνονται η μία δίπλα στην άλλη. Ανάμεσα στις εκφράσεις **δεν** υπάρχει κενό , εκτός από τις περιπτώσεις δυο διαδοχικών αριθμών και αριθμού - συμβολοσειράς (με αυτή τη σειρά), όπου τυπώνεται ένα κενό μεταξύ τους.

Κάθε νέα PRINT ξεκινά την εκτύπωση από την αρχή μιας νέας γραμμής. Εάν όμως βάλουμε **είτε ένα κόμμα είτε ένα ελληνικό ερωτηματικό** στο **τέλος** μιας PRINT (μετά και την τελευταία έκφραση) τότε η **επόμενη** εντολή PRINT θα συνεχίσει την εκτύπωση από την θέση (ζώνη) εκτύπωσης ή την ακριβή θέση (αντίστοιχα για το κόμμα και το ερωτηματικό) που τελείωσε η προηγούμενη PRINT.

Αν θέλουμε να παρεμβάλλουμε μια κενή γραμμή ανάμεσα στα αποτελέσματα του προγράμματός μας, τότε χρησιμοποιούμε την λεγόμενη λευκή

PRINT,δηλαδή γράφουμε απλά XX PRINT και η εκτύπωση μιας επόμενης PRINT θα συνεχιστεί στην επόμενη γραμμή, αφήνοντας την ενδιάμεση γραμμή κενή.

Τέλος πρέπει να σημειώσουμε την ύπαρξη της εντολής LPRINT η οποία είναι πανομοιότυπη με την PRINT με την μόνη διαφορά ότι στέλνει τα προς εκτύπωση όχι στην οθόνη αλλά στον εκτυπωτή.

2.7 Η ΕΝΤΟΛΗ ΕΞΟΔΟΥ PRINT USING

Όπως είδαμε, η εμφάνιση των αποτελεσμάτων με απλή χρήση της PRINT υπόκειται σε περιορισμούς. Όταν όμως θέλουμε τα (κυρίως αριθμητικά) προϊόντα της εκτύπωσης να έχουν συγκεκριμένη μορφή τότε χρησιμοποιούμε την εντολή PRINT USING η οποία έχει την γενική μορφή:

XX PRINT USING <FORMAT>; <ΣΤΑΘΕΡΕΣ ΜΕΤΑΒΛΗΤΕΣ ΚΛΠ>

Σημαντικές παρατηρήσεις είναι οι εξής:

i) Η format περικλείεται πάντοτε από διπλά εισαγωγικά (" ")

ii) Οι προς εκτύπωση εκφράσεις χωρίζονται μεταξύ τους με κόμμα η ελληνικό ερωτηματικό.

iii) Σε μια PRINT USING δεν μπορούμε να έχουμε παρά **μία** format.

Οι format που μπορούμε να χρησιμοποιήσουμε συντίθενται από ειδικά σύμβολα όπως #, , , + , - κλπ. Συνοπτικά η χρήση τους είναι η εξής:

- Το (#).Εκφράζει το κάθε ψηφίο του αριθμού. Χρησιμοποιώντας π.χ. την format ###.### δηλώνουμε ότι θέλουμε τον προς εκτύπωση αριθμό με τρία

ψηφία πριν την υποδιαστολή και τρία μετά. Εάν ο αριθμός έχει παραπάνω δεκαδικά τότε κατά τα γνωστά γίνεται στρογγυλοποίηση του αριθμού.

Πχ.

```
10 A = 123.45678
20 PRINT USING “###.###”;A
30 END
```

Το αποτέλεσμα που εμφανίζεται στην οθόνη είναι 123.457

Αν ο αριθμός είναι μεγαλύτερος από την αντίστοιχη format τότε ο αριθμός εμφανίζεται ως έχει, με ένα σύμβολο (%) στο τέλος που δηλώνει ότι η format πρέπει να μεγαλώσει. Τέλος εάν ο αριθμός έχει λιγότερα δεκαδικά από την format τότε γίνεται συμπλήρωση με μηδενικά.

- Το (+) στην αρχή ή το τέλος της format εμφανίζει στην αρχή ή το τέλος του αριθμού αντίστοιχα το (+) εάν είναι θετικός και το (-) εάν είναι αρνητικός.

  Πχ.

```
10 A = 123.45678
15 B = -34.23487
20 PRINT USING “+###.###”;A
25 PRINT USING “###.###+”;B
30 END
```

Το αποτέλεσμα που εμφανίζεται στην οθόνη είναι:

```
 +123.457
34.235-
```

2.8 Η ΕΝΤΟΛΗ ΕΙΣΟΔΟΥ INPUT

Γενικά οι εντολές εισόδου χρησιμοποιούνται για την είσοδο των δεδομένων του προβλήματος στην κεντρική μνήμη του Η/Υ. Με την εντολή INPUT επιτυγχάνεται ένας άμεσος τρόπος εισαγωγής δεδομένων από το πληκτρολόγιο και κατά την διάρκεια εκτέλεσης του προγράμματος. Έτσι με την βοήθεια της εντολής INPUT μπορούμε να αποδώσουμε τιμές σε διάφορες μεταβλητές.

Η γενική μορφή της INPUT είναι η εξής:

ΧΧ INPUT < ΛΙΣΤΑ ΜΕΤΑΒΛΗΤΩΝ >

Η λίστα μεταβλητών μπορεί να αποτελείται από ονόματα αριθμητικών ή αλφαριθμητικών μεταβλητών, χωριζόμενα μεταξύ τους με κόμμα.

Όταν η εκτέλεση του προγράμματος φτάσει σε μια εντολή INPUT τότε διακόπτεται προσωρινά η εκτέλεση και εμφανίζεται ένα ερωτηματικό (?) που αποτελεί πρόσκληση προς τον προγραμματιστή να εισάγει την τιμή ή τις τιμές των μεταβλητών που ζητούνται. **Η εισαγωγή των δεδομένων πρέπει να γίνει κατά την σειρά και με την μορφή που ζητείται από την κάθε εντολή INPUT.**Προς αποφυγή λαθών έχουμε την δυνατότητα παράλληλα με την εκτέλεση της εντολής INPUT να τυπώνουμε κάποιο βοηθητικό προς τον χρήστη μήνυμα, ώστε να μην γίνει λάθος κατά την εισαγωγή των δεδομένων. Αυτό το μήνυμα περιέχεται μέσα σε διπλά εισαγωγικά και παρεμβάλλεται της INPUT και της λίστας μεταβλητών. Μετά από το τέλος του μηνύματος ακολουθεί είτε ένα κόμμα (,) εάν θέλουμε να μην εμφανίζεται το αγγλικό ερωτηματικό κατά την εκτέλεση της INPUT,ενώ σε αντίθετη περίπτωση αντί για κόμμα βάζουμε το ελληνικό ερωτηματικό (;)

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 INPUT " ΔΩΣΕ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ Α,Β: ", Α,Β
30 LET C = A^2 + B^2
40 LET ΥΡΟΤΕΙΝ = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
45 PRINT "Η ΥΠΟΤΕΙΝΟΥΣΑ ΕΙΝΑΙ: " ;ΥΡΟΤΕΙΝ
50 STOP
60 END
```

Το αποτέλεσμα της εκτέλεσης του παραπάνω προγράμματος είναι:

```
ΔΩΣΕ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ Α,Β:
```

Πληκτρολογούμε *3 , 4* και στην συνέχεια πατάμε ENTER.Το αποτέλεσμα θα είναι το εξής:

```
Η ΥΠΟΤΕΙΝΟΥΣΑ ΕΙΝΑΙ: 5
```

Παρατηρείστε ότι δεν εμφανίστηκε το αγγλικό ερωτηματικό , κάτι που θα συνέβαινε εάν η γραμμή 20 του προγράμματος ήταν:

```
20 INPUT " ΔΩΣΕ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ Α,Β: " ; Α,Β
```

οπότε και το αποτέλεσμα θα ήταν:

```
ΔΩΣΕ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ Α,Β: ?
```

ενώ η εισαγωγή των δεδομένων θα ήταν ακριβώς ίδια.

2.9 ΟΙ ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ READ ΚΑΙ DATA.

Η γενική μορφή των εντολών αυτών είναι:

XX READ < ΛΙΣΤΑ ΜΕΤΑΒΛΗΤΩΝ >

YY DATA < ΛΙΣΤΑ ΣΤΑΘΕΡΩΝ >

Οι δύο αυτές εντολές συνδέονται άμεσα. Με την εντολή READ μπορούμε να αποδώσουμε τιμές (αριθμούς ή και strings) στις μεταβλητές που αναφέρονται στην λίστα της εντολής, οι οποίες μπορεί να είναι αριθμητικές ή και αλφαριθμητικές. **Η εντολή DATA περιέχει μια λίστα σταθερών (δεδομένων) από την οποία "τροφοδοτείται" η εντολή READ και αποδίδει τιμές στις μεταβλητές.** Οι εντολές DATA είναι γραμμένες εξαρχής στο πρόγραμμα και συνεπώς τα δεδομένα δίνονται μέσα από το πρόγραμμα, σε αντίθεση με την INPUT που είδαμε παραπάνω, με την οποία τα δεδομένα δίνονται εκτός προγράμματος.

Π.χ.

```
10 REM *ΥΠΟΛΟΓΙΣΜΟΣ ΥΠΟΤΕΙΝΟΥΣΑΣ ΟΡΘΟΓΩΝΙΟΥ ΤΡΙΓΩΝΟΥ*
20 READ A,B,D$ 'ΔΕΔΟΜΕΝΑ ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ
30 LET C = A^2 + B^2
40 LET YPOTEIN = SQR( C ) ' Η ΥΠΟΤΕΙΝΟΥΣΑ ΒΡΕΘΗΚΕ…
50 DATA 3 , 4 , "ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ"
60 END
```

Μπορείτε να συγκρίνετε το παραπάνω πρόγραμμα με αυτό που βρίσκεται στην παράγραφο 2.5. Και τα δύο κάνουν την ίδια ακριβώς εργασία. Μετά την εκτέλεση της εντολής READ η (αριθμητική) μεταβλητή Α θα πάρει την τιμή 3,η (επίσης αριθμητική) μεταβλητή Β θα πάρει την τιμή 4 ενώ το αλφαριθμητικό θα πάρει την τιμή: "ΤΑ ΜΗΚΗ ΤΩΝ ΠΛΕΥΡΩΝ".

Παρατηρήσεις:

- Τα ονόματα των μεταβλητών της READ και οι τιμές στην λίστα της DATA πρέπει να χωρίζονται με κόμμα.
- Σε ένα πρόγραμμα μπορούμε να έχουμε πολλές READ και πολλές DATA και όχι απαραίτητα ίσες στον αριθμό. Οι εντολές READ πρέπει να είναι σε σημεία μέσα στο πρόγραμμα έτσι ώστε οι μεταβλητές να παίρνουν τιμή πριν χρησιμοποιηθούν (με άλλα λόγια εκεί που θα τις τοποθετούσαμε ούτως ή άλλως ,κρατώντας μια λογική σειρά στο πρόγραμμά μας) σε αντίθεση με τις DATA που μπορούν να βρίσκονται οπουδήποτε μέσα στο πρόγραμμά μας.

- **Ο τύπος της μεταβλητής που υπάρχει στην READ πρέπει να είναι ο ίδιος με τον τύπο της (αντίστοιχης βέβαια) σταθεράς, διαφορετικά θα έχουμε λάθος.(π.χ. η αλφαριθμητική σταθερά "Ε.Μ.Π. " δεν θα μπορούσε να αντιστοιχεί στην μεταβλητή D%)**
- Γενικά ,οι σταθερές συμβολοσειρές των εντολών DATA πρέπει να περικλείονται - κατά τα γνωστά - από διπλά εισαγωγικά.
- Όταν υπάρχουν πολλές READ και μία DATA τότε η READ θα διαβάσει τα δεδομένα από την πρώτη κατά σειρά DATA και εάν εξαντληθούν τα δεδομένα αυτής, συνεχίζει να διαβάζει από την ακριβώς επόμενη DATA.
- **Εάν το πλήθος των μεταβλητών της READ είναι μεγαλύτερο από τις σταθερές που υπάρχουν στην (στις) DATA τότε σημειώνεται λάθος και τερματίζεται η εκτέλεση του προγράμματος.**
- Εάν συμβεί το αντίθετο (δηλαδή το πλήθος των μεταβλητών είναι μικρότερο από αυτό των σταθερών) ,τότε προφανώς δεν υπάρχει πρόβλημα αφού οι τελευταίες σταθερές αγνοούνται.
- Όταν σε ένα πρόγραμμα υπάρχουν πολλές READ και πολλές DATA και η πρώτη READ **δεν διαβάζει όλα τα δεδομένα** ,τότε η δεύτερη READ θα συνεχίσει την ανάγνωση ακριβώς εκεί που είχε σταματήσει η πρώτη READ.Αν πάλι και η δεύτερη READ δεν διαβάσει όλα τα δεδομένα ,συνεχίζει από το ίδιο σημείο η τρίτη READ κοκ. Με άλλα λόγια υπάρχει ένα είδος **δείκτη (pointer)** που υποδεικνύει ποιο είναι κάθε φορά το δεδομένο που πρέπει να "διαβαστεί" από την εντολή READ.

Π.χ.

```
10 READ A,G,X%
20 READ C$,D,V$
30 DATA 10.34,-100.23
40 DATA 3456.78,"E.M.Π.",-7,34.67
50 END
```

Μετά την εκτέλεση του παραπάνω προγράμματος, οι μεταβλητές θα έχουν τις παρακάτω τιμές:

```
A = 10.34
G = -100.23
X% = 3457 (Μεταπίπτουμε στην αμέσως επόμενη DATA και γίνεται ταυτόχρονη στρογγυλοποίηση του 3456.78 σε ακέραιο)
C$ = "Ε.Μ.Π."
D = -7 (Πραγματικός απλής ακρίβειας)
V$ = "34.67"
```

Παρατηρείστε ότι εάν η μεταβλητή δεν έχει δηλωθεί ως π.χ. ακέραιος τότε λαμβάνεται ως πραγματικός απλής ακρίβειας και ότι η μεταβλητή **V$ δέχεται τον "αριθμό" 34.67 αλλά αυτός αποθηκεύεται ως συμβολοσειρά 3,4,.,6,7 και όχι ως αριθμός και συνεπώς δεν μπορεί να χρησιμοποιηθεί σε αλγεβρικές πράξεις.**

2.10 Η ΕΝΤΟΛΗ RESTORE

Όπως ήδη έχουμε αναφέρει ,όταν σε ένα πρόγραμμα υπάρχουν πολλές READ και DATA , τότε **όλα** τα δεδομένα των εντολών DATA δημιουργούν ένα διατεταγμένο σε σειρά "σωρό δεδομένων" από τον οποίο παίρνουν τιμές οι εντολές READ, με βάση έναν δείκτη (pointer) ο οποίος υποδεικνύει κάθε φορά ποιο στοιχείο πρέπει να διαβαστεί. Μετακινούμενος ο δείκτης αυτός σαρώνει με τη σειρά τα δεδομένα των εντολών DATA.

Η γενική μορφή της εντολής RESTORE είναι:

XX RESTORE <YY>

όπου XX είναι κατά τα γνωστά ο αριθμός της γραμμής που βρίσκεται η εντολή RESTORE και YY **είναι ο αριθμός της γραμμής μιας εντολής DATA.** Η

εντολή RESTORE λοιπόν δεν κάνει τίποτα άλλο **από το να επαναφέρει τον δείκτη (pointer) στο πρώτο κατά σειρά δεδομένο της συγκεκριμένης εντολής DATA ,που υποδεικνύει ο αριθμός γραμμής YY.** Εάν δεν γράψουμε τον αριθμό γραμμής YY (δηλ. γράψουμε απλώς XX RESTORE) τότε ο δείκτης μεταφέρεται **στο πρώτο δεδομένο της πρώτης εντολής DATA (δηλ. στην αρχή του σωρού δεδομένων)**

Π.χ.

```
10 READ A,G,X%
20 READ C$,D
30 RESTORE 60
40 READ V$
50 DATA 10.34,-100.23
60 DATA 3456.78,"E.M.Π.",-7,34.67
70 END
```

Μετά την εκτέλεση του παραπάνω προγράμματος, οι μεταβλητές θα έχουν τις παρακάτω τιμές:

A = 10.34

G = -100.23

X% = 3457 (Μεταπίπτουμε στην αμέσως επόμενη DATA και γίνεται ταυτόχρονη στρογγυλοποίηση του 3456.78 σε ακέραιο)

C$ = "E.M.Π."

D = -7 (Πραγματικός απλής ακρίβειας)

(γίνεται επαναφορά του δείκτη στο πρώτο δεδομένο της DATA της γραμμής 60)

V$ = "3456.78"

Π.χ.

```
10 READ A,G,X%
20 READ C$,D
30 RESTORE
40 READ V$
50 DATA 10.34,-100.23
```

```
60 DATA 3456.78,"E.M.Π.",-7,34.67
70 END
```

Μετά την εκτέλεση του παραπάνω προγράμματος, οι μεταβλητές θα έχουν τις παρακάτω τιμές:

```
A = 10.34
G = -100.23
X% = 3457 (Μεταπίπτουμε στην αμέσως DATA και γίνεται
ταυτόχρονη στρογγυλοποίηση του 3456.78 σε ακέραιο)
C$ = "E.M.Π."
D = -7 (Πραγματικός απλής ακρίβειας)
(γίνεται επαναφορά του δείκτη στο πρώτο δεδομένο της
πρώτης DATA δηλ. στη γραμμή 50)
V$ = "10.34"
```

2.11 ΟΙ ΕΝΤΟΛΕΣ ΔΙΑΚΛΑΔΩΣΗΣ GOTO , ON … GOTO ΚΑΙ GOSUB.

Όπως έχουμε δει έως τώρα, ο Η/Υ εκτελεί τα προγράμματα ακολουθώντας πιστά την αύξουσα σειρά των γραμμών. Πολλές φορές όμως θέλουμε να αλλάξουμε την ροή εκτέλεσης του προγράμματος, δηλαδή θέλουμε ο Η/Υ να εκτελέσει κάποια άλλη γραμμή και όχι την επόμενη από αυτή που βρίσκεται ήδη. Αυτό είναι δυνατό να γίνει με τις **εντολές διακλάδωσης ή εντολές αλλαγής ροής.**

Η αλλαγή αυτή στη ροή μπορεί να γίνει είτε χωρίς όρους ,(δηλαδή ο Η/Υ όταν φτάσει σε μια εντολή διακλάδωσης χωρίς όρους υποχρεωτικά ακολουθεί την διακλάδωση) είτε μπορούμε να θέσουμε κάποια λογική συνθήκη ,ανάλογα με το αποτέλεσμα της οποίας να ακολουθείται και διαφορετικός "δρόμος" στην εκτέλεση του προγράμματος.

Η εντολή αλλαγής ροής χωρίς όρους έχει την γενική μορφή:

XX GOTO YY

όπου ΧΧ ο αριθμός της γραμμής της εντολής GOTO και ΥΥ ο αριθμός της γραμμής που θα εκτελεστεί μετά από την ΧΧ.Ο μόνος περιορισμός είναι ότι η ΥΥ πρέπει να είναι διαφορετική της ΧΧ.Η αλλαγή μπορεί να γίνει είτε προς τα πίσω (ΥΥ < ΧΧ) είτε προς τα εμπρός (ΥΥ > ΧΧ).Μετά την εκτέλεση της ΥΥ και εφόσον φυσικά δεν υπάρχει νέα εντολή αλλαγής ροής, θα εκτελεστεί κανονικά η επόμενη της ΥΥ γραμμή.

Τέλος μπορούμε να ομαδοποιήσουμε τις αλλαγές ροών χρησιμοποιώντας την εντολή ON … GOTO. Συγκεκριμένα η εντολή αυτή έχει την μορφή:

XX ON <K> GOTO <YY1,YY2,….,YYn>

όπου ΧΧ είναι ο αριθμός γραμμής, Κ είναι μια αριθμητική μεταβλητή και ΥΥ1,ΥΥ2 κλπ. είναι διατεταγμένη σειρά (υπαρχόντων φυσικά) αριθμών γραμμών του προγράμματός μας.

Η εντολή αυτή λειτουργεί ως εξής: Εάν η Κ είναι 1,τότε η αλλαγή ροής γίνεται προς την γραμμή ΥΥ1,εάν είναι Κ = 2 η αλλαγή ροής γίνεται προς την ΥΥ2 κοκ. Εάν Κ = m και m>n ή εάν m=0 τότε η εκτέλεση του προγράμματος συνεχίζεται στην επόμενη μετά την ΧΧ γραμμή. Εάν m<0 ή m>255 τότε το πρόγραμμα διακόπτεται και έχουμε λάθος.

Συνηθίζεται η μεταβλητή Κ να είναι ακέραιος. Η εντολή όμως ON … GOTO εκτελείται και αν η Κ είναι πραγματικός, **αφού πρώτα γίνει στρογγυλοποίηση προς ακέραιο.**

Π.χ.

```
10 READ A,G,X%
20 READ C$,D
30 RESTORE :GOTO 50
40 PRINT "ΑΥΤΗ Η ΓΡΑΜΜΗ ΔΕΝ ΕΚΤΕΛΕΙΤΑΙ ΠΟΤΕ"
50 READ V$
```

```
40 DATA 10.34,-100.23
60 DATA 3456.78,"E.M.Π.",-7,34.67
70 PRINT A,G,X%,C$,D,V$
80 END
```

Η εντολή GOSUB χρησιμοποιείται ακριβώς όπως και η GOTO (δηλ. XX GOSUB YY) με μόνη διαφορά ότι **καθώς συνεχίζεται κανονικά η εκτέλεση του προγράμματος μετά τη νέα γραμμή YY, την στιγμή που θα συναντηθεί για πρώτη φορά γραμμή της μορφής ZZ RETURN (ZZ αριθμός γραμμής) τότε η εκτέλεση του προγράμματος συνεχίζεται ("επιστρέφει") στην μετά από την XX γραμμή!**

Προσπαθήστε να ακολουθήσετε την ροή του παρακάτω προγράμματος:

```
Π.χ.
10 GOSUB 40
20 PRINT "ΑΥΤΗ Η ΓΡΑΜΜΗ ΤΩΡΑ ΕΚΤΕΛΕΙΤΑΙ !!!"
30 GOTO 60
40 PRINT "ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ"
50 RETURN
60 END
```

Το αποτέλεσμα του παραπάνω προγράμματος θα είναι:

```
ΑΥΤΟ ΕΙΝΑΙ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ
ΑΥΤΗ Η ΓΡΑΜΜΗ ΤΩΡΑ ΕΚΤΕΛΕΙΤΑΙ !!!
```

2.12 ΟΙ ΕΝΤΟΛΕΣ ΣΥΓΚΡΙΣΗΣ IF

Σε πολλές περιπτώσεις στον προγραμματισμό η εκτέλεση μιας ή περισσότερων εντολών εξαρτάται από το αποτέλεσμα μιας παράστασης ή την τιμή μιας μεταβλητής κλπ.

Π.χ. έστω ότι είχαμε την συνάρτηση:

$$f(x)=\begin{cases} x^2+1, x\geq 1 \\ x, x<1 \end{cases}$$

και θέλαμε να φτιάξουμε ένα πρόγραμμα με το οποίο εισάγοντας το x να μας δίνει το αποτέλεσμα y=f(x).Είναι προφανές ότι με κάποιο τρόπο πρέπει το πρόγραμμά μας να "καταλαβαίνει" εάν το x είναι μεγαλύτερο ή μικρότερο του 1 αλλιώς τα αποτελέσματα θα είναι σαφώς λαθεμένα. Πρέπει λοιπόν να θέσουμε μια λογική συνθήκη ,αλλιώς δεν μπορούμε να αντιμετωπίσουμε το πρόβλημα. Σύμφωνα με αυτή την **λογική συνθήκη (x < 1 ή x > 1)** το πρόγραμμά μας θα διακλαδίζεται και θα γίνεται ο υπολογισμός του αποτελέσματος με τον σωστό κάθε φορά τύπο.

Αυτή η λογική συνθήκη υλοποιείται μέσω των εντολών σύγκρισης IF.Η γενική μορφή είναι η εξής:

XX IF < ΣΥΝΘΗΚΗ > THEN < ΕΝΤΟΛΗ >

Η λειτουργία της είναι απλή: Εάν η λογική συνθήκη είναι αληθής τότε εκτελείται η < ΕΝΤΟΛΗ > αλλιώς το πρόγραμμα αγνοεί την < ΕΝΤΟΛΗ > και συνεχίζει στην επόμενη της XX γραμμή εντολών.

Η συνθήκη μπορεί να είναι μια ισότητα , μια ανισότητα κλπ. της οποίας ελέγχουμε την ορθότητα. Γενικά στις λογικές συνθήκες της BASIC χρησιμοποιούνται οι παρακάτω **τελεστές σύγκρισης:**

Μεγαλύτερο	>
Μικρότερο	<
ίσο	=
μεγαλύτερο ή ίσο	>= ή =>
μικρότερο ή ίσο	<= ή =<
διάφορο	>< ή < >

Επειδή συνήθως ο ένας κλάδος της λογικής συνθήκης (π.χ. έστω X>1 στο προηγούμενο παράδειγμα) δεν είναι δυνατόν να αναλυθεί σε μία εντολή ,συνηθίζεται μετά την λογική συνθήκη να οδηγούμαστε σε επιμέρους εντολές GOTO ,που οδηγούν σε διαφορετικά σημεία του προγράμματος όπου και αντιμετωπίζεται καλύτερα ο κάθε κλάδος.

Τέλος υπάρχει και η παρακάτω μορφή της εντολής IF:

XX IF < ΣΥΝΘΗΚΗ > THEN < ΕΝΤΟΛΗ 1 > ELSE < ΕΝΤΟΛΗ 2 >

Ο τρόπος λειτουργίας της είναι προφανής: Εάν η < ΣΥΝΘΗΚΗ > είναι αληθής τότε εκτελείται η < ΕΝΤΟΛΗ 1 > και εφόσον αυτή δεν είναι εντολή αλλαγής ροής, εκτελείται η επόμενη της XX γραμμή. Η < ΕΝΤΟΛΗ 2 > σ' αυτή την περίπτωση αγνοείται. Εάν η < ΣΥΝΘΗΚΗ > είναι ψευδής τότε αγνοείται η < ΕΝΤΟΛΗ 1 > ,εκτελείται η < ΕΝΤΟΛΗ 2 > και πάλι εφόσον αυτή δεν είναι εντολή αλλαγής ροής, εκτελείται η επόμενη της XX γραμμή. Όπως είπαμε συνηθίζεται οι < ΕΝΤΟΛΗ 1 & 2 > να είναι GOTO ή GOSUB.

Π.χ.

Το παρακάτω πρόγραμμα υπολογίζει τις τιμές της συνάρτησης f(x) για οποιοδήποτε x, με f(x):

$$f(x)=\begin{cases} x^2+1, x\geq 1 \\ x, x<1 \end{cases}$$

```
10 INPUT "ΑΡΙΘΜΟΣ X:",X
20 IF X>= 1 THEN GOTO 40
30 Y = X^2 + 1 : GOTO 50 'ΠΗΓΑΙΝΕ ΣΤΗΝ 50 ΑΠΕΥΘΕΙΑΣ
40 Y = X
50 PRINT "ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΕΙΝΑΙ:" ;Y
60 END
```

Είναι προφανές ότι η εντολή IF διακλαδισθεί το πρόγραμμά μας. Προσοχή πρέπει να δείξουμε στο ότι εάν εκτελεστεί η 30 (δηλαδή X<1) τότε **δεν πρέπει να εκτελεστεί η 40** διότι το αποτέλεσμα έχει ήδη αποδοθεί στην μεταβλητή Υ και εάν εκτελεστεί η 40 η τιμή της Υ θα αλλάξει. Γι' αυτό και με την εντολή GOTO 50 περνάμε απευθείας στην εκτύπωση των αποτελεσμάτων.

Προς εξυπηρέτησή μας, η BASIC δέχεται και ορισμένους **τελεστές σύγκρισης** ώστε να μπορούμε να συνδέουμε λογικές συνθήκες μεταξύ τους ή να αντιστρέψουμε συνθήκες(με τον τελεστή NOT). Για παράδειγμα ,έστω ότι στο προηγούμενο παράδειγμα η συνάρτηση είχε τρεις κλάδους, και ο ένας αντιστοιχούσε για Χ μεταξύ 1 και 3.Εστω ότι η περιγραφή αυτού του κλάδου γίνεται στην γραμμή 30. Τότε θα έπρεπε να γράψουμε:

```
10 IF X > 1 GOTO 20 ELSE …
20 IF X < 3 GOTO 30 ELSE …
30 REM ***ΑΝΑΛΥΣΗ ΚΛΑΔΟΥ***
κλπ.
```

Συνεπώς εάν το Χ **είναι μεταξύ των 1,3** θα πρέπει να περάσει από το πρώτο "φίλτρο" της γραμμής 10 .Στη συνέχεια οδηγείται στο δεύτερο "φίλτρο" της γραμμής 20 και εφόσον και η δεύτερη συνθήκη είναι αληθής, γνωρίζουμε πια ότι 1 < X < 3 και οδηγούμαστε στην γραμμή 30.Αντί για τα παραπάνω μπορούμε να γράψουμε:

```
10 IF X > 1 AND X < 3 GOTO 20 ELSE …
20 REM ***ΑΝΑΛΥΣΗ ΚΛΑΔΟΥ***
```

Εκτός από τον λογικό τελεστή AND έχουμε και τους τελεστές OR , NOT. Έστω ότι έχουμε δύο συνθήκες, τις Σ1 , Σ2 .Τότε ισχύουν:

Σ1	ΤΕΛΕΣΤΗΣ	Σ2	ΑΠΟΤΕΛΕΣΜΑ
ΑΛΗΘΗΣ	AND	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
ΑΛΗΘΗΣ	AND	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ
ΨΕΥΔΗΣ	AND	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ
ΨΕΥΔΗΣ	AND	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ

Σ1	ΤΕΛΕΣΤΗΣ	Σ2	ΑΠΟΤΕΛΕΣΜΑ
ΑΛΗΘΗΣ	OR	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
ΑΛΗΘΗΣ	OR	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	OR	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	OR	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ

Ο τελεστής NOT είναι πολύ πιο απλός και αντιστρέφει την συνθήκη. Για παράδειγμα θα μπορούσαμε αντί για:

```
IF X < 1 THEN …
```

Να γράψουμε:

```
IF NOT X > 1 THEN …
```

Εάν έχουμε συνδυασμό των παραπάνω, γενικά εκτελούνται πρώτα οι NOT μετά οι AND και τέλος οι OR και κατά την γενική κατεύθυνση από τα αριστερά προς τα δεξιά. Μπορούμε όμως να αλλάξουμε την σειρά υπολογισμού χρησιμοποιώντας παρενθέσεις, ακριβώς όπως στα απλά μαθηματικά.

2.13 ΟΙ ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ FOR - NEXT

Πολλές φορές χρειάζεται να εκτελεσθεί μια ομάδα εντολών αρκετές φορές. Αυτή η επανάληψη είναι δυνατή με την βοήθεια των **εντολών επανάληψης.** Για να μπορέσουμε να ελέγξουμε την επανάληψη αυτή (δηλ. για να μην έχουμε επανάληψη επ' άπειρον!) χρειαζόμαστε είτε έναν **μετρητή** είτε ένα **κριτήριο** από το οποίο θα "καταλαβαίνουμε" ότι η επανάληψη πρέπει να σταματήσει.

Οι εντολές FOR - NEXT χρησιμοποιούν έναν μετρητή (ή αλλιώς μια αριθμητική μεταβλητή ελέγχου) και είναι ιδιαίτερα εύχρηστες. Συγκεκριμένα η εντολή FOR έχει την γενική μορφή:

XX FOR <I> = <A> TO <T> STEP <B>

όπου I είναι η μεταβλητή ελέγχου , η οποία παίρνει τιμές από Α (Αρχή) έως Τ (Τέλος) με βήμα (STEP) Β. Εάν Β = 1 τότε το τμήμα STEP <B> μπορεί να παραληφθεί.

Η γενική μορφή της NEXT είναι:

XX NEXT < I >

Κάθε εντολή FOR συνδέεται με μια εντολή NEXT ως εξής:

```
XX FOR I = A TO T STEP B
……
…….(γραμμές εντολών που επαναλαμβάνονται)
…….
YY NEXT I
```

Πρακτικά , τα παραπάνω σημαίνουν ότι ο μετρητής μας (I) θα πάρει κατά την πρώτη επανάληψη την τιμή Α και θα εκτελεστούν οι γραμμές ανάμεσα στις XX,YY με το I να έχει την τιμή Α. Μόλις φτάσει η εκτέλεση του προγράμματος στην αντίστοιχη εντολή NEXT (διότι μπορούμε να έχουμε πολλές εντολές επανάληψης του τύπου FOR - NEXT) η τιμή του I αυξάνεται κατά Β και επιστρέφουμε στην επόμενη της XX γραμμής. Δηλαδή η μεταβλητή I στην επόμενη επανάληψη θα πάρει την τιμή Α + Β, την μεθεπόμενη την Α + 2*Β κοκ. και η επανάληψη επαναλαμβάνεται για όσο διάστημα η I είναι μικρότερη ή ίση του Τ. Μόλις γίνει και η τελευταία επανάληψη, η εκτέλεση συνεχίζεται από την αμέσως επόμενη της YY γραμμή.

Συνηθίζεται οι αριθμητικές μεταβλητές Α,Τ,Β να είναι ακέραιες. Σε αντίθετη περίπτωση τα πράγματα περιπλέκονται. Το βήμα Β δεν μπορεί να είναι μηδέν. Ανάμεσα σε μια FOR και μία NEXT επιτρέπεται να υπάρχουν εντολές αλλαγής ροής. Εάν η ροή οδηγείται εκτός της FOR - NEXT ο βρόχος επανάληψης ,όπως λέγεται αλλιώς η επαναληπτική αυτή διαδικασία, διακόπτεται και δεν είναι δυνατόν να συνεχίσει από το ίδιο σημείο (πρέπει να ξεκινήσει από την αρχή).Έτσι δεν επιτρέπεται να μπούμε στον κύκλο της FOR - NEXT όταν είμαστε εκτός από αυτόν, παρά μόνο εάν πάμε στην αρχή της FOR.

Μέσα σε ένα πρόγραμμα μπορούμε να έχουμε πολλαπλά FOR - NEXT**, αρκεί το ένα να βρίσκεται ολόκληρο μέσα στο άλλο.**Η εκτέλεση των πολλαπλών FOR -NEXT γίνεται από τα εσωτερικά προς τα εξωτερικά. Κάθε μία FOR θα συνδέεται με ξεχωριστή μεταβλητή ελέγχου με μια εντολή NEXT.

Π.χ. επιτρέπεται να γράψουμε:

```
10 FOR I = 1 TO 456
……..
100 FOR M = 6 TO 345 STEP 2
……
450 FOR EPAN = 50 TO 64
……
```

```
……
600 NEXT EPAN
……
700 NEXT M
……..
900 NEXT I
```

Προσέξτε ότι δεν επιτρέπεται ένα FOR - NEXT να καλύπτει μερικώς κάποιο άλλο. Έτσι είναι **σοβαρό λάθος** να γράψουμε:

```
100 FOR M = 6 TO 345 STEP 2
……
450 FOR EPAN = 50 TO 64
……
……
600 NEXT M
……
700 NEXT EPAN
```

2.14 Η ΕΝΤΟΛΗ DIM

Μέχρι τώρα ασχοληθήκαμε με απλές μεταβλητές. Όπως όμως ήδη έχει λεχθεί, μπορούμε στο όνομα μιας μεταβλητής να αντιστοιχούμε πολλές μεταβλητές, και σ' αυτή την περίπτωση για να τις ξεχωρίζουμε χρησιμοποιούμε **δείκτες.** Με άλλα λόγια ,μπορούμε να δημιουργήσουμε πίνακες. Οι πίνακες, όπως φαίνεται και στα προγράμματα των κεφαλαίων 3,4 είναι πολύ σημαντικοί.

Για την δημιουργία των πινάκων χρησιμοποιούμε την απλούστατη εντολή DIM (εκ του DIMENSION).Η γενική μορφή της εντολής αυτής είναι:

XX DIM <ΜΕΤΑΒΛΗΤΗ> (ΔΙΑΣΤΑΣΕΙΣ)

όπου μεταβλητή είναι το όνομα της μεταβλητής που επιλέξαμε για τον πίνακά μας και διαστάσεις είναι προφανώς οι διαστάσεις του (που υποδηλώνουν

τον μέγιστο αριθμό στοιχείων που μπορεί να δεχτεί).Εάν ο πίνακας είναι πολυδιάστατος, τότε οι διαστάσεις δίνονται διατεταγμένες και χωριζόμενες μεταξύ τους με κόμμα. Θα μας απασχολήσουν μόνο πίνακες μιας διάστασης (πίνακες - στήλη ή γραμμή) και πίνακες δύο διαστάσεων. Με την ίδια DIM μπορούμε να ορίσουμε παραπάνω πίνακες, χωρίζοντάς τους με κόμμα.

Είναι προφανές ότι μπορούμε να αφήσουμε κάποια στοιχεία του πίνακά μας χωρίς να τους αποδώσουμε τιμή. Τέλος, τα στοιχεία των πινάκων χρησιμοποιούνται ακριβώς όπως οι απλές μεταβλητές, με την διαφορά ότι πρέπει κάθε φορά (με την βοήθεια δεικτών) να προσδιορίζεται για ποιο στοιχείο της μεταβλητής πρόκειται.

Π.χ. ο πίνακας AVER του παραδείγματός μας είναι πίνακας μιας διάστασης με μέγιστο αριθμό στοιχείων 9 ,ενώ ο Α είναι πίνακας 3 x 3. Διαβάζουμε τα δεδομένα από την εντολή DATA της γραμμής 130 και τα αποδίδουμε αρχικά στον πίνακα AVER.Στην συνέχεια επαναφέρουμε τον δείκτη στο πρώτο στοιχείο της εντολής DATA με την εντολή RESTORE της γραμμής 60, και διαβάζουμε ξανά τα δεδομένα (χρησιμοποιώντας διπλό βρόχο επανάληψης FOR - NEXT),και αποδίδουμε τις τιμές στα στοιχεία του πίνακα Α.

```
10 DIM AVER(9) , A(3,3)
20 REM ---ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΟΝ ΠΙΝΑΚΑ AVER---
30 FOR I = 1 TO 10
40 READ AVER(I)
50 NEXT I
60 RESTORE
70 REM ---ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΟΝ ΠΙΝΑΚΑ A---
80 FOR K = 1 TO 3
90 FOR L = 1 TO 3
100 READ A(K,L)
110 NEXT L
120 NEXT K
130 DATA 1,2,3,4,5,6,7,8,9
```

```
140 END
```

Εάν μπορούσαμε να παραστήσουμε τους πίνακες AVER,A μετά την εκτέλεση του παραδείγματος ,θα ήταν ως εξής:

$$AVER = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}, A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Αποδώσαμε σχηματικά ότι η πρώτη διάσταση του A είναι η "γραμμή" ενώ η δεύτερη διάσταση είναι η "στήλη", ενώ θεωρήσαμε ότι ο πίνακας AVER είναι "πίνακας - στήλη", κάτι που δεν έχει φυσική υπόσταση στην BASIC.(ο πίνακας AVER θα μπορούσε π.χ. να παρασταθεί ως "πίνακας γραμμή". Αυτό που μας ενδιαφέρει είναι **η σειρά** των στοιχείων).Επίσης θα είχα για παράδειγμα AVER(4) = 4 , A(2,3) = 6 A(3,1) = 7 κλπ.

2.15 ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ

Για να διαχειριστούμε αρχεία (άνοιγμα , κλείσιμο , επεξεργασία , εγγραφή κλπ.) χρησιμοποιούμε διάφορες εντολές όπως:OPEN , CLOSE , LOF (LENGTH OF FILE), EOF (END OF FILE) ,INPUT ,WRITE κλπ. η χρήση των οποίων ποικίλει ανάλογα με το είδος του αρχείου που γίνεται επεξεργασία (σειριακό , τυχαίας προσπέλασης).

Η χρήση των παραπάνω εντολών και γενικότερα η διαχείριση αρχείων αναλύεται διεξοδικά στο κεφάλαιο 3 του φυλλαδίου και συγκεκριμένα στις παραγράφους 3.2 , 3.3 .

ΚΕΦΑΛΑΙΟ 3 : ΠΑΡΑΔΕΙΓΜΑΤΑ

3.1 ΑΠΛΑ ΠΑΡΑΔΕΙΓΜΑΤΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

3.1.1 Άσκηση

Υποθέτουμε ότι θέλομε να φτιάξομε ένα πρόγραμμα που να υπολογίζει τον όγκο κύβου και σφαίρας όταν δοθούν μια ακμή του και η ακτίνα της αντίστοιχα.

Λύση:

Καταρχήν η εισαγωγή των δεδομένων (στην προκειμένη περίπτωση αναφερόμαστε στην εισαγωγή της ακτίνας και της ακμής) θα γίνουν με την εντολή INPUT.

Έτσι είναι απαραίτητες οι εντολές:

```
INPUT"Ακμή κύβου =",A
```

καθώς και η

```
INPUT"Ακτίνα σφαίρας =",R
```

Υπενθυμίζεται ότι μετά την εκτέλεση των εντολών αυτών οι μεταβλητές A και R θα περιέχουν τις τιμές της ακμής του κύβου και της ακτίνας της σφαίρας.

Ο όγκος του κύβου είναι ως γνωστόν $V=a^3$ οπότε θα πρέπει να δώσουμε:

```
VK=A^3
```

Αντίστοιχα για τη σφαίρα θα έχουμε $V=\frac{4}{3}\pi R^3$,όπου π το 3,14.

Εδώ χρειάζεται ιδιαίτερη προσοχή καθόσον οι πιο πολλές εκδόσεις της BASIC δεν έχουν έτοιμο το νούμερο π, γι' αυτό πριν από την εντολή υπολογισμού του όγκου της σφαίρας πρέπει να δώσουμε:

```
PI=3.14159
```

```
και ύστερα  VS=(4/3)*PI*(R^3)
```

Τέλος πρέπει τα αποτελέσματα αυτά να εμφανίζονται στην οθόνη του Η/Υ οπότε αυτό θα γίνει με την εντολή PRINT δηλαδή:

```
PRINT"Ο όγκος του κύβου είναι:";VK
PRINT"Ο όγκος της σφαίρας είναι:";VS
```

Συνοψίζοντας παραθέτουμε το ολοκληρωμένο πλέον πρόγραμμα:

```
10 INPUT"Ακμή κύβου =",A
20 INPUT"Ακτίνα σφαίρας=",R
30 VK=A^3
40 PI=3.14159
50 VS=(4/3)*PI*(R^3)
60 PRINT"Ο όγκος του κύβου είναι:";VK
70 PRINT"Ο όγκος της σφαίρας είναι:";VS
80 END
```

(Η τελευταία εντολή είναι προαιρετική και δείχνει απλώς το τέλος του προγράμματος).

3.1.2 Άσκηση

Ας υποθέσουμε τώρα ότι μας δίνονται 20 αριθμοί , ανακατεμένοι , και ότι θέλομε να τους βάλουμε σε αύξουσα σειρά . Οι αριθμοί βρίσκονται σε μια εντολή DATA σε τυχαία σειρά.

Λύση:

Καταρχήν μόλις ακούμε την εντολή DATA σκεφτόμαστε δυο ακόμα εντολές:

- την εντολή READ με την οποία τα στοιχεία της DATA περνούν σε μια μεταβλητή
- την εντολή RESTORE με την οποία δείχνουμε στον υπολογιστή ποιος είναι ο αριθμός της εντολής που περιέχει την DATA

Αν δηλαδή υποθέσω ότι η DATA εμφανίζεται στην εντολή 100 τότε πρέπει **απαραίτητα** για τις εξετάσεις να προσθέσομε την εντολή RESTORE 100.

Για να επιλύσω το πρόβλημα αυτό θα δημιουργήσω έναν πίνακα στήλη 20x1.Αυτό γίνεται με την εντολή DIM A(20). Ο σκοπός αυτής της ενέργειας είναι απλός: έτσι συγκεντρώνω όλα τα στοιχεία σε ένα πίνακα ώστε να τα επεξεργαστώ όλα μαζί και όχι ένα κάθε φορά.

Δημιουργώ το βρόχο:

```
FOR I=1 TO 20
READ AI)
NEXT
```

Αυτός ο βρόχος είναι των 20 επαναλήψεων και διαβάζει τους 20 αριθμούς από την εντολή DATA. Την πρώτη τιμή A(1) την τοποθετεί στην 1η γραμμή. Επειδή ο πίνακας έχει μια στήλη, η επόμενη τιμή θα πάει στην 2η γραμμή κοκ, μέχρι τη συμπλήρωση του πίνακα.

Η λογική του προγράμματος θα είναι η εξής: Ας υποθέσω ότι έχω 4 αριθμούς με την σειρά: 5,1,3,2.

Ελέγχω αν το δεύτερο στοιχείο είναι μικρότερο από το πρώτο. Αν αληθεύει ανταλλάσσω τις τιμές των δηλαδή αν 1<5 τότε παίρνω τη σειρά 1,5,3,2.Ομοίως ελέγχω αν το τρίτο στοιχείο είναι μικρότερο από το δεύτερο, δηλαδή αν 5<3 οπότε επειδή αληθεύει το ανταλλάσσω παίρνοντας τη σειρά

1,3,5,2 συνεχίζω και τελικά παίρνω 1,3,2,5. Αν επαναλάβω τότε παίρνω 1,3,2,5 - 1,2,3,5 και οι υπόλοιπες αλλαγές είναι άνευ ουσίας δηλαδή δε μεταβάλλουν τη λίστα. Ας εκφράσομε αυτή τη σκέψη σε ένα πρόγραμμα BASIC. Παίρνω:

```
FOR K=1 TO 20
FOR J=1 TO 19 (γιατί συνολικά θα έχω 19*20 ελέγχους)
IF A(J+1)<A(J) THEN D=A(J+1):A(J+1)=A(J):A(J)=D
Αυτό σημαίνει ότι αν A(J+1)=12 τότε D=12 και
A(J+1)=A(J)=15 και τέλος A(J)=12.
NEXT J
NEXT K
```

Τέλος χρειάζομαι μια ρουτίνα που θα τυπώνει στην οθόνη του Η/Υ τα αποτελέσματα, δηλαδή τα στοιχεία του πίνακα Α που όμως έχουν πλέον αλλάξει.

```
FOR L=1 TO 20
PRINT"Το στοιχείο ";L;",1 είναι " ;A(L)
NEXT L
```

Γράφω τώρα ολόκληρο το πρόγραμμα:

```
10 RESTORE 100
20 FOR I=1 TO 20
30 READ A(I)
40 NEXT
50 FOR K=1 TO 20
60 FOR J=1 TO 19
65 IF A(J+1)<A(J) THEN D=A(J+1):A(J+1)=A(J):A(J)=D
70 NEXT J
80 NEXT K
85 FOR L=1 TO 20
90 PRINT"Το στοιχείο ";L;",1 είναι " ;A(L)
95 NEXT L
96 END
100 DATA 9,1,5,-
10,0,99,999,12,13,14,15,11,12,13,15,19,22,11,-9,.8,-.8
```

3.1.3 Θέμα 3ο/ Ιούνιος 1992

Δίνεται η συνάρτηση $y=x^2+5x$.Να συνταχθεί πρόγραμμα που να υπολογίζει τις τιμές του y όταν το x μεταβάλλεται από 2 ως 15, να τις τοποθετεί σε πίνακα και μετά την ολοκλήρωση του υπολογισμού να τις εκτυπώνει. Δεν υπάρχει περιορισμός στην εκτύπωση.

Λύση:

Πρέπει να προσέξουμε δυο πράγματα:

i) τη λέξη "πίνακες" που μας φέρνει στο μυαλό την εντολή DIM.

ii) τη λέξη "συνάρτηση" που είναι η εντολή DEFine FuNction της BASIC.

Δηλαδή αν υποθέσομε ότι έχομε τον πίνακα Α τότε απαραίτητα λέμε:

```
DIM A(14)
```

Γιατί το 14; Πολύ απλά γιατί θέλουμε (15-2)+1 τιμές = 14 δηλαδή για χ=2,3,…,15. Δηλώνουμε τη συνάρτηση στην εντολή DEF FN , ας την πούμε F(X), οπότε:

```
DEF FNF(X)=X^2+5*X
```

<u>Προσοχή!!!:</u> μεταξύ του FN και της F(X) δεν υπάρχει κενό.

Αυτό που εμείς λέμε στα μαθηματικά 5x+7y=9z , στη BASIC δεν υπάρχει. Πρέπει να βάζουμε το σύμβολο της εκάστοτε πράξης πχ 5*x+7*y=9*z για να το καταλάβει ο μεταφραστής (interpreter) της BASIC.Στη συνέχεια γράφουμε ένα βρόχο από ι=2 ως ι=15 που να υπολογίζει τις τιμές της F(X) καθώς το χ μεταβάλλεται από 2 ως 15.

```
FOR I=2 TO 15
S=FNF(I)
A(I-1)=S
NEXT I
```

Πρέπει να προσέξουμε τον τρόπο απόδοσης της F(I) στη μεταβλητή S. Παρεμβάλλεται η εντολή FN που λέει στο μεταφραστή ότι χρησιμοποιείται η συνάρτηση F(X) .Επίσης είπαμε A(I-1) και όχι A(I).

Αυτό οφείλεται στο γεγονός ότι για χ=2 το F(2) θα είναι το πρώτο στοιχείο του πίνακα Α δηλαδή το Α(1) ή το Α(2-1) ή το Α(I-1).Τέλος δεν παραλείπομε τη ρουτίνα εκτύπωσης :

```
FOR J=1 TO 14
PRINT A(J)
NEXT J
```

Γράφουμε τώρα όλο το πρόγραμμα:

```
 10 DIM E(14)
 20 DEF FNF(X)=X^2+5*X
 30 FOR I=2 TO 15
 40 S=FNF(I)
 50 A(I-1)=S
 60 NEXT I
 70 FOR J=1 TO 14
 80 PRINT A(J)
 90 NEXT J
100 END
```

3.1.4 Θέμα 1ο/ Ιούνιος 1991

Να δημιουργηθεί ένας πίνακας που να περιλαμβάνει 12 αριθμούς. Τα στοιχεία του πίνακα εισάγονται από το πληκτρολόγιο με δείκτες 1 ως 12.Ζητείται :

i) Να υπολογίζεται το γινόμενο των αριθμών με μονό δείκτη.

ii) Να υπολογίζεται το άθροισμα των αριθμών με ζυγό δείκτη.

iii) Να εκτυπώνονται όλοι οι αριθμοί και το τετράγωνό των.

Λύση:

Αρχικά εισαγωγή στοιχείων από το πληκτρολόγιο σημαίνει χρήση της εντολής INPUT ενώ αποθήκευση σε πίνακα με δείκτες σημαίνει χρήση της εντολής DIM. Άρα:

```
DIM A(12)
```

Τώρα κρίνεται απαραίτητο να γίνει **πρώτα** η ανάγνωση και των 12 στοιχείων και κατόπιν η επεξεργασία των. Άρα με βρόχο 12 επαναλήψεων ζητούμε από το χρήστη την εισαγωγή των δεδομένων:

```
FOR I=1 TO 12
PRINT"Εισάγετε το στοιχείο";I;" της λίστας:"
INPUT A(I)
NEXT I
```

Επειδή έχομε 12 αριθμούς, βάλαμε την εντολή PRINT να τυπώνει το δείκτη I κάθε αριθμού ώστε ο χρήστης να γνωρίζει ποιο στοιχείο εισάγει κάθε φορά.

Εν συνεχεία θα απαντήσομε το 1ο ερώτημα. Για να υπολογίσω γινόμενο αριθμών με μονό δείκτη σκέφτομαι ως εξής: θέλω το γινόμενο A(1)*A(3)*….*A(11) δηλαδή 6 αριθμούς.

Άρα ένας βρόχος της μορφής FOR J=1 TO 12 STEP +2 θα μας έδινε το επιθυμητό αποτέλεσμα.

Αν καλέσω P το γινόμενο των αριθμών παίρνω:

```
P=1
FOR J=1 TO 12 STEP+2
P=P*A(J)
NEXT J
```

Προσοχή, έδωσα P=1 γιατί αλλιώς (αν P=0) τελικά θα έπαιρνα P=0, ενώ τώρα πολλαπλασιάζεται το P με κάθε αριθμό που έχει μονό δείκτη και έτσι παίρνω

το γινόμενό των. Δε συμβαίνει όμως το ίδιο με το άθροισμα δηλαδή στο δεύτερο ερώτημα.

```
S=0
FOR K=2 TO 12  STEP +2
S=S+A(K)
NEXT K
```

Αξίζει να σταθούμε στις διαφορές των δυο βρόχων ώστε να γίνουν απόλυτα κατανοητές. Καταρχήν ο πρώτος βρόχος αρχίζει από το 1 ώστε αυξανόμενος ανά 2 να περιλάβει τους περιττούς αριθμούς δηλαδή τους 1,3,5,7,9,11. Ο δεύτερος βρόχος αρχίζει από το 2 , ώστε να περιλάβει όλους τους άρτιους 2,4,6,8,10,12. Στην αρχή έθεσα P=1 για να έχω P=1*A(1)*A(3)*… ενώ στον άλλο βρόχο έθεσα S=0 για να έχω S=0+A(2)+A(4)+… Παρατηρείστε ότι οι εντολές STEP+2 και STEP 2 είναι εντελώς ισοδύναμες.

Περνούμε στο τελευταίο ερώτημα. Όλοι οι αριθμοί είναι 12 και το τετράγωνό τους θα δίδεται από τον εξής βρόχο 12 επαναλήψεων:

```
FOR M=1 TO 12
T=A(M)^2
PRINT A(M);" με τετράγωνο ";T
NEXT M
```

Το πρώτο μέρος της εντολής PRINT τυπώνει τον αριθμό ως έχει και το δεύτερο μέρος τυπώνει το τετράγωνό του Τ. Δεν παραλείπω να εκτυπώσω και τα προηγούμενο δυο αποτελέσματα το S και το P.

```
PRINT"Το ζητούμενο γινόμενο είναι:";P
PRINT"Το ζητούμενο άθροισμα είναι:";S
```

Το πρόγραμμα είναι τώρα πλέον έτοιμο:

```
 10 DIM A(12)
 20 FOR I=1 TO 12
 30 PRINT"Εισάγετε το στοιχείο";I;" της λίστας:"
 40 INPUT A(I)
 50 NEXT I
 60 P=1
 70 FOR J=1 TO 12 STEP+2
 80 P=P*A(J)
 90 NEXT J
100 S=0
110 FOR K=2 TO 12  STEP +2
120 S=S+A(K)
130 NEXT K
140 PRINT"Το ζητούμενο γινόμενο είναι:";P
150 PRINT"Το ζητούμενο άθροισμα είναι:";S
160 FOR M=1 TO 12
170 T=A(M)^2
180 PRINT A(M);" με τετράγωνο ";T
190 NEXT M
200 END
```

3.2 ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΡΧΕΙΩΝ ΣΕΙΡΙΑΚΗΣ ΠΡΟΣΠΕΛΑΣΗΣ

3.2.0 Επιγραμματικά

Πολλές φορές (ή μάλλον τις περισσότερες) επιθυμούμε να φυλάσσονται δεδομένα προγραμμάτων μετά το κλείσιμο του Η/Υ. Αυτό γίνεται με αποθήκευσή των εις αρχεία. Τα αρχεία τα χωρίζομε σε δυο μεγάλες κατηγορίες. Η πρώτη ονομάζεται αρχεία σειριακής προσπέλασης και η δεύτερη τυχαίας προσπέλασης. Γιατί υπάρχουν δυο είδη αρχείων ; Διότι καθένα έχει τα πλεονεκτήματά του καθώς και τα μειονεκτήματά του. Τα αρχεία σειριακής προσπέλασης είναι εν γένει ευκολότερα για τον προγραμματιστή. Επιτρέπουν την αποθήκευση δεδομένων

οποιασδήποτε μορφής και μήκους. Πλην όμως για να διαβαστούν πρέπει να έχουν προηγουμένως διαβαστεί όλα τα δεδομένα που έχουν εγγραφεί πριν από αυτά.

Για παράδειγμα για να διαβαστεί το 1000ο στοιχείο πρέπει να διαβαστούν πριν τα πρώτα 999 στοιχεία!. Από την άλλη σε ένα αρχείο τυχαίας προσπέλασης μπορώ να διαβάσω οιαδήποτε πληροφορία ανεξάρτητα θέσης εγγραφής. Όμως όλες οι εγγραφές πρέπει απαραίτητα να έχουν το ίδιο μήκος. Δηλαδή αν αποθηκεύομε ονόματα των 20 γραμμάτων και συναντήσομε ένα των 11, προσθέτομε 9 κενούς χαρακτήρες για να γίνει 20. Αν όμως συναντήσομε ένα των 22 χαρακτήρων τότε αποθηκεύομε μόνον τους πρώτους 20.

Στα θετικά των αρχείων τυχαίας προσπέλασης μπορούμε να προσμετρήσομε την κατάληψη λιγότερου χώρου επί του μέσου αποθήκευσης από τα αρχεία σειριακής προσπέλασης λόγω του τρόπου εγγραφής των δεδομένων.

3.2.1 Άσκηση

Γράψτε ένα πρόγραμμα που να αποθηκεύει σε αρχείο σειριακής προσπέλασης ένα αριθμητικό και ένα αλφαριθμητικό στοιχείο που εισάγονται από το πληκτρολόγιο.

Λύση:

Διαβάζοντας τη λέξη "αρχείο σειριακό" μας έρχονται στο μυαλό οι εντολές διαχείρισης των αρχείων αυτών δηλαδή οι OPEN, CLOSE, WRITE.

Καταρχήν πρέπει να ρωτήσομε το χρήστη ποιο είναι το αριθμητικό και ποιο το αλφαριθμητικό στοιχείο. Αυτό γίνεται όπως ήδη έχομε γνωρίσει με την εντολή INPUT.Δηλαδή:

```
INPUT" Εισάγετε ένα αριθμητικό στοιχείο";NUM
INPUT" Εισάγετε ένα αλφαριθμητικό στοιχείο ";ALF$
```

Προσοχή, είναι απαραίτητο για αλφαριθμητικά στοιχεία να προστίθεται το σύμβολο $ ενώ για αριθμητικά να μην χρησιμοποιείται. Για να αποθηκευτούν σε αρχείο πρέπει αρχικά να ανοιχτεί. Αυτό γίνεται με την εντολή OPEN χωρίς να μας ενδιαφέρει το κανάλι ροής που θα το θέτομε πάντα ίσο με 1 , εκτός και αν λέει αλλιώς η άσκηση.

```
OPEN" OUTFILE.DAT" FOR OUTPUT AS #1
```

Το όνομα του αρχείου επιλέχθηκε τυχαία. Για να πούμε στον υπολογιστή ότι θα γράψομε στοιχεία στο αρχείο αυτό βάλαμε το FOR OUTPUT ενώ επίσης τυχαία διαλέξαμε το κανάλι ροής δεδομένων 1.

Επόμενο βήμα είναι να μεταφέρω τις τιμές με την εντολή WRITE των στοιχείων στο αρχείο που μόλις ανοίχτηκε. Προσοχή στο κανάλι ροής που πρέπει να είναι ίδιο με αυτό που αναγράφεται στην εντολή OPEN.

```
WRITE #1,NUM
WRITE #1,ALF$
```

ή μαζί και τα δυο `WRITE #1,NUM,ALF$`

Τώρα που τελείωσαν τα στοιχεία, πρέπει να πούμε στον Η/Υ να κλείσει το αρχείο , άρα προσθέτομε την εντολή CLOSE #1. Συνολικά το πρόγραμμα έχει ως εξής:

```
10 INPUT" Εισάγετε ένα αριθμητικό στοιχείο";NUM
20 INPUT"  Εισάγετε ένα αλφαριθμητικό στοιχείο ";ALF$
30 OPEN" OUTFILE.DAT" FOR OUPUT AS #1
40 WRITE #1,NUM,ALF$
50 CLOSE #1
60 END
```

3.2.2 Επέκταση της 3.2.1

Με τα δεδομένα του προηγούμενου προγράμματος, να συνταχθεί πρόγραμμα που να διαβάζει τα δεδομένα που γράφτηκαν στο αρχείο του 3.2.1.

Λύση:

Πρέπει να προσέχουμε κατά την ανάγνωση δεδομένων από αρχεία τυχαίας προσπέλασης , να μην διαβάσουμε αριθμητική μεταβλητή ενώ στα δεδομένα έχει εγγραφεί αλφαριθμητική ή το αντίστροφο. Δηλαδή πρέπει να τηρηθεί αυστηρά η σειρά που γράφτηκαν τα στοιχεία. Έτσι πρώτα θα διαβάσομε την αριθμητική και κατόπιν την αλφαριθμητική για το συγκεκριμένο πάντα πρόγραμμα. Πρώτα από όλα όμως πρέπει να ανοίξουμε το αρχείο για ανάγνωση, άρα:

```
OPEN"OUTFILE.DAT" FOR INPUT AS #1
```

Το κανάλι ροής συνεχίζομε να το παίρνομε τυχαία. Δηλαδή αν αντί για 1 είχα 2 ή 3 θα έπαιρνα ακριβώς το ίδιο αποτέλεσμα. Στη συνέχεια διαβάζω τα δεδομένα:

```
INPUT #1,A,A$
```

Τώρα στη μεταβλητή A θα αποδοθεί η τιμή που είχε η μεταβλητή NUM κατά την εγγραφή του αρχείου. Δηλαδή δεν έχει σημασία αν θα χρησιμοποιήσομε τα ίδια ονόματα για τα δεδομένα κατά την ανάγνωση με αυτά που χρησιμοποιήσαμε κατά την εγγραφή. Έτσι η A$ θα έχει την τιμή της ALF$.

Κατόπιν λέμε στον υπολογιστή ότι τελειώσαμε με το αρχείο και άρα μπορεί να το κλείσει καθόσον δε θα το χρειαστούμε άλλο: CLOSE #1. Δεν ξεχνάμε να εκτυπώσουμε στην οθόνη τα αποτελέσματα:

```
PRINT ”Η αριθμητική μεταβλητή έχει τιμή:”;A
PRINT ”Η αλφαριθμητική μεταβλητή έχει τιμή:”;A$
```

Συγκεντρωτικά το πρόγραμμα είναι:

```
10 OPEN”OUTFILE.DAT” FOR INPUT AS #1
20 INPUT #1,A,A$
30 CLOSE #1.
40 PRINT ”Η αριθμητική μεταβλητή έχει τιμή:”;A
50 PRINT ”Η αλφαριθμητική μεταβλητή έχει τιμή:”;A$
60 END
```

3.2.3 Άσκηση

Να δημιουργηθεί αρχείο τυχαίας προσπέλασης που να αποθηκεύονται όλα τα στοιχεία ενός πίνακα Α διαστάσεως 5x3.

Λύση:

Όπως και στα προηγούμενα παραδείγματα επιλέγομε ένα τυχαίο όνομα για το αρχείο καθώς και έναν αριθμό για το κανάλι ροής πχ το 2.

```
OPEN”DATAF.OUT” FOR OUTPUT AS #2
```

Τώρα πρέπει να δημιουργήσομε ένα βρόχο που να παίρνει κάθε στοιχείο του Α και να το αποθηκεύει στο αρχείο:

```
FOR I=1 TO 5
FOR J=1 TO 3
WRITE #2,A(I,J)
NEXT J
NEXT I
```

Παρατηρήστε ότι δεν υπάρχει περιορισμός αν κατά την αποθήκευση πρέπει να εισαχθούν τα δεδομένα κατά γραμμές ή κατά στήλες. Έτσι (τυχαία) επιλέχθηκε η αποθήκευση κατά γραμμές, πράγμα όμως ιδιαίτερα σημαντικό κατά την ανάγνωση των δεδομένων. Το ολικό πρόγραμμα έχει ως εξής:

```
10 OPEN”DATAF.OUT” FOR OUTPUT AS #2
20 FOR I=1 TO 5
30 FOR J=1 TO 3
40 WRITE #2,A(I,J)
50 NEXT J
60 NEXT I
70 CLOSE #2
80 END
```

Ας υποθέσομε ότι το θέμα μας απαιτούσε να διαβαστούν τα στοιχεία αυτά και να τοποθετηθούν σε ένα άλλο πίνακα πχ τον Β. Τώρα αφού ο Β δεν είναι έτοιμος αλλά θα τον δημιουργήσομε εμείς επικαλούμαστε την εντολή DIM έχομε δηλαδή τα εξής:

```
10 DIM B(5,3)
20 OPEN”DATAF.OUT” FOR INPUT AS #2
30 FOR I=1 TO 5
40 FOR J=1 TO 3
50 INPUT#2,B(I,J)
60 NEXT J
70 NEXT I
80 CLOSE #2
90 END
```

3.3 ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΡΧΕΙΩΝ ΤΥΧΑΙΑΣ ΠΡΟΣΠΕΛΑΣΗΣ

3.3.0 Επιγραμματικά

Τα αρχεία τυχαίας προσπέλασης δεν είναι τόσο εύκολα δια τον προγραμματιστή. Για το λόγο αυτό κρίνομε σκόπιμο να κάνομε μια παρένθεση για να εξετάσομε λεπτομερέστερα τη δομή και λειτουργία των αριθμών κατά τη BASIC που έχει να κάνει με τον τρόπο που "σκέφτεται" και "ενεργεί" ένας Η/Υ.

3.3.1 Τι σημαίνει η "ακρίβεια" στους αριθμούς;

Οι αριθμητικές τιμές αποθηκεύονται στη BASIC ως ακέραιοι, απλής ακρίβειας ή διπλής ακρίβειας. Οι ακέραιοι αριθμοί καταλαμβάνουν 2 bytes (δηλαδή θέσεις μνήμης). Οι αριθμοί απλής ακρίβειας καταλαμβάνουν 4 bytes και αποθηκεύονται με 7 ψηφία ακρίβεια ενώ της διπλής ακρίβειας καταλαμβάνουν 8 bytes και αποθηκεύονται με 17 ψηφία ακρίβεια ενώ τυπώνονται τα 16.Τι όμως θεωρεί η BASIC ως ακέραιους;

ΑΚΕΡΑΙΟΙ: όλοι οι ακέραιοι αριθμοί από το -32768 ως το 32767. Άρα το 65000 είναι αριθμός απλής ακρίβειας για τη BASIC και όχι ακέραιος.

ΑΠΛΗΣ ΑΚΡΙΒΕΙΑΣ: οι αριθμοί που πληρούν μια από τις εξής προϋποθέσεις:

a) έχουν 7 ή λιγότερα ψηφία και δεν είναι ακέραιοι

b) είναι δυνάμεις με το Ε πχ ο 17Ε-09.

c) οι μεταβλητές έχουν (!) δηλαδή η Α! Περιέχει απλής ακρίβειας αριθμό

ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑΣ: οι αριθμοί που πληρούν μια από τις εξής προϋποθέσεις:

a) έχουν 8 ως 16 ψηφία

b) εκθετική μορφή με το D (double precision)

c) οι μεταβλητές έχουν (#) δηλαδή η A# περιέχει διπλής ακρίβειας αριθμό.

3.3.2 Ορισμένα παραδείγματα

Αν υποθέσομε ότι καλούμε 1 την κατηγορία των ακεραίων, 2 την κατηγορία των αριθμών απλής ακρίβειας και 3 αυτή των διπλής ακρίβειας τότε:

2.3 (2) 0.9999 (2) 987 (1) -32000 (1) 16E30 (2) 16E-30 (2)

A!=99 (2) Z#=18927 (3) -.928327362891 (3) 165.223678213 (3)

3.3.3 Άσκηση

Να γραφεί πρόγραμμα που να ανοίγει αρχείο τυχαίας προσπέλασης και να γράφει 5 ομάδες στοιχείων που αποτελούνται από 1 αριθμητική και 2 αλφαριθμητικές μεταβλητές και των οποίων οι τιμές παίρνονται από το πληκτρολόγιο. (να υποτεθεί ότι οι αριθμητικές τιμές είναι ακέραιοι)

Λύση:

Αρχικά ανοίγω το αρχείο . Κάθε όμως φορά που ανοίγομε ένα αρχείο τυχαίας προσπέλασης, μετράμε πόσα bytes δηλαδή πόσες θέσεις μνήμης καταλαμβάνονται από κάθε ομάδα εγγραφών. Εδώ έχω 2+10+10=22 θέσεις. Τα 2 bytes είναι των ακεραίων ενώ για τα 10+10 υπέθεσα ότι πρόκειται για αλφαριθμητικές μεταβλητές μήκους δέκα χαρακτήρων. Δηλαδή αφού το πρόβλημα

δεν έδινε κανένα στοιχείο όσον αφορά το μέγεθος των αλφαριθμητικών μεταβλητών , το όρισα εγώ χωρίς εν γένει σφάλμα της γενικότητας. Με άλλα λόγια δε θα ήταν λάθος (κάτω από τις δεδομένες συνθήκες της εκφώνησης)να έγραφα 2+12+12=26 για παράδειγμα θέσεις.

```
OPEN" OUTFILE.DAT" AS #1 LEN=22
```

Τώρα ναι μεν ο υπολογιστής γνωρίζει ότι κάθε εγγραφή θα έχει μήκος 22 θέσεις μνήμης , πλην όμως δε γνωρίζει πως θα κατανείμει τις θέσεις αυτές. Δηλαδή θα μπορούσαν να είναι 3 ακέραιες μεταβλητές 1 διπλής ακρίβειας και 2 απλής ακρίβειας γιατί 3*2+1*8+2*4=22 θέσεις. Η κατανομή καθώς και το είδος των μεταβλητών εισάγεται μη την εντολή FIELD ως εξής:

```
FIELD #1, 2 AS NO$,10 AS E1$,10 AS E1$
```

Είναι σαφές ότι E1$,E2$ εννοώ τις αλφαριθμητικές μεταβλητές. Πώς όμως ο υπολογιστής θα καταλάβει ότι στη θέση NO$ θα βάλω εγώ αριθμούς αφού όπως ξέρουμε στη NO$ κανονικά θέτομε αλφαριθμητικές τιμές; Αυτό θα το διαπιστώσομε στη συνέχεια. Πάντως γενικός κανόνας είναι ότι στην εντολή FIELD βάζομε πάντα αλφαριθμητικές μεταβλητές και κατόπιν δείχνομε στον Η/Υ αν είναι αριθμοί και τι είδους είναι.

```
FOR I=1 TO 5 (πέντε ομάδες στοιχείων)
INPUT"Εισάγετε ακέραια τιμή:",NUM
INPUT"Εισάγετε αλφαριθμητική τιμή 1:";ALF1$
INPUT"Εισάγετε αλφαριθμητική τιμή 2:";ALF2$
```

τώρα πριν κλείσομε το βρόχο πρέπει να γράφομε ταυτόχρονα τις ομάδες δεδομένων στο αρχείο που ήδη έχομε ανοίξει. Δηλαδή:

```
LSET E1$=EL1$
```

Λέμε ότι το E1$ τελικά είναι ίσο με την αλφαριθμητική μεταβλητή EL1$.

```
LSET E2$=EL2$
```

Οι εντολές LSET δεν κάνουν τίποτα άλλο από το να προετοιμάζουν τα δεδομένα για εγγραφή σε αρχείο τυχαίας προσπέλασης. Αυτή η ενέργεια είναι απαραίτητη για αρχεία τέτοιας μορφής αλλά δε σημαίνει ότι αποθηκεύτηκαν ακόμα τα δεδομένα.

Απομένει ο αριθμός, που θέλει ιδιαίτερη προσοχή:

```
LSET NO$=MKI$(NUM)
```

Επειδή είναι ακέραιος είπα στον υπολογιστή: Ετοίμασε (LSET) τη θέση που υπάρχει στο FIELD ως NO$ για καταχώρηση ακεραίου αριθμού (MKI$) που έχει καταχωρηθεί ήδη στην μεταβλητή NUM.

Χρησιμοποιώ δηλαδή τις εντολές:

```
MaKe Integer  (MKI$) για ακέραιους αριθμούς
MaKe Single (MKS$) για αριθμούς απλής ακρίβειας
MaKe Double (MKD$) για αριθμούς διπλής ακρίβειας
```

Τέλος για εγγραφή των ετοιμασμένων πλέον στοιχείων στο αρχείο χρησιμοποιώ την εντολή PUT μαζί με το κανάλι ροής (εδώ το 1) καθώς και το δείκτη εγγραφής (δηλαδή το δείκτη που μας δίνει τον αριθμό της ομάδας).

```
PUT 1,I
```

(Παρατηρούμε ότι το κανάλι ροής για πρώτη και μοναδική φορά δεν θέλει το σύμβολο # απαραίτητο όμως για όλα τα άλλα.)

```
NEXT I
CLOSE #1
```

Οπότε συνολικά το πρόγραμμά μας έχει ως εξής:

```
10 OPEN" OUTFILE.DAT" AS #1 LEN=22
20 FIELD #1, 2 AS NO$,10 AS E1$,10 AS E1$
30 FOR I=1 TO 5
40 INPUT"Εισάγετε ακέραια τιμή:",NUM
```

```
 50 INPUT"Εισάγετε αλφαριθμητική τιμή 1:";ALF1$
 60 INPUT"Εισάγετε αλφαριθμητική τιμή 2:";ALF2$
 70 LSET E1$=EL1$
 80 LSET E2$=EL2$
 90 LSET NO$=MKI$(NUM)
100 PUT I,1
110 NEXT I
120 CLOSE #1
130 END
```

3.3.4 Επέκταση της 3.3.3

Να συνταχθεί πρόγραμμα που να διαβάζει τα δεδομένα που υπάρχουν σε αρχείο τυχαίας προσπέλασης με τη μορφή που γράφτηκαν από το πρόγραμμα 3.3.3. Να θεωρηθούν οι ίδιοι περιορισμοί που ετέθησαν κατά την εκφώνηση του 3.3.3.

Λύση:

Ας υποθέσομε ότι έχομε να συντάξομε μια ρουτίνα ανάγνωσης των δεδομένων του αρχείου τυχαίας προσπέλασης που δημιουργήθηκε από το 3.3.3. Επομένως το αρχείο υπάρχει, είναι με άλλα λόγια έτοιμο για ανάγνωση.

Ενώ στα αρχεία σειριακής προσπέλασης ήμασταν αναγκασμένοι να δηλώσομε στον Η/Υ αν θα ανοίξομε το αρχείο για εγγραφή ή ανάγνωση, εδώ δεν υπάρχει κάτι τέτοιο. Δηλαδή ανοίγουμε το αρχείο και μπορούμε να το διαβάσομε, να εγγράψομε δεδομένα σε αυτό ή να προσθέσομε χωρίς να έχει καμία σημασία ποια σειρά θα ακολουθήσομε ή ποιες εργασίες θα εκτελέσομε. Μετά από αυτά, είναι σαφές ότι πρέπει να ανοίξομε το αρχείο:

```
OPEN"OUTFILE.DAT" AS #1 LEN=22
```

Κατόπιν τούτου, θα δώσουμε στο μεταφραστή της BASIC πληροφορίες για το πως κατανέμονται οι 22 θέσεις μνήμης, κάτι που κάναμε και κατά την εγγραφή. Συνεπώς χρησιμοποιούμε πάλι την εντολή FIELD ως εξής:

```
FIELD #1,2 AS NO$,10 AS E1$,10 AS E2$
```

Παρατηρούμε ότι κατά την εγγραφή, ανάγνωση ή πρόσθεση στοιχείων σε αρχεία τυχαίας προσπέλασης οι δυο πρώτες εντολές OPEN και FIELD είναι εντελώς ίδιες.

Αυτό είναι γενική παρατήρηση και ισχύει πάντα. Αξίζει όμως να σημειώσομε ότι δεν έχει σημασία αν θα χρησιμοποιήσομε τα ίδια ονόματα για τις μεταβλητές στο FIELD,δηλαδή τα NO$,E1$ και E2$. Διαβάζω τα στοιχεία με τη χρήση της εντολής GET. Αυτή θα δουλέψει ως εξής:

Οι αλφαριθμητικές μεταβλητές E1$ και E2$ θα πάρουν τις τιμές που έχουν αποθηκευτεί καθόσον ο Η/Υ μπορεί να αναγνωρίσει αν ένα δεδομένο είναι αριθμός ή μείγμα χαρακτήρων. Αυτό που δεν μπορεί να κάνει είναι να αναγνωρίσει τον τύπο του αριθμού , δηλαδή ακέραιος, απλής ή διπλής ακρίβειας. Αυτό πρέπει να του το δώσει ο χρήστης μέσω τον παρακάτω εντολών:

```
CVI  (ConVert Integer)  για ακέραιους αριθμούς
CVS  (ConVert Single) για απλής ακρίβειας αριθμούς
CVD  (ConVert Double) για διπλής ακρίβειας αριθμούς
```

Άρα το πρόγραμμα πρέπει να έχει τις εξής εντολές:

```
FOR I=1 TO 5
GET #1,I
PRINT E1$
PRINT E2$
NUM=CVI(NO$)
PRINT NUM
NEXT I
```

Σημειώνομε ότι γίνεται (έτσι όπως το φτιάξαμε) αυτόματη εκτύπωση των δεδομένων στην οθόνη μέσω των εντολών PRINT αμέσως μετά την ανάγνωση των δεδομένων. Συγκεντρωτικά το πρόγραμμά μας έχει ως εξής:

```
10 OPEN"OUTFILE.DAT" AS #1 LEN=22
20 FIELD #1,2 AS NO$,10 AS E1$,10 AS E2$
30 FOR I=1 TO 5
40 GET #1,I
 50 PRINT E1$
 60 PRINT E2$
 70 NUM=CVI(NO$)
 80 PRINT NUM
 90 NEXT I
100 CLOSE #1
110 END
```

3.3.5 Θέμα 1ο/ Ιούνιος 1992

Ανοίξτε αρχείο τυχαίας προσπέλασης, στο οποίο θα είναι δυνατή η εγγραφή των παρακάτω στοιχείων δοκών: Το όνομα της δοκού που μπορεί να περιέχει μέχρι πέντε χαρακτηριστικά ψηφία, το μήκος του ανοίγματος, το συγκεντρωμένο φορτίο, το ομοιόμορφο φορτίο (πχ 1.78), τη ροπή κάμψεως (πχ 12.42) και την τέμνουσα (πχ 11.12).

Λύση:

Για να ανοίξουμε το αρχείο τυχαίας προσπέλασης , πρέπει να γνωρίζομε πόσες θέσεις μνήμης καταλαμβάνονται από κάθε εγγραφή, δηλαδή από το σύνολο των στοιχείων της δοκού που εγγράφομε κάθε φορά.

Για το όνομα έχω πέντε χαρακτηριστικά στοιχεία, άρα κρατώ 5 θέσεις. Το μήκος της δοκού μπορεί να είναι ακέραιος (το πλέον πιθανό) αλλά και δεκαδικός πχ 14.5 m, για το λόγο αυτό κρατώ 4 θέσεις μνήμης καθώς είναι γνωστό ότι οι δεκαδικοί απαιτούν 4 θέσεις μνήμης για την αποθήκευσή των. Το συγκεντρωμένο φορτίο υποθέτω ότι είναι μάλλον ένας ακέραιος και για το λόγο αυτό κρατώ 2 θέσεις. Για τα άλλα 3 στοιχεία έχω ότι ενδέχεται να είναι δεκαδικοί άρα κρατώ 4 θέσεις μνήμης για το καθένα. Αθροίζω και έχω LEN=5+4+2+4+4+4=23

```
OPEN"ARXEIO.DAT" AS #1 LEN=23
```

Στη συνέχεια δηλώνω πως θα κατανεμηθούν οι 23 αυτές θέσεις μεταξύ των διαφόρων δεδομένων που ετοιμάζομαι να εγγράψω.

```
FIELD #1,5 AS NAM$,4 AS MH$,2 AS SF$,4 AS OF$,4 AS RK$,4
AS TF$
```

Όπου NAM$ το όνομα της δοκού

MH$ το μήκος

SF$ το συγκεντρωμένο φορτίο

RK$ η ροπή κάμψεως

OF$ το ομοιόμορφο φορτίο και

TF$ η τέμνουσα δύναμις

Με τη χρήση DATA ή απευθείας από το πληκτρολόγιο ορίζω τις μεταβλητές:

```
INPUT"Εισάγετε το όνομα της δοκού=";NAME$
INPUT"Ποιο το μήκος του ανοίγματος=";MH
INPUT"Ποιο το συγκεντρωμένο φορτίο=";SF
INPUT"Ποιο το ομοιόμορφο φορτίο=";OF
INPUT"Ποια η ροπή κάμψεως=";RK
INPUT"Ποια η τέμνουσα δύναμη=";TF
```

Στη συνέχεια τα "ετοιμάζω" για εγγραφή έτσι όπως είδαμε στο θέμα 3.3.1.

```
LSET NAM$=NAME$
LSET  MH$=MKS(MH)
LSET  SF$=MKI$(SF)
LSET OF$=MKS$(OF)
LSET RK$=MKS$(RK)
LSET TF$=MKS$(TF)
```

Τέλος για την εγγραφή των δεδομένων πρέπει να χρησιμοποιήσω την εντολή PUT #1,1 ακολουθούμενη από την CLOSE για να δηλώσω ότι δεν χρειάζομαι άλλο το αρχείο. Προσέχουμε τη διαφορά μεταξύ MKI$,MKS$ καθότι σε διαφορετική περίπτωση η εκτέλεση του προγράμματος διακόπτεται βγάζοντας μήνυμα λάθους. Υπενθυμίζεται ότι για την ανάγνωση των δεδομένων (από άλλη ρουτίνα) θα πρέπει να ακολουθηθεί <u>απαραίτητα η ίδια σειρά</u> ανάγνωσης των δεδομένων που είχε ακολουθηθεί κατά την εγγραφή. Συνολικά το πρόγραμμα έχει ως εξής:

```
 10 OPEN"ARXEIO.DAT" AS #1 LEN=23
 20 FIELD #1,5 AS NAM$,4 AS MH$,2 AS SF$,4 AS OF$,4 AS
RK$,4 AS TF$
 30 INPUT"Εισάγετε το όνομα της δοκού=";NAME$
 40 INPUT"Ποιο το μήκος του ανοίγματος=";MH
 50 INPUT"Ποιο το συγκεντρωμένο φορτίο=";SF
 60 INPUT"Ποιο το ομοιόμορφο φορτίο=";OF
 70 INPUT"Ποια η ροπή κάμψεως=";RK
 80 INPUT"Ποια η τέμνουσα δύναμη=";TF
 90 LSET NAM$=NAME$
100 LSET  MH$=MKS(MH)
110 LSET  SF$=MKI$(SF)
120 LSET OF$=MKS$(OF)
130 LSET RK$=MKS$(RK)
140 LSET TF$=MKS$(TF)
150 PUT #1,1
160 CLOSE #1
```

```
170 END
```

3.4 ΣΥΝΤΑΣΣΟΝΤΑΣ ΕΥΠΑΡΟΥΣΙΑΣΤΑ ΠΡΟΓΡΑΜΜΑΤΑ

3.4.1 Ο λόγος

Μέχρι στιγμής τα προγράμματα που μελετήσαμε ήταν αρκετά απλά και δεν συνέτρεχε λόγος για κάποια ιδιαίτερη παρουσίαση. Υπάρχουν όμως πολλά προγράμματα - μεγάλα σε μέγεθος - τα οποία θα ήταν αδύνατο να τα χρησιμοποιήσει κανείς αν δεν παρουσίαζαν τα αποτελέσματα με κομψό τρόπο. Για παράδειγμα ένα στατικό πρόγραμμα που επιλύει ένα φορέα πρέπει να παρουσιάζει τα αποτελέσματα με κάποια σειρά και μια ορισμένη μορφή έτσι ώστε να είναι ευανάγνωστα και εύκολα επεξεργάσιμα.

3.4.2 Οι απαιτήσεις

Στο μάθημα της BASIC οι απαιτήσεις είναι περιορισμένες όσον αφορά την παρουσίαση των προγραμμάτων αλλά όχι αμελητέες. Ζητείται συνήθως η ευθυγράμμιση αποτελεσμάτων (δεξιά ή αριστερά).

```
     0.928
12.268        ακανόνιστη εκτύπωση

  0.928
12.2628       δεξιά ευθυγράμμιση

  0.928
  12.268      αριστερή ευθυγράμμιση
```

3.4.3 Οι μέθοδοι

Για την ευθυγράμμιση θα χρησιμοποιηθεί η PRINT USING της οποίας η αναλυτική σύνταξη καθώς και παραδείγματα μπορούν να βρεθούν στη θεωρία.

Για να καθαριστεί η οθόνη και να αρχίσει το πρόγραμμα να γράφει από την πρώτη σειρά, θα χρησιμοποιηθεί η εντολή CLS ως έχει.

Αυτές οι δυο εντολές σε γενικές γραμμές θα μας βοηθήσουν στο να παρουσιάσομε με κομψό τρόπο τα προγράμματά μας.

3.4.4 Θέμα 3ο/ Ιούνιος 1991

Να συνταχθεί πρόγραμμα το οποίο να υπολογίζει το εμβαδόν της επιφάνειας και τον όγκο σφαίρας με ακτίνα R και να εκτυπώνει τα αποτελέσματα, όταν η ακτίνα της σφαίρας μεταβάλλεται από R=6 έως R=20 στην παρακάτω μορφή:

```
 R       E         V
 6    452.16    904.32
 ………………………………………………………
20   5024.00  25120.00
```

Λύση:

Αρχικά καθαρίζομε την οθόνη με την εντολή CLS. Έπειτα σκεφτόμαστε να βάλουμε την επικεφαλίδα R,E,V των αποτελεσμάτων και κατόπιν να ορίσομε ένα βρόχο που να υπολογίζει εμβαδόν επιφάνειας και όγκο σφαίρας καθώς η ακτίνα μεταβάλλεται.

```
CLS
PRINT"       R       E         V"
```

Για το R αφήσαμε 7 κενά διαστήματα , για το E αφήσαμε 7 ενώ για το V 9 καθότι ο όγκος φτάνει μέχρι 7 ψηφία. Έχοντας τυπώσει τα αρχικά ετοιμάζομε το βρόχο:

```
FOR R=6 TO 20
PRINT TAB(6);R;
ή εντελώς ισοδύναμα PRINT"       ";R;
```

Το (;) μετά το R σημαίνει ότι θα εκτυπωθούν και άλλα δεδομένα στην ίδια γραμμή και αποτρέπουν τον Η/Υ να συνεχίσει τη ροή της εκτύπωσης από την επόμενη γραμμή.

```
E=4*PI*(R^2)
V=(4/3)*PI*(R^3)
```

Δεν λησμονούμε ότι η BASIC δεν γνωρίζει τον αριθμό PI τον οποίο πρέπει να τον εισάγομε εμείς και προφανώς πριν από τη χρησιμοποίησή του.

```
PI=3.14159
```

Παρατηρούμε στη συνέχεια ότι ενώ λογικά θα έπρεπε τα E,V να υπολογίζονται με ακρίβεια περισσοτέρων δεκαδικών ψηφίων, εν τούτοις στην εκφώνηση παρουσιάζονται μόνο δυο. Αυτό σημαίνει ότι πρέπει απαραίτητα να γίνει χρήση της εντολής PRINT USING. Πράγματι γράφομε στη συνέχεια:

```
PRINT USING"####.##";E;
PRINT USING"#####.##";V
```

Παρατηρείστε ότι την πρώτη φορά θέλω τέσσερα ψηφία πριν την υποδιαστολή ενώ τη δεύτερη πέντε. Αυτό το διαπιστώνομε εύκολα από την εκφώνηση που δίνει τις μέγιστες τιμές που μπορούν να πάρουν οι μεταβλητές. Αν δεν τις έδινε τότε αναγκαστικά έπρεπε να ορίσομε περισσότερα ψηφία για ασφάλεια.

Μετά το δεύτερο PRINT USING δεν χρησιμοποιώ (;) . Αυτό σημαίνει ότι δεν επιθυμώ να τυπώσω περισσότερα στοιχεία στην ίδια γραμμή και επομένως μπορεί ο Η/Υ να συνεχίσει τη ροή της εκτύπωσης από την επόμενη γραμμή. Συνοψίζοντας:

```
   10 CLS
   20 PRINT"       R         E           V"
   30 PI=3.14159
   40 FOR R=6 TO 20
   50 PRINT TAB(6);R;
   60 E=4*PI*(R^2)
   70 V=(4/3)*PI*(R^3)
 80 PRINT USING"####.##";E;
 90 PRINT USING"#####.##";V
100 NEXT R
110 END
```

ΚΕΦΑΛΑΙΟ 4: ΛΥΜΕΝΑ ΠΡΟΓΡΑΜΜΑΤΑ

Α. ΕΝΟΤΗΤΑ

4.1 ΘΕΜΑ 4ο/ ΙΟΥΝΙΟΣ 1992

Δίδεται γεωμετρικό σχήμα αποτελούμενο από τρία τρίγωνα. Ζητείται να υπολογιστεί το εμβαδόν του. Τα μήκη των πλευρών του τριγώνου να δοθούν από το πληκτρολόγιο ή από σειριακό αρχείο ή από DATA γραμμένα σύμφωνα με την επιλογή του συντάκτη του προγράμματος. Το εμβαδόν του τριγώνου όταν είναι γνωστές οι πλευρές του δίδεται από τη σχέση:

$$E= \sqrt{t(t-a)(t-b)(t-c)}$$

όπου t η ημιπερίμετρος του τριγώνου.

Λύση:

Α' ΤΡΟΠΟΣ : Με χρήση εντολών DATA

```
  10 CLS
  20 DIM E(3)
  30 RESTORE 120
  40 FOR X=1 TO 3
  50 READ A,B,C
  60 T=(A+B+C)/2
  70 E(X)=SQR(T*(T-A)*(T-B)*(T-C))
  80 NEXT X
  90 E=E(1)+E(2)+E(3)
100 PRINT"Το ζητούμενο εμβαδό είναι E=";E
110 END
120 DATA 7,4,5,5,3,6,6,2,4
```

Η εντολή SQR( ) μας δίνει την τετραγωνική ρίζα Square Root της περιεχόμενης παράστασης (βλ. Θεωρία). Παρατηρείστε ότι λόγω της δομής του προγράμματος, κάθε κοινή πλευρά των τριγώνων εμφανίζεται δυο φορές.

Β' ΤΡΟΠΟΣ : Με εισαγωγή από το πληκτρολόγιο

```
   10 CLS
   20 DIM A(3),E(3)
   30 FOR X=1 TO 3
   40 FOR I=1 TO 3
   50 PRINT"Εισάγετε το μήκος της πλευράς ";I;" για το
τρίγωνο ";X
   60 INPUT A(I)
   70 NEXT I
   80 T=(A(1)+A(2)+A(3))/2
   90 E(X)=SQR(T*(T-A(1))*(T-A(2))*(T-A(3)))
  100 NEXT X
  110 E=E(1)+E(2)+E(3)
  120 PRINT"Το ζητούμενο εμβαδόν είναι E=";E
  130 END
```

Αξίζει να προσεχθεί η χρήση της εντολής PRINT της σειράς 50 που δίνει ταυτόχρονα πληροφορίες για τον αριθμό του τριγώνου καθώς και για τον αριθμό της πλευράς του. Χρησιμοποιήθηκαν δυο βρόχοι ο ένας εντός του άλλου για λόγους ταχύτητας. Θα μπορούσαμε δηλαδή να αντικαταστήσομε τον ένα βρόχο (I) με τρία INPUT ένα για κάθε πλευρά.

Γ' ΤΡΟΠΟΣ: Με ανάγνωση από σειριακό αρχείο

```
   10  CLS:DIM E(3)
   20 OPEN"OUTFILE.DAT" FOR INPUT AS #1
   30 FOR X=1 TO 3
   40 INPUT #1,A,B,C
   50 T=(A+B+C)/2
```

```
  60 E(X)=SQR(T*(T-A)*(T-B)*(T-C))
  70 NEXT X
  80 E=E(1)+E(2)+E(3)
  90 PRINT" Το ζητούμενο εμβαδό είναι E=";E
100 CLOSE #1
110 END
```

4.2 ΘΕΜΑ 1ο/ ΙΟΥΝΙΟΣ 1995

Να συνταχθεί πρόγραμμα που να γράφει σε αρχείο τυχαίας προσπέλασης τον κωδικό, το ονοματεπώνυμο, το όνομα πατρός και τον βαθμό κάθε φοιτητή. Να έχει επίσης δυνατότητα αναζήτησης ενός φοιτητή καθώς και δυνατότητα εμφάνισης όλων των δεδομένων που έχουν εγγραφεί.

Λύση:

Ας υποθέσομε ότι έχομε να κάνομε με κωδικούς της μορφής 01094018. Τότε αυτοί θα έχουν επτά ψηφία (το 0 στην αρχή δεν προσμετράται) και άρα είναι αριθμοί απλής ακρίβειας και καταλαμβάνουν 4 θέσεις μνήμης. Το όνομα κάθε φοιτητή υποθέτω ότι είναι μέχρι 5 γράμματα και το επώνυμο μέχρι 29.Επίσης το όνομα πατρός μέχρι 5 γράμματα και ο βαθμός (0-10) προφανώς ακέραιος άρα κρατώ 2 θέσεις μνήμης. Συνολικά έχω LEN=4+5+29+5+2=45.

```
 10 CLS
 20 PRINT"1. Εισαγωγή στοιχείου"
 30 PRINT"2. Αναζήτηση στοιχείου"
 40 PRINT"3. Λίστα στοιχείων"
 50 PRINT"4. Τέλος εργασιών"
 60 INPUT"Επιλέξτε";CH
 70 IF CH=1 THEN GOTO 110
 80 IF CH=2 THEN GOTO 300
```

```
  90 IF CH=3 THEN GOTO 470
100 END
110 OPEN"OUTPUT.DAT" AS #1 LEN=47
120 FIELD #1,4 AS CD$,29 AS SN$,5 AS NM$,5 AS FA$,2 AS
BA$
130 DX=LOF(1)/45
140 INPUT"Κωδικός φοιτητή:";C
150 INPUT"Επώνυμον       :";S$
160 INPUT"Ονομα          :",N$
170 INPUT"Ονομα πατρός   :",F$
180 INPUT"Βαθμός φοιτητή :";B
190 LSET CD$=MKS$( C )
200 LSET SN$=S$
210 LSET NM$=N$
220 LSET FA$=F$
230 LSET BA$=MKI$(B)
240 DX=DX+1
250 PUT 1,DX
260 CLOSE #1
270 GOTO 30


300 OPEN"OUTPUT.DAT" AS #1 LEN=45
310 FIELD #1,4 AS CD$,29 AS SN$,5 AS NM$,5 AS FA$,2 AS
BA$
320 DX=LOF(1)/45
330 INPUT"Επώνυμο φοιτητή προς αναζήτηση:",SNF$
340 SNF$=SNF$+SPACE$(29-LEN(SNF$))
350 FOR A=1 TO DX
360 GET #1,A
370 IF SNF$=SN$ THEN GOTO 430
380 NEXT A
390 PRINT"Το όνομα αυτό δε βρέθηκε!"
400 GOTO 40
430 PRINT CVS(CD$);"  ";SN$;"  ";NM$;"  του ";FA$
440 PRINT"Ο βαθμός του φοιτητή είναι ";CVS(BA$)
450 GOTO 40
```

```
470 OPEN”OUTPUT.DAT” AS #1,LEN=45
480 FIELD #1,4 AS CD$,29 AS SN$,5 AS NM$,5 AS FA$,2 AS
BA$
490 DX=LOF(1)/45
500 FOR A=1 TO DX
510 GET #1,A
520 PRINT CVD(CD$);”  “;SN$;”  “;NM$;”  “;FA$;”
“;CVS(BA$)
530 NEXT A
540 GOTO 40
```

Για να γίνει το πρόβλημα αυτό πιο κατανοητό , επικεντρώνομε αρχικά το ενδιαφέρον μας στη νέα τεχνική που ακολουθήθηκε κατά τη διαχείριση των αρχείων. Πιο συγκεκριμένα γνωρίζομε ότι σε ένα αρχείο είναι εγγεγραμμένα κάποιες ομάδες δεδομένων των οποίων γνωρίζουμε τη σειρά και το είδος πλην όμως δε γνωρίζομε το πλήθος.

Έτσι όταν μας ζητείται να γίνει η λίστα αυτών των δεδομένων εμείς προφανώς πρέπει να ξέρουμε πόσες ομάδες έχουμε , για να θέσομε τον αριθμό αυτό στο βρόχο επανάληψης. Αυτό άμεσα δεν το ξέρομε, είναι όμως δυνατό να το προσδιορίσομε έμμεσα. Η εντολή LOF (1) μας δίνει ως γνωστό το μήκος σε bytes δηλαδή σε θέσεις μνήμης, του αρχείο που έχει ανοιχτεί στο κανάλι ροής 1.

Ας υποθέσουμε ότι είναι 900 το αποτέλεσμα. Αν τώρα το διαιρέσομε δια του αριθμού που μας δίνει το μήκος κάθε ομάδας , είναι σαφές ότι παίρνομε το πλήθος των ομάδων, δηλαδή 900/45=20 ομάδες και άρα όταν μας ζητείται να προσθέσομε στη λίστα έναν νέο φοιτητή αυτό που γίνεται στις γραμμές 130 και 240 δεν είναι τίποτα άλλο από την ανεύρεση του πλήθους των ομάδων (DX) και την πρόσθεση της μονάδας (+1) καθότι ετοιμαζόμαστε να προσθέσουμε μια ομάδα η οποία θα είναι η 21η για παράδειγμα.

Μετά από αυτά το πρόβλημα έχει γίνει φυσικά πολύ πιο κατανοητό. Σημειώνομε ότι τη μέθοδο του να βρίσκομε έμμεσα το πλήθος των εγγραφών σε ένα αρχείο τυχαίας (και σειριακής με τον ίδιο τρόπο) προσπέλασης θα χρησιμοποιείται χωρίς ιδιαίτερο σχόλιο.

Τέλος αξίζει να προσέξει κανείς πως με την εντολή PRINT τυπώνομε όλα τα απαραίτητα δεδομένα σε μια γραμμή με τις εντολές 430 και 520.

4.3 Άσκηση

Χρησιμοποιώντας υπορουτίνες να συντάξετε πρόγραμμα που να υπολογίζει τον όγκο σφαίρας ή τον όγκο κύβου ανάλογα με την προτίμηση του χρήστη.

Λύση:

Πριν προχωρήσομε στην επίλυση του θέματος, θα παραθέσομε λίγα λόγια περί υπορουτίνων. Αυτές καλούνται με την εντολή GOSUB και η ροή επιστρέφεται στο κυρίως πρόγραμμα όταν συναντηθεί η εντολή RETURN. Είναι εν γένει καλό και για εμάς αλλά και για τον διορθωτή να προσθέτομε πριν από κάθε υπορουτίνα ένα μικρό σχόλιο που να περιγράφει τη λειτουργία της μέσω της εντολής REM.

```
10 CLS
20 PRINT"1. Υπολογισμός όγκου σφαίρας"
30 PRINT"2. Υπολογισμός όγκου κύβου"
40 PRINT"3. Εξοδος από το πρόγραμμα"
50 INPUT"Επιλέξατε 1-3 :";CH
60 IF CH=3 THEN END
70 IF CH=1 THEN GOSUB 100
80 IF CH=2 THEN GOSUB 160
```

```
90 GOTO 50

100 REM * Υπολογισμός όγκου σφαίρας *
110 INPUT"Ποια η ακτίνα της σφαίρας=";R
120 PI=3.14159
130 V=(4/3)*PI*(R^3)
140 PRINT"Ο όγκος της σφαίρας είναι V=";V
150 RETURN

160 REM * Υπολογισμός όγκου κύβου *
170 INPUT"Ποια η ακμή του κύβου =";A
180 V=A^3
190 PRINT"Ο όγκος του κύβου είναι V=";V
200 RETURN
```

4.4 Θέμα 1ο/ Σεπτέμβριος 1995

Να συνταχθεί πρόγραμμα που να υπολογίζει τις αμοιβές μηχανικού για έργα στατικά και για έργα οδοποιίας. Ο τύπος που δίνει την αμοιβή του μηχανικού είναι:

$$M = S\sqrt{\frac{10l(4-a)}{16-2a}}5a$$

όπου S είναι η αμοιβή που δίδεται για το εκάστοτε έργο στο μηχανικό, α ένας συντελεστής ίσος με 2 για στατικά έργα και με 3 για έργα οδοποιίας και *l* ένας άλλος συντελεστής ίσος με :

$$l = \alpha^{\frac{1}{3}}$$

Λύση:

```
10 CLS
20 INPUT"Εισάγετε αμοιβή μηχανικού για το έργο:";S
30 INPUT"Εισάγετε τον αριθμό 1 για έργο οδοποιίας :";E
40 IF E=1 THEN A =3 ELSE A=2
50 L=A^(1/3)
60 PAR=16-2*A
70 ARI=10*L*(4-A)*5*A
80 KL=ARI/PAR
90 M=S*SQR(KL)
100 PRINT"Η αμοιβή του μηχανικού είναι ";M
110 END
```

Διευκρίνιση: ο τύπος δεν είναι αυτός που δόθηκε κατά την εξέταση αλλά επαυξήθηκε για να αποτελέσει το θέμα αυτό και ένα αξιόλογο παράδειγμα μαθηματικών πράξεων με την BASIC .

4.5 Θέμα 2ο/Σεπτέμβριος 1995

Δίνεται ένα αρχείο τυχαίας προσπέλασης με 100 ζεύγη στοιχείων (x,y).Ζητείται για κάθε ένα από τα ζεύγη αυτά να υπολογιστεί η απόσταση του σημείου με συντεταγμένες τα (x,y) από το O(0,0) και κατόπιν το αποτέλεσμα να αποθηκευτεί στο αρχείο αυτό.

Λύση:

```
10 CLS
20 OPEN"INPUT" AS #1 LEN=4
30 FIELD #1,2 AS X$,2 AS Y$
40 DIM D(100),X(100),Y(100)
50 FOR I=1 TO 100
60 GET #1,I
70 X=CVI(X$):X(I)=X
```

```
  80 Y=CVI(Y$):Y(I)=Y
  90 D(I)=SQR(X^2+Y^2)
100 NEXT I
110 CLOSE #1
120 OPEN”INPUT AS #1 LEN=8
130 FIELD #1,2 AS X$,2 AS Y$,4 AS D$
140 FOR J=1 TO 100
150 LSET X$=MKI$(X(J))
160 LSET Y$=MKI$(Y(J))
170 LSET D$=MKS$(D(J))
180 PUT  1,J
190 NEXT J
200 CLOSE #1
210 END
```

Με λίγα λόγια, φορτώνω όλους τους αριθμούς x και y σε δυο πίνακες στήλες. Την δε απόσταση την υπολογίζω ταυτόχρονα και την αποθηκεύω σε ένα τρίτο πίνακα. Ξανανοίγω το αρχείο και γράφω τα στοιχείο με τη σειρά x,y,D.Κάθε άλλη λύση έχει προβλήματα (εκτός από μια επίπονη περίπτωση) διότι το FIELD αλλάζει προκειμένου να λυθεί το θέμα κάτι που δεν επιτρέπεται να γίνει όσο το αρχείο είναι ανοικτό.

Β. ΕΝΟΤΗΤΑ

Εντολές αριθμητικού ελέγχου

ABS η απόλυτη τιμή του αριθμού (ABSolute value)
FIX επιστρέφει το ακέραιο μέρος ενός αριθμού
INT επιστρέφει ακέραιο αριθμό στρογγυλεύοντας τον προηγούμενο

4.6 Άσκηση

Δίνονται δύο πίνακες ο Α και ο Β με 1000 και 2000 στοιχεία αντίστοιχα (μονοδιάστατοι).Ζητείται να συνταχθεί πρόγραμμα που να κάνει τα ακόλουθα:

a) να βρίσκει το μεγαλύτερο στοιχείο του Α και τη θέση του

b) το μικρότερο στοιχείο το Β και τη θέση του

c) να ανταλλάσσει τον αριθμό των δεκάδων του μικρότερου στοιχείου με τον αριθμό των χιλιάδων του μεγαλύτερου στοιχείου.

d) να εκτυπώνει τη νέα τιμή των και τους αντίστοιχους δείκτες.

Λύση:

```
 10 CLS
 20 REM * Ψάχνω για το μεγαλύτερο της λίστας Α *
 30 AMAX=A(1):AI=1
 40 FOR I=1 TO 1000
 50 IF AMAX<A(I) THEN AI=I:AMAX=A(I)
 60 NEXT I
 70 REM *  Ψάχνω για το μικρότερο στοιχείο της λίστας Β *
 80 BMIN=B(1):BI=1
 90 FOR J=1 TO 2000
100 IF BMIN>B(J) THEN BI=J:BMIN=B(J)
```

```
110 NEXT J
120 REM Ανταλλαγή ψηφίων πχ τα 123,6125 θα γίνουν 163 & 2125
130 A=AMAX
140 B=BMIN
160 DEK=FIX(B/10) - (FIX(B/100)*10)
170 IF DEK<0 THEN GOSUB 270
180 XIL=FIX(A1000)-(FIX(A/10000)*10)
190 IF XIL<0 THEN GOSUB 280
200 A=A-XIL*1000+DEK*1000
210 B=B-DEK*10+XIL*10
220 IF AAP=1 THEN A=-A
230 IF BAP=1 THEN B=-B
240 PRINT"Ο αριθμός Β είναι τώρα ο ";B;" με δείκτη ";BI
250 PRINT"Ο αριθμός Α είναι τώρα ο ";A;" με δείκτη ";AI
260 END
270 DEK=ABS(DEK):A=ABS(A):AAP=1:RETURN
280 XIL=ABS(XIL):B=ABS(B):BAP=1:RETURN
```

Προσοχή πρέπει να δοθεί στο γεγονός ότι μπορεί ένας ή και οι δύο αριθμοί να είναι αρνητικοί οπότε πρέπει να χρησιμοποιηθεί αφενός η ABS για να γίνουν θετικοί και αφετέρου πρέπει να ορίσομε ένα δείκτη ώστε να επαναφέρομε τις τιμές των δυο αριθμών μετά τους υπολογισμούς. (εδώ AAP και BAP).

4.7 Άσκηση

Γράψτε μια υπορουτίνα που να υπολογίζει το δεκαδικό μέρος πηλίκου δυο ακεραίων αριθμών.

Λύση:

```
10 CLS
20 INPUT"Εισάγετε έναν ακέραιο:";I1
30 IF I1<>INT(I1) THEN PRINT"Λάθος!!":GOTO 20
40 INPUT"Εισάγετε και έναν άλλο ακέραιο:";I2
```

```
  50 IF I2<>INT(I2) THEN PRINT"Λάθος!!":GOTO 40
  60 R=I1/I2
  70 R1=FIX( R )
  80 R=R-R1
  90 IF R<0 THEN R=-R
100 PRINT"Το δεκαδικό μέρος είναι το ";R
110 END
```

4.8 Άσκηση

Γράψτε ένα πρόγραμμα που να υπολογίζει τα αθροίσματα των 20 πρώτων όρων των εξής σειρών:

i) $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{20}$

ii) $1-\frac{1}{2}+\frac{1}{3}-....-\frac{1}{20}$

iii) $1-\frac{1}{2*3}+\frac{1}{3*4}-....-\frac{1}{20*21}$

iv) $1-\frac{1}{2!}+\frac{1}{3!}-....-\frac{1}{20!}$

Λύση:

α)

```
10 CLS
20 P=0:SUM=0
30 FOR I=1 TO 20 STEP+1
40 P=1/I
50 SUM=SUM+P
60 NEXT I
70 END
```

β)
```
10 CLS
20 P=0
30 SUM=0
   40 FOR I=1 TO 20 STEP+1
   50 P=1/I
   60 IF I/2=I\2 THEN SUM=SUM-P ELSE SUM=SUM+P
   70 NEXT I
   80 END
```

γ)
```
10 CLS
   20 P=0
30 SUM=0
40 FOR I=2 TO 20 STEP+1
50 P=1/(I*(I+1))
60 IF I/2=I\2 THEN SUM=SUM-P
70 IF I/2<>I\2 THEN SUM=SUM+P
80 NEXT I
   90 SUM=SUM+1
  100 END
```

δ)
```
10 CLS
20 P=1
30 SUM=0
40 FOR I=1 TO 20 STEP+1
50 P=P*I
60 P=1/P
70 IF I/2=I\2 THEN SUM=SUM-P ELSE SUM=SUM+P
80 NEXT I
90 END
```

Παρατήρηση: σε όλα τα προβλήματα καλό θα ήταν να προστεθεί η εξής γραμμή αμέσως πριν την εντολή END:

```
PRINT"Το άθροισμα της σειράς είναι =";SUM
```

4.9 Άσκηση

Γράψτε ένα πρόγραμμα που να αντιγράφει αρχεία τυχαίας προσπέλασης χρησιμοποιώντας όποιο όνομα θέλει ο χρήστης.

Λύση:

```
10 CLS
20 OPEN"INPUTFL.DAT" AS #1 LEN=1
30 FIELD #1,1 AS A$
40 INPUT"Εισάγετε ένα όνομα για το αρχείο μέχρι
8 γράμματα χωρίς επέκταση:",OFT$
50 IF LEN(OFT$)>8 THEN GOTO 40
60 OFT$=OFT$+".DAT"
70 OPEN OFT$ AS #2 LEN=1
80 FIELD #2,1 AS B$
90 S=LOF(1)
100 FOR X=1 TO S
110 GET #1,X
120 LSET B$=A$
130 PUT 2,X
140 NEXT X
150 CLOSE #1
160 CLOSE #2
170 PRINT"Η αντιγραφή ολοκληρώθηκε."
180 END
```

<u>Παρατήρηση:</u> Η συνάρτηση LEN(OFT$) που χρησιμοποιήθηκε στην γραμμή 50 του παραπάνω προγράμματος, επιστρέφει το μήκος (LENGTH) του αλφαριθμητικού OFT$.Δεν θα πρέπει να συγχέεται π.χ. με το LEN = 1 της γραμμής 20 όπου ανοίγουμε το αρχείο για επεξεργασία.

4.10 Άσκηση

Γράψτε πρόγραμμα που να ενώνει δυο αρχεία τυχαίας προσπέλασης σε ένα νέο αρχείο.

Λύση:

```
 10 CLS
 20 OPEN"INPUT1.DAT" AS #1 LEN=1
 30 OPEN"INPUT2.DAT" AS #2 LEN=1
 40 OPEN"OUTPUT1.DAT" AS #3 LEN=1
 50 FIELD #1,1 AS A$
 60 FIELD #2,1 AS B$
 70 FIELD #3,1 AS C$
 80 S1=LOF(1)
 90 S2=LOF(2)
100 FOR P=1 TO S1
110 GET #1,P
120 LSET C$=A$
130 PUT 3,P
140 NEXT P
150 CLOSE #1
160 FOR DX=1 TO S2
170 GET #2,DX
180 LSET C$=B$
190 PUT 3,S1+DX
200 NEXT DX
210 CLOSE #2
220 CLOSE #3
230 PRINT"Η ένωση (merging) ολοκληρώθηκε."
240 END
```

4.11 Άσκηση

Να ανοιχτεί αρχείο τυχαίας προσπέλασης και να εκτυπωθούν τα περιεχόμενά του στον εκτυπωτή του συστήματος.

Λύση:

```
10 CLS
20 OPEN”INPUT1.DAT” AS #1 LEN=1
30 FIELD #1,1 AS A$
40 FOR DX=1 TO LOF(1)
50 GET #1,DX
60 LPRINT A$;
70 NEXT DX
80 CLOSE #1
90 END
```

Παρατήρηση: η εντολή LPRINT λειτουργεί όπως ακριβώς και η PRINT αλλά αντί να εμφανίζονται τα αποτελέσματα στην οθόνη του χρήστη , εκτυπώνονται.

4.12 Άσκηση

Να εκτυπωθούν τα περιεχόμενα αρχείου σειριακής προσπέλασης στον εκτυπωτή του συστήματος.

Λύση:

```
10 CLS
```

```
20 OPEN”INPUT1.DAT” FOR INPUT AS #1
30 IF EOF(1) THEN GOTO 60
40 INPUT #1,A$
50 LPRINT A$:GOTO 30
60 CLOSE #1
70 END
```

Θυμίζομε ότι η εντολή EOF ελέγχει αν τελείωσαν τα δεδομένα του αρχείου.