ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

SUPER BASIC

GIGATRONICS

ΗΛΕΚΤΡΟΝΙΚΟΙ ΥΠΟΛΟΓΙΣΤΕΣ ΤΗΣ ΕΛΛΑΔΑΣ

Οἱ ὑπολογιστὲς GIGATRONICS ὑπερέχουν
γιατί...

* σχεδιάζονται ἀπὸ Ἕλληνες γιὰ Ἕλληνες

* εἶναι τελευταίας τεχνολογίας

* ἔχουν χαμηλὸ κόστος καὶ μεγαλώνουν οἰκονομικὰ

* μιλᾶνε τὴ γλώσσα μας

* ἔχουν τὸ πιὸ ὑπεύθυνο SERVICE

ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

SUPER BASIC

ΕΙΣΑΓΩΓΗ

Κάθε ηλεκτρονικός υπολογιστής (Η/Υ) είναι εφοδιασμένος με μία ταυλάχιστον γλώσσα προγραμματισμού υψηλού επιπέδου, που επιτρέπει στους προγραμματιστές να γράφουν προγράμματα, για την επίλυση συγκεκριμμένων προβλημάτων. Τα προγράμματα αυτά γράφονται σύμφωνα με το "συντακτικό" της γλώσσας, που χρησιμοποιεί ο Η/Υ.

Σκοπός των οδηγιών αυτών είναι να περιγραφεί η γλώσσα, που χρησιμοποιούν τα συστήματα της GIGATRONICS, αλλά και άλλα ανάλογα συστήματα, ώστε να επιτυγχάνεται ένας σωστός, έξυπνος και ευέλικτος προγραμματισμός.

Η γλώσσα, της οποίας θα περιγραφεί η δομή, οι εντολές και οι δυνατότητες, είναι η **SUPER BASIC.** Αυτή είναι μία γλώσσα STANDARD BASIC, εξελιγμένη από την GIGATRONICS.

Η SUPER BASIC είναι συμβιβαστή με τις γλώσσες BASIC, που χρησιμοποιούν άλλοι υπολογιστές, όσον αφορά τον απαραίτητο κεντρικό κορμό εντολών. Πέρα όμως από τον βασικό αυτό κορμό, είναι εφοδιασμένη με εντολές και χαρακτηριστικά, που της εξασφαλίζουν πολύ μεγάλη ευελιξία προγραμματισμού, μεγάλες ταχύτητες, ιδίως σε διαχείρηση αρχείων, την δυνατότητα επέμβασης στο λειτουργικό σύστημα, ώστε να επιτυγχάνονται συγκεκριμένοι σκοποί όταν απαιτείται, και τέλος οικονομία χώρου της μνήμης.

ΚΕΦΑΛΑΙΟΝ 1

ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΗΝ SUPER BASIC

1.1 Φόρτωμα της γλώσσας στην μνήμη

Το φόρτωμα της γλώσσας SUPER BASIC στην μνήμη του Η/Υ γίνεται αυτόματα, μόλις ανοίξουμε το σύστημα GIGATRONICS και κάνουμε BOOT, πληκτρολογώντας τον κωδικό της μονάδας δίσκου που βρίσκεται το λειτουργικό σύστημα. Τότε, φορτώνεται το λειτουργικό σύστημα και ταυτόχρονα η γλώσσα SUPER BASIC στις διευθύνσεις μνήμης 0400H έως 2550H.

1.2 Λειτουργικότητα της γλώσσας SUPER BASIC

Οταν η SUPER BASIC φορτωθεί στην μνήμη, στην οθόνη εμφανίζεται η ένδειξη GIGATRONICS. Στο σημείο αυτό, ο προγραμματιστής μπορεί είτε να γράψει ένα πρόγραμμα BASIC, είτε να χρησιμοποιήσει κάποια από τις εντολές της γλώσσας, είτε ακόμη να χρησιμοποιήσει κάποια από τα βοηθητικά προγράμματα (UTILITIES) του συστήματος.

1.2.1 Εντολές της γλώσσας και UTILITIES του συστήματος

Η χρήση κάποιας εντολής ή κάποιας UTILITY του συστήματος δεν απαιτεί την χρήση αριθμού εντολής. Παράλληλα, όταν δίνονται εντολές BASIC, για την κατασκευή ενός μικρού προγράμματος, που χωράνε σε μία γραμμή, πάλι δεν απαιτείται η χρήση αριθμού εντολής και πρόγραμμα ή η εντολή εκτελούνται, μόλις πατηθεί το RETURN του πληκτρολογίου, που είναι ταυτόχρονα οι εντολές LF (=LINE FEED) και CR (=CARRIAGE RETURN).

Παράδειγμα: PRINT "NIKOS" <RETURN>
ή
FORI=1TO 10: PRINT "NIKOS":NEXT I <RETURN>

1.2.2 Κατασκευή προγράμματος BASIC και ενεργοποίηση του

Οταν το σύστημα βρίσκεται στην κατάσταση MONITOR, δηλαδή έχει φορτωθεί το λειτουργικό σύστημα και η γλώσσα SUPER BASIC και στην οθόνη υπάρχει η ένδειξη GIGATRONICS, μπορούμε να φορτώσουμε και να ενεργοποιήσουμε ένα πρόγραμμα BASIC, ακολουθώντας τις παρακάτω διαδικασίες.

GIGATRONICS
α) DSK "*" (*=κωδικός μαγνητικού μέσου) <RETURN>
GIGATRONICS
β) NEW <RETURN>
GIGATRONICS
γ) ΠΡΟΓΡΑΜΜΑ BASIC (δίνουμε τις εντολές με τον τρόπο που περιγράφεται στην παράγραφο 1.3)
δ) SAVE "PROG", "PASSWORD" (άν υπάρχει) <RETURN>, όπου το αρχείο PROG έχει δημιουργηθεί σε κάποιο μαγνητικό μέσο από πρίν.
SAVE PROG
GIGATRONICS
ε) RUN ή RUN NN (NN=αριθμός εντολής προγρ/τος)<RETURN>

1.3 Τύπος της γραμμής - εντολή της SUPER BASIC

Οι γραμμές - εντολές BASIC, για την γλώσσα SUPER BASIC, έχουν τον ακόλουθο τύπο (οι οδηγίες ή εντολές ή παράμετροι εντολής, που περιλαμβάνονται σε [ ] δεν είναι απαραίτητες, αλλά προαιρετικές).

NNNNN BASIC εντολή [:BASIC εντολή .....] <RETURN>

Από τον τύπο αυτό γίνεται αμέσως αντιληπτό ότι μία εντολή-γραμμή BASIC πρέπει να αρχίζει από ένα αριθμό το πολύ πενταψήφιο (NNNNN), που χαρακτηρίζει τη γραμμή (ετικέττα γραμμής) και καθορίζει την σειρά φορτώματος και εκτέλεσης της εντολής. Ο αριθμός NNNNN πρέπει να είναι μικρότερος από 64000.

Στην συνέχεια ακολουθεί μία εντολή SUPER BASIC από αυτές που θα περιγραφούν αργότερα. Χωρίς να είναι υποχρεωτικό, αν ο προγραμματιστής θέλει, μπορεί να χρησιμοποιήσει και άλλες εντολές - BASIC στην ίδια γραμμή μέχρι να καλύψει το φυσικό της μέγεθος, ξεχωρίζοντας τις εντολές αυτές με το σύμβολο (:).

Το φυσικό μέγεθος της γραμμής BASIC είναι 72 χαρακτήρες το πολύ. Κάθε γραμμή φορτώνεται στην μνήμη κατά τον ακόλουθο τρόπο: Στην αρχή χρησιμοποιούνται 2 χαρακτήρες για τον αριθμό της, ακολουθούν 2 χαρακτήρες που δείχνουν την αρχή της επόμενης γραμμής BASIC (LIMITED ALLOCATION), στην συνέχεια καταχωρείται η εντολή ή οι εντολές της γραμμής καταλαμβάνοντας τόσους χαρακτήρες όσα αναλογούν στο μέγεθος της γραμμής και στο τέλος καταχωρείται ο χαρακτήρας 00H, που για τις εντολές της SUPER BASIC παίζει τον ρόλο του CR. Είναι λοιπόν προφανές ότι, αν σε μία γραμμή δοθούν περισσότερες της μίας εντολές έσω v, τότε κερδίζουμε σε μνήμη στο WORKSPACE 4*(v-1) χαρακτήρες.

1.4 Ομάδα χαρακτήρων

Η ομάδα χαρακτήρων της SUPER BASIC αποτελείται από τις εξής ομάδες:

α) Αλφαριθμητικοί χαρακτήρες : Λατινικοί και ελληνικοί
β) Αριθμητικοί χαρακτήρες
γ) Ειδικοί χαρακτήρες.

Οι αλφαβητικοί χαρακτήρες είναι κεφαλαία λατινικά γράμματα, όταν πρόκειται να χρησιμοποιηθούν για εντολές BASIC ή COMMANDS ή UTILITIES και αδιακρίτως κεφαλαία ή μικρά, λατινικά ή ελληνικά γράμματα όταν πρόκειται να χρησιμοποιηθούν για δεδομένα.

Οι αριθμητικοί χαρακτήρες είναι οι αριθμοί 0-9 και οποισδήποτε συνδιασμός τους όπως και τα σύμβολα + και -.

Ειδικοί χαρακτήρες είναι τα παρακάτω σύμβολα:

	Κενό
=	Ίσο
+	Σύμβολο πρόσθεσης
-	Σύμβολο αφαίρεσης
*	Σύμβολο πολλαπλασιασμού, (αστέρι)
/	Σύμβολο διαίρεσης, (πλάγια)
^	Σύμβολο εκθέτη.
(	Αριστερή παρένθεση
)	Δεξιά παρένθεση
%	Σήμα επί τοις εκατό
#	Σήμα δίεσης
$	Σήμα δολλαρίου

!	Θαυμαστικό
[	Αριστερή αγκύλη
]	Δεξιά αγκύλη
:	Ανω κάτω τελεία
;	Ερωτηματικό
,	Κόμμα
?	Λατινικό ερωτηματικό
<	Σύμβολο "μικρότερο"
>	Σύμβολο "μεγαλύτερο"
"	Απόστροφοι

1.5 Μεταβλητές

Ενας υπολογιστής δεν είναι παρά μία ηλεκτρονική μηχανή, που παίρνει πληροφορίες από τον έξω κόσμο (πληκτρολόγιο, δίσκοι κλπ) τις επεξεργάζεται με το πρόγραμμα που τρέχει εκείνη την στιγμή και στην συνέχεια επιστρέφει νέες πληροφορίες και αποτελέσματα στον έξω κόσμο.

Διασαφηνίζεται ότι έξω κόσμος για την SUPER BASIC είναι οτιδήποτε υπάρχει έξω από τον χώρο της μνήμης που καταλαμβάνει και εργάζεται η γλώσσα και το πρόγραμμα όπως π.χ οι μαγνητικοί δίσκοι, οι δισκέττες, η οθόνη, ο εκτυπωτής, ακόμη και μέρος της μνήμης που είναι εκτός χώρου BASIC (MEMORY DEVICE).

Οι πληροφορίες που διακινούνται μεταξύ της BASIC και του έξω κόσμου εκφράζονται με αριθμούς, σύμβολα και αλφαβητικούς χαρακτήρες ή και με συνδιασμό αυτών. Κάθε πληροφορία λοιπόν διακινείται μέσω των καλουμένων "μεταβλητών" της SUPER BASIC που συγκαταλέγονται σε 2 ομάδες : α) Τις αριθμητικές μεταβλητές
β) Τις αλφαριθμητικές μεταβλητές

1.5.1 Αριθμητικές μεταβλητές

Οι αριθμητικές μεταβλητές (NUMERIC) χρησιμοποιούνται για να επεξεργάζονται καθαρά αριθμητικές πληροφορίες. Τις αριθμητικές μεταβλητές μπορούμε να τις συνδιάσουμε και με τις θεμελιώδεις πράξεις της αριθμητικής όπως π.χ πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση ή και ακόμη με τις γνωστές μαθηματικές συναρτήσεις COS, SIN κλπ.

Συμβολίζονται με 2 το πολύ χαρακτήρες αλφαριθμητικούς που ο πρώτος χαρακτήρας κατ' ανάγκη θα είναι αλφαβητικός, ενώ ο δεύτερος αλφαβητικός ή αριθμητικός αδιάκριτα, π.χ AA,B,G1, FQ κλπ.

Μπορούμε να δηλώσουμε το όνομα μεταβλητής, με περισσότερα γράμματα αλλά μόνο τα πρώτα δύο λαμβάνονται υπόψη.

Κατά την αποθήκευσή τους καταλαμβάνουν 5 χαρακτήρες (4 για σημαντικό μέρος, "MANTISSA" και ένα για εκθέτη "EXPONENT") ενώ κατά την επεξεργασία τους καταλαμβάνουν 6 χαρακτήρες (τα 5 όπως περιγράφησαν ήδη και ένας για το λειτουργικό σύστημα για καθορισμό της ακρίβειας).

Για την έκδοση V3.2 και νεώτερες, οι αριθμοί στην αποθήκευσή τους καταλαμβάνουν 6 χαρακτήρες (επιτυγχάνοντας ακρίβεια 12 σημαντικών ψηφίων)

1.5.2 **Αλφαριθμητικές μεταβλητές**

Αλφαριθμητικές μεταβλητές (STRINGS) καλούνται εκείνες οι μεταβλητές που διακινούν οποιαδήποτε μή αριθμητική πληροφορία (π.χ ονόματα, διευθύνσεις κλπ). Οπως και οι αριθμητικές μεταβλητές έτσι και οι αλφαριθμητικές μεταβλητές συμβολίζονται με 2 το πολύ χαρακτήρες, που ο πρώτος υποχρεωτικά να είναι αλφαβητικός ενώ ο δεύτερος αδιάκριτα αλφαβητικός ή αριθμητικός, και ακολουθεί το σύμβολο δολλαρίου ($) π.χ AA$, B$, C1$, FQ$ κλπ.

Κατά την αποθήκευσή του καταλαμβάνουν τόσα BYTES όσοι είναι και οι χαρακτήρες του περιεχομένου του π.χ αν A$ ="NIKOS" τότε η A$ καταλαμβάνει 5 χαρακτήρες ενώ άν B$ = "10" η B$ καταλαμβάνει 2 χαρακτήρες κ.ο.κ

1.5.3 **Περιεχόμενο μεταβλητών**

Κάθε μεταβλητή μπορεί να πάρει ένα περιεχόμενο μόνο με το σύμβολο της ισότητας (=). Ετσι γράφουμε το όνομα της μεταβλητής, το σύμβολο της ισότητας και δεξιά την τιμή που θέλουμε να πάρει τη μεταβλητή ως εξής:

α) Αν η μεταβλητή είναι αριθμητική γράφουμε τον αριθμό.

β) Αν η μεταβλητή είναι αλφαριθμητική γράφουμε άνω αποστρόφους ("), το STRING και πάλι άνω αποστρόφους.

Παραδείγματα:

```
A = 3, B = 45, ER = A
A$="NOKOS", B$="15", ER$="A500", A1$=B$
```

1.5.4 Πολυδιάστατες μεταβλητές (ARRAYS)

Πολυδιάστατες μεταβλητές καλούνται εκείνες που έχουν κοινό όνομα και ξεχωρίζουν μεταξύ τους με κάποιο δείκτη, που είναι ένας φυσικός αριθμός ή μια αριθμητική μεταβλητή που εκφράζει φυσικό αριθμό. Τέτοιες μεταβλητές μπορεί να είναι αριθμητικές ή αλφαριθμητικές και μπορούν να καθορίζονται από μία διάσταση, δύο διαστάσεις π.χ A(J), T$(I,J) ή περισσότερες, όπου το I,J είναι αριθμητικές προκαθορισμένες μεταβλητές.

Η SUPER BASIC έχει την δυνατότητα να χρησιμοποιεί με το ίδιο όνομα μία αριθμητική και μία αλφαριθμητική πολυδιάστατη μεταβλητή χωρίς να υπάρχει σύγχιση π.χ A$(I), A(J) μπορούν να χρησιμοποιηθούν ταυτόχρονα στο ίδιο πρόγραμμα.

Οταν σε ένα πρόγραμμα χρησιμοποιούνται πολυδιάστατες μεταβλητές, πριν αυτές χρησιμοποιηθούν πρέπει να δηλωθεί η διάστασή τους στην αρχή του προγράμματος (βλέπε την εντολή DIM)

Σε περίπτωση που δεν έχει γίνει δήλωση της διάστασης ή των διαστάσεων και χρησιμοποιήσουμε κάποια πολυδιάστατη μεταβλητή στο πρόγραμμα, τότε η SUPER BASIC δημιουργεί δυναμικά και αυτόματα διάσταση ή διαστάσεις για την μεταβλητή αυτή με μέγιστο δείκτη 10 δηλαδή A(10) ή B$(10,10) κλπ.

Μόλις ορισθεί κάποια πολυδιάστατη μεταβλητή, που ουσιαστικά είναι ένας πίνακας, όλα του τα στοιχεία αρχικά παίρνουν την τιμή 0 αν είναι αριθμητικά και κενό αν είναι αλφαριθμητικά. Στην συνέχεια μπορούμε να δώσουμε διάφορες τιμές.

Οσον αφορά την διάσταση ενός πίνακα, δεν υπάρχει κανένας περιορισμός παρά μόνο από τον διαθέσιμο χώρο της μνήμης.

1.6 Εκφράσεις και τελεστές πράξεων

Μία έκφραση μπορεί να είναι μία αριθμητική μεταβλητή, ή σύνολο αριθμητικών μεταβλητών που συνδιάζονται με τις θεμελιώδεις μαθηματικές πράξεις, ή και ακόμη με μαθηματικές συναρτήσεις και θεμελιώδεις πράξεις.

Παραδείγματα

Αλγεβρική έκφραση	SUPER BASIC έκφραση
$\chi+2\psi$	χ+2*ψ
$\chi-\frac{\psi}{\xi}$	χ-ψ/ξ
$\frac{\chi+\psi}{\xi}$	(χ+ψ)/ξ
$(\chi^{2})^{\psi}$	(χ∧2)∧ψ
$\chi(-\psi)$	χ* (-ψ)

```
A$=LEFT$ (A$,10) + "*"+MID$(A$,3,2,) + STR$(χ)
B$=LEFT$(A$+"0000",5)
```

1.6.1 Τελεστές πράξεων

Οι τελεστές πράξεων είναι τα ειδικά εκείναι σύμβολα που καθορίζουν μία αριθμητική πράξη (θεμελιώδη). Οι τελεστές δίδονται στην συνέχεια, με τέτοια διάταξη που καθορίζει την προτεραιότητα εκτέλεσης των, όταν ευρίσκονται ταυτόχρονα σε μία έκφραση.

	Τελεστής	Λειτουργία	Απλή έκφραση
1.	∧	Υψωση σε δύναμη	∧
2.	-	Δήλωση αρνητικού αριθμού	-χ
3.	*	Πολλαπλασιασμός	χ*ψ
4.	/	Διαίρεση (FLOAT-ING POINT)	χ/ψ
5.	+	Πρόσθεση	χ+ψ
6.	-	Αφαίρεση	χ-ψ

Για να γίνει αλλαγή της διάταξης προτεραιότητας εκτέλεσης σε μία έκφραση, πρέπει να χρησιμοποιηθούν παρενθέσεις. Ετσι οι τελεστές μέσα στις παρενθέσεις εκτελούνται πρώτοι, με τη φυσική τους διάταξη φυσικά, δηλαδή ύψωση σε δύναμη, πολλαπλασιασμός-διαίρεση, πρόσθεση-αφαίρεση και κατόπιν εκτελούνται οι τελεστές έξω από τις παρενθέσεις.

Στο παράδειγμα που ακολουθεί, δίδεται η προτεραιότητα της εκτέλεσης των τελεστών, με τον αριθμό που ευρίσκεται κάτω από το αντίστοιχο τελεστή.

```
A = B  + (-χ) * ((X+(A-B/X)^3)/5))
       8  6   7    4  2 1  3  5
```

1.6.2 Τελεστές σύγκρισης

Εκτός από τους τελεστές πράξεων, ορίζονται και οι τελεστές σύγκρισης. Με τους τελεστές σύγκρισης επιτυγχάνεται η σύγκριση μεταξύ των μεταβλητών ή εκφράσεων ή και συνδιασμό μεταβλητών και εκφράσεων. Μετά από μία σύγκριση το αποτέλεσμα μπορεί να είναι αληθές (TRUE) ή ψευδές (FALSE) και ανάλογα λαμβάνεται κάποια απόφαση σχετικά με την ροή του προγράμματος.

Τελεστής	Λειτουργία	Απλή έκφραση
=	Ισότητα	χ = ψ
< >	Ανισότητα	χ< >ψ
<	Μικρότερο από	χ < ψ
>	Μεγαλύτερο από	χ > ψ
<=	Μικρότερο ή ίσο από	χ <=ψ
>=	Μεγαλύτερο ή ίσο από	χ >=ψ

Οι σχέσεις σύγκρισης <= και >= μπορούν να χρησιμοποιηθούν και με την μορφή =< και => αντίστοιχα.

Αν σε κάποια έκφραση έχουμε σύγκριση μεταβλητής και έκφρασης, τότε προτεραιότητα έχουν οι αριθμητικοί τελεστές και κατόπιν εκτελούνται οι τελεστές σύγκρισης.

Παραδείγματα

1. 15 IF χ <> ψ THEN 100 σημαίνει ότι αν χ =/ψ, το πρόγραμμα θα συνεχίσει την εκτέλεσή του από την με αριθμό 100, αλλιώς συνεχίζει με την αμέσως επόμενη εντολή.

2. 10 IF χ >= A+B*C THEN 300 σημαίνει ότι αν η τιμή του χ είναι μεγαλύτερη ή ίση από την τιμή που προκύπτει, με την εκτέλεση των πράξεων στην έκφραση A+B*C, τότε το πρόγραμμα θα συνεχίσει την εκτέλεσή του από την εντολή 300, διαφορετικά συνεχίζει με την αμέσως επόμενη εντολή.

1.6.3 Λογικοί τελεστές

Οι λογικοί τελεστές ή αλλιώς Μπουλιανοί τελεστές καθορίζουν σύγκριση μεταξύ BYTES, με την έννοια BIT BY BIT και ανάλογα με το αποτέλεσμα της πράξης (FALSE ή TRUE) λαμβάνεται απόφαση σχετικά με την ροή του προγράμματος.

Ουσιαστικά οι λογικοί τελεστές καθορίζουν BOOLEAN πράξεις μεταξύ αριθμών ή αποτελεσμάτων σύγκρισης.

α) **Λογικός τελεστής NOT**

χ	NOT χ
1	0
0	1

β) **Λογικός τελεστής AND**

χ	ψ	χ AND ψ
1	1	1
1	0	0
0	1	0
0	0	0

γ) **Λογικός τελεστής OR**

χ	ψ	χ OR ψ
1	1	1
1	0	1
0	1	1
0	0	0

ΚΕΦΑΛΑΙΟ 2

ΛΕΙΤΟΥΡΓΙΕΣ (COMMANDS) ΤΗΣ SUPER BASIC

Υπάρχουν ορισμένες λειτουργίες (COMMANDS) που αναγνωρίζονται από την SUPER BASIC και εκτελούν αυτόματα, μόλις κληθούν, ένα συγκεκριμένο κομμάτι του λειτουργικού συστήματος ή αλοιώνουν το λειτουργικό σύστημα, ώστε να επιτευχθεί κάποιος σκοπός.

Οι λειτουργίες αυτές εκτελούνται από την θέση MONITOR δήλαδή στην οθόνη εμφανίζεται η ένδειξη GIGATRONICS, ή σαν εντολές μέσα σε κάποιο πρόγραμμα.

2.1 CLEAR

ΔΟΜΗ : CLEAR

ΣΚΟΠΟΣ : Μηδενίζει τα περιεχόμενα όλων των μεταβλητών.

ΠΕΡΙΓΡΑΦΗ : Οταν εκτελείται ένα πρόγραμμα στην αρχή όλες οι μεταβλητές καθαρίζονται αυτομάτως ενώ κατά την εκτέλεση του προγράμματος, αυτές λαμβάνουν ποικίλες τιμές. Στις περιπτώσεις που απαιτείται να καθαρισθούν οι μεταβλητές μετά την εκτέλεση του προγράμματος, χρησιμοποιείται η εντολή CLEAR. Δίδοντας λοιπόν την εντολή CLEAR όλες οι μεταβλητές καθαρίζουν δηλαδή οι αριθμητικές μεταβλητές παίρνουν την τιμή 0, ενώ οι αλφαριθμητικές παίρνουν την τιμή κενό.

2.2 CONT

ΔΟΜΗ : CONT

ΣΚΟΠΟΣ : Συνεχίζει την εκτέλεση κάποιου προγράμματος από το σημείο που είχε διακοπεί μετά από το CONTROL-C ή την εντολή STOP

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή δεν μπορεί να ευρίσκεται μέσα στο πρόγραμμα.

ΠΡΟΣΟΧΗ !! Αν το πρόγραμμα που εκτελείτο και διακόπηκε είχε τεμαχίσει την μνήμη, τότε μετά την διακοπή του από το CONTROL-C ή την εντολή STOP ο τεμαχισμός δεν ισχύει. Συνεπώς αν χρησιμοποιείται τεμαχισμένη μνήμη και γίνει μία διακοπή με STOP, οι αμέσως επόμενες εντολές ή εντολή, πρέπει να ξανακαθαρίζουν τον τεμαχισμό της μνήμης, οπότε δίνοντας την εντολή CONT να εξακολουθεί να είναι τεμαχισμένη η μνήμη, όπως ήταν πρίν.
π.χ. POKE133,208:CONT

2.3 DSK

ΔΟΜΗ : DSK "<ΟΝΟΜΑ ΔΙΣΚΟΥ>"

ΣΚΟΠΟΣ : Προσδιορίζει την μονάδα δίσκου με την οποία θέλουμε να επικοινωνήσουμε.

Παραδείγματα

DSK"α <RETURN>
από τη θέση MONITOR

10 DSK"A": OPEN"NIOBH1",1:INDEX<1>=20 <RETURN>
μέσα σε πρόγραμμα

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή χρησιμοποιείται για να καθορίσουμε την μονάδα δίσκου, με την οποία θέλουμε να επικοινωνήσουμε. Τα ονόματα μονάδων δίσκων που χρησιμοποιούνται είναι:

E : πρώτος HARD DISK FIX DRIVER
F : πρώτος HARD DISK REMOVABLE DRIVER
G : δεύτερος HARD DISK FIX DRIVER
H : δεύτερος HARD DISK REMOVABLE DRIVER

E, F, G, H ανάλογα με τον υπολογιστή

Οπως φαίνεται στα παραδείγματα αν χρησιμοποιείται η εντολή DSK από την θέση MONITOR, οι δεύτεροι άνω απόστροφοι δεν είναι απαραίτητοι ενώ αντίθετα αν χρησιμοποιείται μέσα στο πρόγραμμα, το όνομα του δίσκου απαραιτήτως πρέπει να περικλείεται σε αποστρόφους. Αν πρίν χρησιμοποιήσουμε την εντολή DSK σε κάποιο πρόγραμμα, καθορίσουμε σε μία αλφαριθμητική μεταβλητή το όνομα του δίσκου, τότε χρησιμοποιώντας την εντολή DSK δεν χρειάζονται καθόλου απόστροφοι.

Παράδειγμα

```
10 FD$="A":DSK FD$: RUN"PROG"
```

2.4 FLAG

ΔΟΜΗ : FLAG <αριθμός>

ΣΚΟΠΟΣ : (σε ειδικό έντυπο με ονομασία FLAGS αναλύονται όλα τα FLAGS που χρησιμοποιεί η SUPER BASIC)

2.5 LIST

α) LIST
β) LIST -<αριθμός γραμμής>
γ) LIST <αριθμός γραμμής>-
δ) LIST <αριθμός γραμμής>-<αριθμός γραμμής)
ε) LIST <αριθμός καναλιού εξόδου>
στ) LIST#<αριθμός καναλιού εξόδου>,-<αριθμός γραμμής>
ζ) LIST#<αριθμός καναλιού εξόδου>,<αριθμός γραμμής>-
η) LIST#<αριθμός καναλιού εξόδου>,<αριθμός γραμμής>-<αριθμός γραμμής>

ΣΚΟΠΟΣ : Εμφανίζει το πρόγραμμα που υπάρχει στην κεντρική μνήμη, σε κάποιο τερματικό (οθόνη, σειριακό εκτυπωτή, παράλληλο εκτυπωτή), με όλες του τις γραμμές ή μερικές από αυτές.

Παραδείγματα

1. **LIST**

Εμφανίζει όλες τις γραμμές του προγράμματος στην οθόνη

2. **LIST-100**

Εμφανίζει τις γραμμές μέχρι και την γραμμή που έχει αριθμό 100, στην οθόνη.

3. **LIST 100-**

Εμφανίζει τις γραμμές από αυτή που έχει αριθμό 100 μέχρι τέλους του προγράμματος στην οθόνη.

4. **LIST 100-200**

Εμφανίζει τις γραμμές από αυτή που έχει αριθμό 100 μέχρι και αυτή που έχει αριθμό 200 στην οθόνη.

5. **LIST#W,100-100** όπου

W = 5 : παράλληλος εκτυπωτής
W = 6 : σειριακός εκτυπωτής
W = 4 : μνήμη

2.6 LOAD

ΔΟΜΗ : LOAD<όνομα αρχείου>,<PASSWORD=κλειδί>

ΣΚΟΠΟΣ : Φορτώνει ένα πρόγραμμα BASIC από ένα αρχείο δίσκου στην κεντρική μνήμη

Παραδείγματα

α) LOAD"TEST ή LOAD"TEST"
β) LOAD"PROG", "KEY1"

ΠΕΡΙΓΡΑΦΗ : Για να φορτώσουμε ένα πρόγραμμα BASIC από κάποιο αρχείο δίσκου στην κεντρική μνήμη, χρησιμοποιούμε την εντολή LOAD. Πριν όμως την χρησιμοποιήσουμε, πρέπει να καθορίσουμε τον δίσκο που περιέχει το αρχείο που ζητάμε και αυτό γίνεται με την εντολή DSK. Κατόπιν φορτώνουμε το πρόγραμμα στην κεντρική μνήμη με την εντολή LOAD, οπότε μπορούμε να το εκτελέσουμε με την εντολή RUN ή να δούμε τις εντολές του με την εντολή LIST.

Σε περίπτωση που το αρχείο είναι κλειδωμένο δηλαδή εκτός από το όνομά του έχει και κάποιο κλειδί, τότε για να εκτελεσθεί η εντολή LOAD, πρέπει μαζί με το όνομα αρχείου, να δοθεί και το σωστό κλειδί. Στο δεύτερο παράδειγμα το όνομα του αρχείου είναι PROG και το όνομα του κλειδιού KEY1. Και το όνομα του αρχείου και το όνομα του κλειδιού πρέπει να περικλείονται σε άνω αποστρόφους. Αν το αρχείο δεν έχει κλειδί τότε οι δεξιοί απόστροφοι δεν είναι απαραίτητοι.

2.7 NEW

ΔΟΜΗ : NEW

ΣΚΟΠΟΣ : Σβήνει από την κεντρική μνήμη κάθε πρόγραμμα που τυχόν υπάρχει και ταυτόχρονα καθαρίζει τις μεταβλητές.

ΠΕΡΙΓΡΑΦΗ : Η εντολή NEW σβήνει από την κεντρική μνήμη κάθε πρόγραμμα που προυπήρχε, εκτος από το λειτουργικό σύστημα και την γλώσσα BASIC, ενώ ταυτόχρονα καθαρίζει τις μεταβλητές.

Συνεπώς, προτού αρχίσουμε να πληκτρολογούμε ένα πρόγραμμα, πρέπει να είμαστε βέβαιοι, ότι στην κεντρική μνήμη δεν υπάρχει κανένα άλλο, γιατί τότε οι εντολές του νέου προγράμματος θα αναμειχθούν με εκείνες του παλαιού. Για το σκοπό αυτό πρέπει απαραίτητα προηγουμένως να χρησιμοποιήσουμε την εντολή NEW.

2.8 PEEK

ΔΟΜΗ : PEEK <διεύθυνση μνήμης>

ΣΚΟΠΟΣ : Ανάγνωση περιεχομένου μιας διεύθυνση μνήμης.

Παράδειγμα

```
PRINT PEEK(45000)
```

ΠΕΡΙΓΡΑΦΗ : Η εντολή PEEK επιτρέπει την ανάγνωση του περιεχομένου μιάς διεύθυνσης στην κεντρική μνήμη. Την διεύθυνση της μνήμης την δίνουμε με κάποιο αριθμό του δεκαδικού συστήματος, γραμμένο μέσα σε παρενθέσεις.

2.9 PRL

ΔΟΜΗ : PRL(χ)

ΣΚΟΠΟΣ : Υπολογίζει τον χώρο μνήμης που καταλαμβάνει ένα πρόγραμμα.

ΠΕΡΙΓΡΑΦΗ : Η εντολή PRL υπολογίζει το μέγεθος του προγράμματος, που βρίσκεται την στιγμή αυτή στην μνήμη. Μέσα στην παρένθεση μπορούμε να γράψουμε οποιονδήποτε αριθμό, χωρίς αυτός να παίξει κανένα ρόλο. Αν το πρόγραμμα που είναι στην μνήμη έχει εκτελεσθεί, τότε η PRL θα υπολογίσει τον συνολικό χώρο που καταλαμβάνει το πρόγραμμα κατά την εκτέλεσή του. Στην περίπτωση αυτή, αν θέλουμε να μην υπολογιστεί ο χώρος που καταλαμβάνουν οι μεταβλητές, αρκεί να δώσουμε προηγουμένως την εντολή CLEAR που αποδεσμεύει το χώρο των μεταβλητών.

2.10 POKE

ΔΟΜΗ : POKE <Διεύθυνση μνήμης>,<αριθμός>

ΣΚΟΠΟΣ : Γράφει ένα δεκαδικό αριθμό σε μία συγκεκριμένη διεύθυνση της μνήμης.

Παράδειγμα

POKE 37000,123

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή χρησιμοποιείται για την δραστική επέμβαση σε μία συγκεκριμένη διεύθυνση μνήμης για να γράψουμε έναν αριθμό. Η διεύθυνση της μνήμης δίδεται με έναν αριθμό του δεκαδικού συστήματος.
Επειδή ένα BYTE (λέξη μνήμης) έχει μήκος 8 BITES, ο αριθμός που θα γράψουμε θα γράψουμε θα πρέπει να είναι μεταξύ του 0 και 255.

Χρησιμοποιώντας την εντολή POKE υπάρχει κίνδυνος να αλλοιωθεί καταστροφικά το λειτουργικό σύστημα ή η γλώσσα SUPER BASIC που είναι φορτωμένη στην μνήμη άν η διεύθυνση μνήμης που επεμβαίνει ο προγραμματιστής είναι μικρότερη του 25000, γιαυτό απαιτείται ιδιαίτερη προσοχή στην χρήση της.

Οταν σε ένα πρόγραμμα χρησιμοποιηθεί η εντολή POKE και το πρόγραμμα εκτελεσθεί, μόλις το σύστημα επανέλθει στην θέση MONITOR δηλαδή στην οθόνη εμφανίζεται η ένδειξη GIGATRONICS, τότε το POKE παύει να ισχύει και η BASIC αυτόματα ξαναπαίρνει την φυσική της κατάσταση.

Σε ειδικό έντυπο με την ονομασία POKES αναλύεται η χρήση των εντολών POKE και PEEK που χρησιμοποιούνται στην SUPER BASIC

2.11 PUTD

ΔΟΜΗ : PUTD
<"όνομα αρχείου">,<[διεύθυνση αρχείου]>,
<αριθμός χαρακτήρων>,<διεύθυνση μνήμης>.

ΣΚΟΠΟΣ : Κάνει αυτόματη μεταφορά δεδομένων από το MEMORY DEVICE στο αρχείο κάποιου δίσκου.

Παράδειγμα: PUTD"PELAT"[200],6000,37000

ΠΕΡΙΓΡΑΦΗ : Αν θέλουμε να φυλάξουμε πληροφορίες σε ένα αρχείο δίσκου μπορούμε να τις φράφουμε στο MEMORY DEVICE και αργότερα τις μεταφέρουμε στο αρχείο με την εντολή PUTD.

Στο παράδειγμα, το "PELAT" είναι το όνομα του αρχείου που απαραίτητα πρέπει να περικλείεται σε αποστρόφους; 200 είναι η διεύθυνση του αρχείου από όπου θα αρχίσουν να καταχωρούνται οι πληροφορίες.
Η διεύθυνση γράφεται με έναν αριθμό του δεκαδικού συστήματος, μέσα σε αγκύλες.

Ο 6000 (στο δεκαδικό σύστημα) είναι ο αριθμός των χαρακτήρων που θέλουμε να περάσουν από το MEMORY DEVICE στο αρχείο δίσκου.

Ο 37000 (στο δεκαδικό σύστημα) είναι η αρχική διεύθυνση του MEMORY DEVICE,όπου είναι φυλαγμένες οι πληροφορίες.

2.12 PUTM

ΔΟΜΗ : PUTM
<"όνομα αρχείου">,<[αρχική διεύθυνση αρχείου]>,
<αριθμός χαρακτήρων>,<αρχική διεύθυνση μνήμης>.

ΣΚΟΠΟΣ : Μεταφορά πληροφοριών από ένα αρχείο δίσκου στην μνήμη (MEMORY DEVICE)

Παράδειγμα: PUTM"PELAT"[20],600,34000

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή είναι πολύ χρήσιμη στην επεξεργασία αρχείων και σχετίζεται με την MEMORY DEVICE.
Ξέχωρα από την κλασσική επικοινωνία με τα αρχεία δίσκων μπορούμε με την PUTM να μεταφέρουμε μία περιοχή ενός αρχείου σε ένα κομμάτι μνήμης που το θεωρούμε σαν έξω κόσμο της BASSIC (MEMORY DEVICE). Τότε χρησιμοποιόντας το κανάλι (INPUT/OUTPUT) 4 επικοινωνίας με την MEMORY DEVICE, μπορούμε να διαβάσουμε ή να αλλοιώσουμε πληροφορίες. μέσα στην μνήμη, με πολύ μεγαλύτερη ταχύτητα, απ΄ότι άν εργαζόμασταν στον δίσκο.

Αλλη περίπτωση χρήσης της εντολής PUTM είναι όταν σε ένα OTHER αρχείο έχουμε φυλάξει πρόγραμμα γραμμένο σε γλώσσα μηχανής (MACHINE -CODE) και θέλουμε να το μεταφέρουμε σε μία περιοχή της μνήμης για να το εκτελέσουμε με την βοήθεια της εντολής χ = COD(χ)

2.13 REN (Δεν είναι σε χρήση)

ΔΟΜΗ : REN

ΣΚΟΠΟΣ : Αλλάζει τους αριθμούς γραμμών ενός προγράμματος δίνοντας νέους, αρχίζοντας από 10 και αυξάνοντας κατά 10

ΠΕΡΙΓΡΑΦΗ : Η εντολή REN δεν μπορεί να ευρίσκεται μέσα σε πρόγραμμα. Οι αριθμοί γραμμής που ορίζονται σε εντολές GOTO, GOSUB, ON....GOTO κλπ δεν αλλάζουν αλλά πρέπει να τους αλλάξουμε με το χέρι.

Παράδειγμα:

```
 1 GOTO 15                      :
10 PRINT"ΠΡΟΓΡΑΜΜΑ TEST"        : πρίν την χρήση
11 PRINT:PRINT:RETURN           : της COMMAND REN
15 GOSUB 1:END                  :


10 GOTO 15                      :
20 PRINT"ΠΡΟΓΡΑΜΜΑ TEST"        : μετά την χρήση
30 PRINT:PRINT:RETURN           : της COMMAND REN
40 GOSUB 1:END                  :


10 GOTO 40                      : μετά την διόρθωση
20 PRINT"ΠΡΟΓΡΑΜΜΑ TEST"        : των αριθμών στις
30 PRINT:PRINT:RETURN           : εντολές GOTO και
40 GOSUB 10:END                 : GOSUB
```

2.14 RUN

ΔΟΜΗ : RUN ή
RUN<αριθμός γραμμής> ή
RUN<"όνομα αρχείου">,<αριθμός γραμμής>

Παράδειγμα:

```
α)  RUN
β)  RUN 30
γ)  RUN"SORT",2
```

ΠΕΡΙΓΡΑΦΗ : Αν ένα πρόγραμμα ευρίσκεται σε αρχείο δίσκου τότε αμέσως με την εντολή RUN ακολουθεί το όνομα του αρχείου, μέσα σε εισαγωγικά, στο οποίο ευρίσκεται το πρόγραμμα. Η εκτέλεση του προγράμματος θα αρχίσει από τον αριθμό γραμμής που προδιορίζουμε. Αν δεν δώσουμε τέτοιο αριθμό, το πρόγραμμα θα εκτελεσθεί από την πρώτη εντολή του.

2.15 SAVE

ΔΟΜΗ : SAVE <"όνομα αρχείου">,<"κλειδί">

ΣΚΟΠΟΣ : Φύλαξη ενός προγράμματος BASIC από την μνήμη σε ένα αρχείο BASIC.

Παράδειγμα: SAVE"TEST","PARA"
SAVE"PROG"

ΠΕΡΙΓΡΑΦΗ : Αν έχουμε γράψει ένα πρόγραμμα BASIC στην κεντρική μνήμη, μπορούμε να το φυλάξουμε σε ένα αρχείο BASIC στον δίσκο, με την χρήση της εντολής SAVE. Ετσι, γράφουμε την εντολή SAVE ακολούθως το όνομα του αρχείου σε εισαγωγικά και άν το αρχείο είναι προστατευμένο πρέπει στην συνέχεια να γράψουμε το σωστό κλειδί (=PASSWORD) και αυτό φυσικά κλεισμένο σε εισαγωγικά.

ΠΡΟΣΟΧΗ !!! **Για να γίνει SAVE ενός προγράμματος σε ένα αρχείο πρέπει το αρχείο αυτό να έχει δημιουργηθεί στον δίσκο από πρίν, με την UTILITY CREATE. Συνεπώς πρέπει να ακολουθεί η διαδικασία.**

1) DSK"*":RUN:CREATE, όπου *=όνομα δίσκου
2) NEW
3) Γράψιμο προγράμματος BASIC
4) SAVE"******","#####"
   όπου "******"=όνομα αρχείου και
   "#####"=κλειδί αρχείου άν υπάρχει

2.16 EDIT

<u>ΔΟΜΗ</u> : EDIT

Η SUPER BASIC δεν διαθέτει ένα συγκεκριμένο EDITOR σαν πρόγραμμα ώστε να μπορεί, μέσω αυτής ο προγραμματιστής να κάνει διορθώσεις σε κάποιο πρόγραμμα.

Εν τούτοις τα συστήματα GIGATRONICS σε μία προσπάθεια να αξιοποιηθούν πλήρως οι δυνατότητές τους, χρησιμοποιούν για EDITOR ορισμένες λειτουργίες που επιτυγχάνονται από την κατάλληλη χρήση των ειδικών πλήκτρων του πληκτρολογίου.

Στην συνέχεια δίδονται πληροφορίες και οδηγίες χρήσης των ειδικών αυτών πλήκτρων, που παρέχουν την δυνατότητα για EDITING σε ένα πρόγραμμα BASIC.

Στο εγχειρίδιο των Τερματικών (ΕΚΑΤΗ) αναλύονται όλες οι ιδιότητες και ικανότητες του τερματικού που είναι αποδεκτές από τον EDITOR της SUPER BASIC.

ΚΕΦΑΛΑΙΟ 3

ΕΝΤΟΛΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ SUPER BASIC

Στο κεφάλαιο αυτό δίδονται όλες οι θεμελιώδεις εντολές που δομούν την SUPER BASIC.

Παράλληλα μετά την περιγραφή των εντολών, δίνονται διάφοροι συνδιασμοί αυτών, που εξυπηρετούν συγκεκριμμένους σκοπούς και καθιστούν την SUPER BASIC δυναμική και πλήρη γλώσσα, υψηλού επιπέδου.

Στον καθορισμό των εντολών ακολουθείται η εξής φόρμουλα:

α. **ΔΟΜΗ** : δείχνει το σωστό τρόπο γραφής της εντολής

β. **ΣΚΟΠΟΣ** : αναφέρει τον σκοπό που εξυπηρετεί η εντολή.

γ. **ΠΕΡΙΓΡΑΦΗ** : περιγράφει την λειτουργία της εντολής, για την πλήρη αξιοποίησή της.

δ. **ΠΑΡΑΔΕΙΓΜΑ** : δίδεται τουλάχιστον ένα παράδειγμα χρήσης της εντολής για καλύτερη κατανόηση αυτής.

3.1 COD(χ)

ΔΟΜΗ : χ = COD(χ)

ΣΚΟΠΟΣ : Σύνδεση ενός προγραμματος BASIC, με ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής (MACHINE CODE)

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή επιτρέπει την επικοινωνία ενός προγράμματος BASIC με ένα πρόγραμμα (ρουτίνα) γραμμένο σε γλώσσα μηχανής (ASSEMBLY) για τον μικρουπολογιστή (MICROPROCESSOR) 6502 της MOTOROLA

Η διεύθυνση που αρχίζει η ρουτίνα αυτή δηλώνεται με την COMMAND POKE, ώς εξής:

POKE 8778, LOW
POKE 8779, HIGH

όταν εκτελεστεί η ρουτίνα και φθάσει στην εντολή RTS (RETURN για τον ASSEMBLER 6502), συνεχίζεται η εκτέλεση του προγράμματος BASIC από το σημείο κλήσης της ρουτίνας ASSEMBLY και κάτω.

Παράδειγμα :

```
 90 POKE 8778:POKE 8779,132
100 ER=COD(1)
110 PRINT#ER,"NIKOS"
```

Δηλαδή από την διεύθυνση 34000(=132 256+0) και πάνω έχει φορτωθεί η ρουτίνα ASSEMBLY που καθορίζει το κανάλι εξόδου ER.

3.2 CLOSE

ΔΟΜΗ : CLOSE ή
CLOSE<αριθμό καναλιού αρχείου>

ΣΚΟΠΟΣ : Κλείνει το κανάλι (ή **όλα** τα κανάλια) επικοινωνίας δίσκου.

ΠΕΡΙΓΡΑΦΗ : Οταν τελειώσουμε την επεξεργασία δεδομένων, επικοινωνόντας με το αρχείο δίσκου που τα **περιέχει**, και δεν χρειαζόμαστε άλλο αρχείο αυτό πρέπει να το "κλείσουμε"
Χρησιμοποιώντας λοιπόν την εντολή CLOSE και τον αριθμό καναλιού επικοινωνίας με το αρχείο επιτυγχάνουμε 2 σκοπούς.
Αφ΄ μέν κλείνουμε το αρχείο, οπότε οι τελευταίες πληροφορίες που γράψαμε στο αρχείο περνάνε στο δίσκο, αφ΄ετέρου δε, απελευθερώνουμε ένα κανάλι επικοινωνίας με τα αρχεία για επαναχρησιμοποίησή του αν απαιτείται.
Αν μετά την εντολή CLOSE δεν γράψουμε κανένα αριθμό καναλιού, τότε αυτόματα θα κλείσουν όλα τα κανάλια επικοινωνίας αρχείων που ήταν ανοικτά.

Παράδειγμα :

```
10 OPEN"PELAT",1:INPUT%1,E1:INDEX<1>=10
20 PRINT%1,E1:CLOSE 1:END
```

3.3 DATA

ΔΟΜΗ : DATA
<πληροφορ(α-1>,....,<πληροφορ(α-η>

ΣΚΟΠΟΣ : Αποθήκευση πληροφοριών μέσα στο πρόγραμμα BASIC.

ΠΕΡΙΓΡΑΦΗ : Την εντολή DATA την χρησιμοποιούμε όταν θέλουμε να αποθηκεύσουμε πληροφορ(ες (δεδομένα) μέσα σε ένα πρόγραμμα BASIC που το (διο το πρόγραμμα θα χρησιμοποιήσει τις πληροφορ(ες αυτές.
Αμέσως μετά την εντολή DATA γράφουμε τις πληροφορ(ες χωρ(ζοντας τες σε κομμάτια. Αν οι πληροφορ(ες ε(ναι αριθμητικές απλώς γράφουμε τους αριθμούς, αν όμως ε(ναι αλφαριθμητικές πρέπει να περικλε(ονται σε άνω διπλούς αποστρόφους.
Σε περ(πτωση που οι πληροφορ(ες ε(ναι αρκετές και δεν χωράνε σε μ(α γραμμή BASIC, τότε μπορούμε να χρησιμοποιήσουμε πολλές γραμμές με την εντολή DATA.

Παράδειγμα:

```
1000 DATA 10,5.1,"NIKOS", 100
1005 DATA "BIKENTIOΣ", 5.2,16
```

3.4 DIM

ΔΟΜΗ : DIM
<όνόμα π(νακα (πλήθος στοιχε(ων)>

ΣΚΟΠΟΣ : Ορισμός των στοιχε(ων ενός π(νακα και προκαθορισμός του χώρου της μνήμης για τον π(νακα αυτόν.

ΠΕΡΙΓΡΑΦΗ : Με την εντολή αυτή ορίζεται ένας πίνακας αριθμητικών ή αλφαριθμητικών μεταβλητών, στο οποίο δίνουμε κάποιο όνομα. Δεξιά από το όνομα του πίνακα και μέσα σε παρένθεση γράφουμε την μέγιστη τιμή του δείκτη, για κάθε διάστασή του.

Μόλις ορίσουμε έναν πίνακα, όλα του τα στοιχεία αρχικά παίρνουν τιν τιμή 0 τα αριθμητικά, ενώ τα αλφαριθμητικά την τιμή κενό. Αν δεν χρησιμοποιήσουμε την εντολή DIM για να ορίσουμε έναν πίνακα, μπορούμε δυναμικά να τον ορίσουμε άν δηλώσουμε κάποιο στοιχείο του. Ετσι αν γράψουμε A(5)=100 χωρίς να προηγηθεί η εντολή DIM A(χ) τότε αυτόματα δημιουργείται ο πίνακας A(χ). Σε αυτή την περίπτωση δημιουργίας ενός πίνακα, για κάθε διάστασή του ο μέγιστος δείκτης είναι 10.

Περιορισμός στην διάσταση ενός πίνακα δεν υπάρχει, παρά μόνο από το διαθέσιμο χώρο στην μνήμη. Δεν επιτρέπεται να χρησιμοποιήσουμε την εντολή DIM για κάποιο πίνακα που ήδη έχει οριστεί από πρίν, γιατί τότε έχουμε λάθος (DD=DOUBLE DIMENS). Επίσης σε κάποιον καθορισμένο πίνακα, δεν επιτρέπεται να χρησιμοποιήσουμε κάποιο στοιχείο του με δείκτη μεγαλύτερο από το προκαθορισμένο, γιατί θα παρουσιασθεί λάθος (BS=BAD SUBSCRIPT).

Ενας πίνακας διπλής διάστασης, αποθηκεύεται στην μνήμη κατά γραμμές δηλαδή κάθε γραμμή έχει σταθερή την πρώτη διάσταση και "εξαντλεί" την δεύτερη διάσταση.

ΠΑΡΑΔΕΙΓΜΑ

ο Β (3,4) αποθηκεύεται ώς εξής:

```
B(0,0),B(0,1),B(0,2),B(0,3),B(0,4)
B(1,0),B(1,1),B(1,2),B(1,3),B(1,4)
.................................
B(3,0),B(3,1),B(3,2),B(3,3),B(3,4)
```

ΠΑΡΑΔΕΙΓΜΑ

DIM A(15), B$(5,8), C$(7) κλπ
είναι πίνακες διαστάσεων αντίστοιχα 16 στοιχείων, 54 στοιχείων και 8 στοιχείων.

3.5 END

ΔΟΜΗ : END

ΣΚΟΠΟΣ : Δήλωση του λογικού τέλους ενός προγράμματος BASIC.

ΠΕΡΙΓΡΑΦΗ : Με την εντολή END δηλώνεται το λογικό τέλος της ροής ενός προγράμματος. Αν το λογικό τέλος της ροής του προγράμματος ταυτίζεται με το φυσικό του τέλος, δηλαδή το πρόγραμμα πρόκειται να τελειώσει μετά την τελευταία του γραμμή, τότε η εντολή END είναι προαιρετική.

3.6 FOR....TO....STEP...

ΔΟΜΗ : FOR<δείκτης>=<αρχή>TO<τέλος>STEP<μεταβλητή>

ΣΚΟΠΟΣ : Επαναλαμβανόμενη (LOOP) εκτέλεση ομάδας εντολών ή γραμμών προγράμματος, όσες το πολύ φορές δείχνει ο δείκτης.

ΠΕΡΙΓΡΑΦΗ : Είναι μία σύνθετη εντολή που δίνει την εχέρεια στον προγραμματιστή να επαναλάβει την εκτέλεση ενός συνόλου εντολών - γραμμών προγράμματος, που περικλείονται μεταξύ των εντολών FOR - NEXT τόσες φορές, όσες η μέγιστη τιμή του δείκτη επαναλήψεων (μερτρητής)

Ο μετρητής αυτός είναι κάποια αριθμητική μεταβλητή, που τα όρια εντός των οποίων θα κινηθεί καθορίζονται μέσα στην εντολή FOR.

ΠΑΡΑΔΕΙΓΜΑ

FOR I=1 TO 50
σημαίνει ότι τον ρόλο του μετρητή παίξει η μεταβλητή I με αρχική τιμή 1 και τελική τιμή 50.

Εάν το "βήμα" μεταβολής του δείκτη είναι διάφορο του 1, τότε είναι απαραίτητο να καθορισθεί με την εντολή STEP, ενώ άν είναι 1 καθορίζεται απαραιτήτως.

Οταν το πρόγραμμα, κατά την εκτέλεση του φθάσει στην εντολή NEXT I, τότε ο μετρητής θα μεταβληθεί σύμφωνα με το βήμα πού ορίστηκε (έμμεσα χωρίς την εντολή STEP και θα είναι 1 ή άμεσα με την εντολή STEP)

Τότε η ομάδα των εντολών που βρίσκεται μέσα στο LOOP θα ξαναεκτελεσθεί λαμβάνοντας υπ΄όψιν την νέα τιμή του δείκτη κ.ο.κ.

Το πρόγραμμα θα βγεί από τις συνεχείς αυτές επαναλήψεις, όταν ο μετρητής ξεπεράσει το τελευταίο όριό του.

Τα όρια μέσα στα οποία κινείται ο μετρητής μπορεί να είναι θετικά ή αρνητικά. Για να εκτελεσθεί το LOOP πρέπει το πρώτο όριο να είναι μεγαλύτερο από το δεύτερο άν το βήμα είναι αρνητικό, ενώ το πρώτο όριο να είναι μικρότερο από το δεύτερο άν το βήμα είναι θετικό.

Πρέπει να διευκρινισθεί ότι οι έννοιες "μεγαλύτερο" και "μικρότερο" αναφέρονται στις σχετικές τιμές των ορίων και όχι στις απόλυτες τιμές και γίνεται καλύτερα με τα παραδείγματα:

α) FOR I=2 TO 10 STEP 2
   (2<10, βήμα θετικό)
β) FOR I=-10 TO 0 STEP+ 2
   (-10<0, βήμα θετικό)
γ) FOR I=-1 TO -10 STEP -1
   (-1>-10, βήμα αρνητικό)
δ) FOR I=-10 TO -1 STEP 1
   (-10<-1, βήμα θετικό)

Τα όρια και το βήμα μπορεί να είναι πραγματικοί αριθμοί, αρκεί να ισχύει η σχέση
(<δεύτερο όριο> - <πρώτο όριο>) /<βήμα>=
πολλαπλάσιο <βήμα>+υπόλοιπο

Γενικά τα όρια μεταβολής του δείκτη (μετρητή) και το βήμα μεταβολής μπορούν να δίνονται και παραμετρικά στο πρόγραμμα, με την βοήθεια άλλων μεταβλητών ή και μαθηματικών σχέσεων κ.ο.κ.

ΠΑΡΑΔΕΙΓΜΑ

```
FOR M=(A*B)+(C*VAL(D$)TO(A*B)-(C*VAL(D$))
STEP -SIN(15)
```

ΠΑΡΑΔΕΙΓΜΑΤΑ

```
Α)  10FOR I=1 TO 5:INPUT A$(1):NEXT1
```

```
B)  12FOR R=1 TO TB=INPUT 4, A$,B$
    15PRINT A$" "B$
    20IF LEFT$(A$,5)="    "THEN R=TB
    25 NEXT R
```

3.6.1 Εμφωλιασμένα (NESTED) LOOPS

Σε πιό σύνθετη μορφή προγραμμάτων, είναι δυνατόν να έχουμε επαναλήψεις εντός επαναλήψεων, χρησιμοποιώντας την τεχνική των "εμφιαλωμένων", LOOPS (NESTED LOOPS).

Αυτό σημαίνει ότι σε μία LOOP FOR-NEXT υπάρχει μία ή περισσότερες LOOPS, FOR-NEXT, που αναγκαστικά όμως έχουν διαφορετικές μεταβλητές σαν μετρητές.
Στην περίπτωση αυτή επειδή θα υπάρχουν περισσότερα του ενός NEXT, θα πρέπει υποχρεωτικά στο καθένα να γράφουμε και τον μετρητή του προς την αποφυγή σύγχυσης.

Στην φάση αυτή, ο προγραμματιστής πρέπει να είναι προσεκτικός ώστε η αλληλουχία των NEXT να είναι η αντίστροφη των αντιστοίχων FOR, <μεταβλητή> σε τρόπο που να έχουμε την δομή του "κρεμμυδιού", όπου κάθε LOOP περικλείεται από ένα άλλο κ.ο.κ

ΠΑΡΑΔΕΙΓΜΑ

```
10 FOR M=5 TO 34
15 .............
   .............
40 FOR K=5 TOL STEP 4.6
45 .............
50 NEXT K
55 ...............
60 NEXT M
```

3.7 GOSUB.../RETURN

ΔΟΜΗ : GOSUB αριθμός γραμμής προγράμματος

ΣΚΟΠΟΣ : Κλίση και εκτέλεση μιάς ομάδας εντολών - γραμμών προγράμματος και επιστροφή στο σημείο κλίσης.

ΠΕΡΙΓΡΑΦΗ : Πολλές φορές, σε ένα πρόγραμμα παρουσιάζεται η ανάγκη μίας ομάδας εντολών να πρέπει να εκτελεσθεί η ίδια ακριβώς σε διάφορα σημεία του προγράμματος. Τότε για να αποφύγουμε το γράψιμο των εντολών αυτών πολλές φορές, που σημαίνει σπατάλη χώρου μνήμης και χρόνου για γράψιμο, χρησιμοποιούμε την έννοια του υποπρογράμματος.

ΥΠΟΠΡΟΓΡΑΜΜΑ

Είναι ένα μέρος του προγράμματος, που τελειώνει υποχρεωτικά με την εντολή RETURN και το οποίο μπορούμε να καλέσουμε για εκτέλεση από οποιοδήποτε άλλο σημείο, χρησιμοποιώντας την εντολή GOSUB.

Οταν το υποπρόγραμμα εκτελείται και συναντήσει την εντολή RETURN, αυτόματα επιστρέφει στην αμέσως επόμενη εντολή από την GOSUB που το κάλεσε και συνεχίζεται η εκτέλεση από εκεί και κάτω. Συνεπώς η εντολή RETURN αποτελεί το σημείο εξόδου του υποπρογράμματος.

Ενα υποπρόγραμμα καλείται με την εντολή GOSUB, γράφοντας δεξιά της τον αριθμό γραμμής, που αρχίζει το υποπρόγραμμα. Ενα υποπρόγραμμα μπορεί να καλέσει κάποιο άλλο υποπρόγραμμα, και αυτό κάποιο άλλο κ.ο.κ

3.8 IF.... THEN

ΔΟΜΗ : IF<συνθήκη> THEN<δήλωση>

ΣΚΟΠΟΣ : Εκτέλεση δηλώσεων στην περίπτωση που μεταβλητή ή συνδιασμός μεταβλητών ικανοποιούν κάποια σχέση.

ΠΕΡΙΓΡΑΦΗ : Πολλές φορές, παρουσιάζεται η αναγκαιότητα να ελέγξουμε, μέσα σε ένα πρόγραμμα αν μία σχέση μεταβλητών ικανοποιείται. Τότε γράφουμε την εντολή IF και αμέσως μετά τη σχέση. Στη συνέχεια ακολουθεί η εντολή THEN και διάφορες δηλώσεις (εντολές) που θα εκτελεσθούν τότε και μόνο τότε, αν η προηγούμενη σχέση ικανοποιείται.

Η συνθήκη μπορεί να είναι μία απλή σχέση μεταβλητών ή περισσότερες σχέσεις που συνδέονται μεταξύ τους με BOOLEAN πράξεις.

Οι δυνατές μορφές σχέσης είναι:

=	ισότης
>	μεγαλύτερο
> =	μεγαλύτερο ή ίσο
<	μικρότερο
< =	μικρότερο ή ίσο
< >	διάφορο

Οι δυνατές BOOLEAN πράξεις είναι :

AND	λογικό "και"
OR	λογικό "ή"
NOT	λογική "άρνηση"

Λεπτομέρειες για τις σχέσεις και BOOLEAN πράξεις, δίδονται στις παραγράφους 1.6.2 και 1.6.3. για τελεστές σχέσεων και λογικούς τελεστές αντίστοιχα.

ΠΑΡΑΔΕΙΓΜΑΤΑ

```
α) 10 INPUT K:IF K=99 THEN 60
   30 PRINT K
   60 END

β) 10 INPUT K:IF K=99 OR K<0 THEN 60
   30 PRINT K
   60 END
```

3.9 I N D E X

ΔΟΜΗ : α) INDEX << αριθμός καναλιού >>=<τιμή>
β) <τιμή>=INDEX < (αριθμός καναλιού) >

ΣΚΟΠΟΣ : Ελέγχει τη διεύθυνση επέμβασης της SUPER BASIC σε αρχείο δίσκου ή στο MEMORY DEVICE (μνήμη).

ΠΕΡΙΓΡΑΦΗ : Η εντολή INDEX χρησιμοποιείται γιά να ελέγχουμε την εσωτερική διεύθυνση ενός αρχείου δίσκου, ή τη διεύθυνση του MEMORY DEVICE, που θα επέμβει η BASIC γιά να διαβάσει ή να γράψει πληροφορίες.

Ουσιαστικά ο INDEX είναι ένας λογικός δείκτης που λαμβάνει διάφορες τιμές. Επειδή είναι 8 και τα κανάλια επικοινωνίας με το MEMORY DEVICE είναι 2 έπεται ότι υπάρχουν 10 κανάλια επικοινωνίας, άρα υπάρχουν και 10 διαφορετικά INDEX.

ΚΑΤΑΛΟΓΟΣ INDEX

```
INDEX 1    ΚΑΝΑΛΙ 1
INDEX 2    ΚΑΝΑΛΙ 2
-------------------  )   αρχείο δίσκου
INDEX 8    ΚΑΝΑΛΙ 8
INDEX 9    INPUT  από MEMORY DEVICE ) MEMORY
INDEX 10   OUTPUT στο MEMORY DEVICE ) DEVICE
```

3.9.1. Γενικές αρχές INDEX

1. Οταν θέλουμε να διαβάσουμε τη τιμή που έχει κάποια στιγμή ένας INDEX, τότε γράφουμε τον αριθμό του INDEX μέσα σε παρενθέσεις.

ΠΑΡΑΔΕΙΓΜΑ

```
PRINT   INDEX(3)
PRINT#5, INDEX(6)
PRINT%4, INDEX (10)
```

Αν θέλουμε να φυλάξουμε τη τιμή ενός INDEX σε μία αριθμητική μεταβλητή, γράφουμε:

```
K=INDEX(*)
```

*=αριθμός INDEX δηλαδή καναλιού

2. Οταν θέλουμε να δώσουμε μία νέα τιμή σε κάποιον INDEX, τότε γράφουμε τον αριθμό του INDEX(καναλιού που αναφέρεται ο INDEX) μέσα σε αγκύλες < > και ακολουθεί το σύμβολο της ισότητας =

ΠΑΡΑΔΕΙΓΜΑ

```
INDEX<1>=563
```

3. Τη τιμή ενός INDEX μπορούμε να τη δώσουμε και παραμετρικά, με μία αριθμητική έκφραση αριθμητικών μεταβλητών.

ΠΑΡΑΔΕΙΓΜΑ

```
α) INDEX<5>=K
β) INDEX<8>=K*23-( 25+INDEX(8)-INDEX(10)
```

3.10 INPUT ΑΠΟ ΟΘΟΝΗ

ΔΟΜΗ : INPUT<"μήνυμα">;<μεταβλητή-1>,....,
<μεταβλητή - η>

ΣΚΟΠΟΣ : Εισαγωγή πληροφορίας ή πληροφοριών από το πληκτολόγιο σε πρόγραμμα που εκτελείται.

ΠΕΡΙΓΡΑΦΗ : Οταν πρόκειται να δοθεί στον υπολογιστή μία πληροφορία από το πληκτρολόγιο, στην οθόνη εμφανίζεται ένα λατινικό ερωτηματικό (?) και το πρόγραμμα σταματάει την εκτέλεσή του, μέχρι να πληκτρολογήσουμε την πληροφορία που θα ολοκληρωθεί με το πάτημα του πλήκτρου <RETURN>, ή του <ENTER>.

Πολλές φορές όμως ο χειριστήςδεν θυμάται πιά ακριβώς πληροφορία του ζητάει το πρόγραμμα εκείνη τη στιγμή, π.χ. ημερομηνία, όνομα, τιμή κ.λ.π.

Για να αποφεύγεται τέτοιου είδους σύγχυση, μπορούμε στην εντολή INPUT να ενσωματώσουμε κάποιο μήνυμα μέσα σε εισαγωγικά, αμέσως μετά την εντολή INPUT και πριν το όνομα της μεταβλητής. Στην περίπτωση αυτή το όνομα της μεταβλητής ξεχωρίζει από το μήνυμα με το σύμβολο του ελληνικού ερωτηματικού (;)

ΠΑΡΑΔΕΙΓΜΑ

```
INPUT "ΔΩΣΕ ΟΝΟΜΑ ΠΕΛΑΤΗ" ; NA$
```

Τη στιγμή που η BASIC θα συναντήσει την εντολή αυτή, θα εμφανίσει στην οθόνη το μήνυμα

ΔΩΣΕ ΟΝΟΜΑ ΠΕΛΑΤΗ?

Τη στιγμή αυτή ο CURSOR θ΄αναβοσβήσει δίπλα στο ερωτηματικό, περιμένοντας να πληκτρολογήσουμε κάποια πληροφορία, που θα τη θεωρήσει σαν κάποιο όνομα και θα τη καταχωρήσει στη μεταβλητή NA$.

3.11 INPUT ΑΠΟ MEMORY DEVICE

ΔΟΜΗ : INPUT#4,<όνομα μεταβλητής-1>,...,
<όνομα μεταβλητής -η>

ΣΚΟΠΟΣ : Εισαγωγή πληροφορίας ή πληροφοριών από το MEMORY DEVICE σε πρόγραμμα που εκτελείται.

ΠΕΡΙΓΡΑΦΗ : Σ΄ένα σύνθετο πρόγραμμα που η ταχύτητα εκτέλεσής του μας ενδιαφέρει άμεσα, προσφέρεται η εισαγωγή πληροφοριών από το MEMORY DEVICE, παρά από κάποιο αρχείο δίσκου ή το πληκτρολόγιο.

Προς τον σκοπό αυτό, στην αρχή του προγράμματος καθορίζουμε την περιοχή (επιτρεπτή) της μνήμης σαν MEMORY DEVICE και περνάμε τις πληροφορίες που θα χρησιμοποιήσουμε στην περιοχή αυτή.

Με την εντολή INPUT#4,<όνομα μεταβλητής> ή BASIC αντιλαμβάνεται ότι θα πάρει κάποια πληροφορία από το MEMORY DEVICE και μάλιστα από τη θέση που δείχνει ο INDEX 9.

Συνεπώς η BASIC θα περνά χαρακτήρες στη μεταβλητή, που δώσαμε τ΄όνομά της με την εντολή INPUT#4, με σειριακό τρόπο, αρχίζοντας από εκεί πού δείχνει ο INDEX 9, μέχρι να συναντήσει τον χαρακτήρα $0D στο δεκαεξαδικό σύστημα, που αντιστοιχεί στο GR(=CARRIAGE RETURN) και φανερώνει το τέλος της πληροφορίας.

Μετά την διαδικασία αυτή, ο INDEX 9 θα δείχνει την αρχή της επόμενης πληροφορίας στο MEMORY DEVICE.

ΠΑΡΑΔΕΙΓΜΑ

```
10 POKE 133, INT(34000/256):RUN 15
15 INDEX<10>=34000:PRINT#4, "ΝΙΚΟΣ"
20 INDEX<9>=34000
30 INPUT#4,A$:PRINT A$
40 END
```

Με την εντολή 10 καθορίζουμε το MEMORY DEVICE τη περιοχή 34000_{10}-49150_{10} διευθύνσεις.

Με την εντολή 15 γράφουμε στο MEMORY DEVICE, αρχίζοντας από τη διεύθυνση 3400 τη λέξη-πληροφορία NIKOS.

Με την εντολή 20 θέτουμε το INDEX 9 στην αρχή της πληροφορίας.

Με την εντολή 30 διαβάζουμε την πληροφορία και την τυπώνουμε στην οθόνη.

3.12. INPUT ΑΠΟ ΑΡΧΕΙΟ - ΔΙΣΚΟΥ

ΔΟΜΗ : INPUT% <αριθμός καναλιού επικοινωνίας>, <πληροφορία-1>, ... <πληροφορία-η>

ΣΚΟΠΟΣ : Εισαγωγή πληροφοριών από κάποιο αρχείο δίσκου, μέσω κάποιου καναλιού επικοινωνίας.

ΠΕΡΙΓΡΑΦΗ : Οταν θέλουμε να διαβάσουμε κάποια πληροφορία από ένα αρχείο δίσκου, που το έχουμε "ανοίξει" από πριν, και άρα έχουμε καθορίσει το κανάλι επικοινωνίας (βλέπε την εντολή OPEN) χρησιμοποιούμε την εντολή INPUT% δίδοντας το κανάλι επικοινωνίας και το όνομα ή τα ονόματα των μεταβλητών τα οποία θα περάσουν οι πληροφορίες.

Πριν να χρησιμοποιήσουμε την εντολή INPUT%, πρέπει ο δείκτης του καναλιού να δείχνει την αρχή της πρώτης πληροφορίας που θα διαβαστεί.

ΠΑΡΑΔΕΙΓΜΑ

```
10 OPEN "PELAT1", 1:INTEX<1>=20
15 INPUT%1,A$,B$
20 PRINT A$" "B$
25 END
```

3.13. NEXT

ΔΟΜΗ : NEXT ή NEXT<δείκτης>

ΣΚΟΠΟΣ : Φανερώνει το τέλος κάποιου LOOP

ΠΕΡΙΓΡΑΦΗ : Αρχίζοντας κάποιο LOOP με την εντολή FOR πρέπει σε κάποιο σημείο του προγράμματος να τελειώσει, και το τέλος του LOOP το φανερώνει η εντολή NEXT. Μόλις το πρόγραμμα εκτελέσει μία επανάληψη ενός LOOP και συναντήσει την εντολή NEXT, η BASIC ενημερώνει τον δείκτη του LOOP σύμφωνα με το βήμα του, και ανάλογα επαναλαμβάνει τις εντολές που περικλείονται στο LOOP, ή βλέποντας ότι έγιναν όλες οι απαιτούμενες επαναλήψεις, συνεχίζει την εκτέλεση του προγράμματος από την επόμενη εντολή του NEXT και κάτω.

Εαν έχουμε έν απλό LOOP, τότε αυτό κλείνει με την εντολή NEXT χωρίς δείκτη, ενώ εαν έχουμε περίπτωση NESTED LOOPS, τότε την εντολή NEXT ακολουθεί ο δείκτης του LOOP που κλείνει.

ΠΑΡΑΔΕΙΓΜΑ

α)
```
FOR I=1 TO 10:PRINT I:NEXT
```

β)
```
10 FOR I=1 TO 10
20 FOR J=1 TO 1:PRINT I,J:NEXT J
30 NEXT I
```

3.14. ON ... GOTO .../ON ...GOSUB...

ΔΟΜΗ : ON<δείκτης>GOTO ή GOSUB<αριθμός γραμμής-1>, ...<αριθμός γραμμής-η>

ΣΚΟΠΟΣ : Αλλαγή της ροής εκτέλεσης του προγράμματος, σύμφωνα με την τιμή που έχει εκείνη τη στιγμή ο δείκτης.

ΠΕΡΙΓΡΑΦΗ : Μετά το ON ακολουθεί μία αριθμητική μεταβλητή ή μία έκφραση αριθμητικών μεταβλητών που το αποτέλεσμά της να είναι κάποιος ακέραιος αριθμός.

Την τιμή της μεταβλητής αυτής εξετάζουμε, συγκρίνοντάς την με τους φυσικούς αριθμούς 1, 2, 3 ...

Δεξιά από το GOTO ακολουθεί μία σειρά αριθμών γραμμών προγράμματος, που ξεχωρίζουν μεταξύ τους με κόμματα, και το πλήθος των αριθμών αυτών ισούται με τον αριθμό συγκρίσεων του δείκτη.

Κατά την εκτέλεση της εντολής, ON...GOTO... αν η τιμή του δείκτη βρεθεί ίση με κάποιον φυσικό αριθμό, τότε το πρόγραμμα συνεχίζει να εκτελείται από την γραμμή εκείνη, της οποίας ο αριθμός έχει σειρά στη διάταξη των αριθμών μετά το GOTO, ίση με τη τιμή του δείκτη

ΠΑΡΑΔΕΙΓΜΑ

```
                (1) (2) (3) (4) (5)
ON K GOTO  300,100,500,400,200
```

Αν ο K έχει την τιμή 2, το πρόγραμμα θα συνεχίσει την εκτέλεσή του από τη γραμμή με αριθμό 100.

Αν ο K έχει την τιμή 3, το πρόγραμμα θα συνεχίσει την εκτέλεσή του από τη γραμμή με αριθμό 500 κ.ο.κ.

Σε περίπτωση που ο δείκτης έχει τιμή πραγματικού αριθμού με δεκαδικό μέρος π.χ. 3.5 ή έχει μεν τιμή φυσικού αριθμού, αλλά ο φυσικός αριθμός είναι μεγαλύτερος από τον πληθικό αριθμό, των αριθμών εντολών που ακολουθούν από το GOTO, τότε το πρόγραμμα θα συνεχίσει την εκτέλεσή του από την αμέσως επόμενη εντολή

ΠΑΡΑΔΕΙΓΜΑ

```
15 ON K*A GOTO 10,100,100,25
```

Σε περίπτωση που έχουμε τη διάταξη

ON... GOSUB...

ανάλογα με την τιμή του δείκτη, το πρόγραμμα θα εκτελέσει το αντίστοιχο υποπρόγραμμα (SUBROUTINE) και κατόπιν θα συνεχίσει την εκτέλεσή του από την επόμενη εντολή της ON...GOSUB...

3.14.1. GOTO

ΔΟΜΗ : GOTO <αριθμός εντολής>

ΣΚΟΠΟΣ : Αλλαγή της ροής εκτέλεσης του προγράμματος χωρίς συνθήκη.

ΠΕΡΙΓΡΑΦΗ : Οταν εκτελείται ένα πρόγραμμα BASIC, ακολουθώντας τη πορεία που καθορίζεται από τον αριθμό γραμμών του, παρουσιάζεται μερικές φορές η ανάγκη να μεταβάλλουμε τη πορεία αυτή.

Για το σκοπό αυτό χρησιμοποιούμε την εντολή GOTO, γράφοντας δεξιά της τον αριθμό γραμμής, που θέλουμε να συνεχιστεί η εκτέλεση του προγράμματος.

Τον αριθμό γραμμής μπορούμε να τον δώσουμε απ΄ευθείας με νούμερο ή παραμετρικά με μία αριθμητική μεταβλητή ή κάποια μαθηματική έκφραση αριθμητικών μεταβλητών.

ΠΑΡΑΔΕΙΓΜΑ

α) 100 GOTO 25

β) 20 GOTO INT (A*B+VAL(C$)/A)

Σε περίπτωση που ο αριθμός γραμμής που καθορίζεται στην εντολή GOTO... δεν αντιστοιχεί σε καμμία γραμμή του προγράμματος, τότε έχουμε ένδειξη λάθους.

3.15. OPEN

ΔΟΜΗ : OPEN <"όνομα αρχείου">,<αριθμός καναλιού>

ΣΚΟΠΟΣ : Ανοίγει ένα DATA (μόνο) αρχείο για την επικοινωνία του με την BASIC.

ΠΕΡΙΓΡΑΦΗ : Αν θέλουμε να επικοινωνήσουμε μ΄ένα DATA αρχείο, ώστε στη συνέχεια να γράφουμε ή να διαβάζουμε πληροφορίες (DATA), πρέπει αρχικά να το ανοίξουμε, χρησιμοποιόντας την εντολή OPEN.

Τα κανάλια επικοινωνίας της BASIC με τα αρχεία δίσκου είναι 8. Έτσι ένα αρχείο μπορούμε να το ανοίξουμε σ΄ένα από τα 8 κανάλια επικοινωνίας, αλλά σε κανάλι ελεύθερο, δηλαδή σε κανάλι που δεν είναι κατειλημμένο από κάποιο άλλο ανοικτό αρχείο.

Το κάθε κανάλι επικοινωνίας έχει τον δικό του δείκτη INTEX, οπότε ανοίγοντας ένα αρχείο σε κάποιο κανάλι, αυτομάτως ενεργοποιείται ο INTEX του καναλιού, και μάλιστα μόλις ανοίξει το αρχείο, ο INTEX του καναλιού παίρνει την τιμή μηδέν (0), ώστε να δείχνει το πρώτο χαρακτήρα του αρχείου.

ΠΑΡΑΔΕΙΓΜΑ

```
10 OPEN "PELAT", 1
```

ανοίγουμε το DATA PELAT στο κανάλι 1

3.16. PEEK

ΔΟΜΗ : PEEK <διεύθυνση μνήμης>

ΣΚΟΠΟΣ : Ανάγνωση περιεχόμενου μιάς διεύθυνσης μνήμης.

ΠΕΡΙΓΡΑΦΗ : Η εντολή PEEK μας επιτρέπει να διαβάσουμε το περιεχομενο κάποιας διεύθυνσης μνήμης. Η διεύθυνση της μνήμης δίδεται με αριθμό του δεκαδικού συστήματος, που περικλείεται σε παρενθέσεις ( ).

ΠΑΡΑΔΕΙΓΜΑ

```
PRINT PEEK (35000)
```

Θα τυπώσει στήν οθόνη το περιεχόμενο της μνήμης που βρίσκεται στη διεύθυνση 35000_{10}

3.17. POKE

ΔΟΜΗ : POKE <διεύθυνση μνήμης>,<αριθμός>

ΣΚΟΠΟΣ : Γράφει έναν δεκαδικό (ως προς το σύστημα αρίθμησης) χαρακτήρα σε μία συγκεκριμμένη διεύθυνση της μνήμης.

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή χρησιμοποιείται γιά τη δραστική επέμβαση σε μία συγκεκριμμένη διεύθυνση της μνήμης, ώστε να γράψουμε έναν αριθμό.

Επειδή το μήκος της λέξης μνήμης (BYTE) είναι 8 BITS, ο αριθμός που θα γράψουμε είναι 0 (00 HEX) μέχρι 255 (FF HEX).

Περισσότερες πληροφορίες γιά την εντολή POKE δίδονται στη COMMAND POKE στο κεφάλαιο 2.

ΠΑΡΑΔΕΙΓΜΑ

```
10 POKE 133, INT (34000/256):RUN 30
20 ---------------------------------
```

Στο παράδειγμα αυτό, καθορίζεται η περιοχή μνήμης από 34000_{10} εως 49150_{10} σαν MEMORY DEVICE.

3.18. PRINT

ΓΕΝΙΚΑ : Με την εντολή **PRINT** μπορούμε να διοχετεύουμε πληροφορίες από το χώρο της BASIC στον έξω κόσμο που είναι η οθόνη, ο εκτυπωτής, τα αρχεία δεδομένων και το MEMOPY DEVICE.

Απο την περιγραφή του έξω κόσμου της BASIC, είναι φανερό, ότι η εντολή PRINT, εκτελείται κατά διαγορετικό τρόπο, όσον αφορά τα περιφερειακό δηλαδή το μέσον που θα διοχετεύσει τη πληροφορία.

Ετσι ορόζουμε ειδικά σύμβολα, που ανταποκρίνονται σε συγκεκριμμένα κομμάτια του έξω κόσμου της BASIC, των περιφερειακών δηλαδή.

ΠΙΝΑΚΑΣ ΠΕΡΙΦΕΡΕΙΑΚΩΝ ΚΑΙ ΣΥΜΒΟΛΑ ΤΟΥΣ

Οθόνη	:	0
	:	1 (ή τίποτα)
MEMORY DEVICE	:	4
Εκτυπωτής παράλληλος	:	5
Εκτυπωτής σειριακός	:	6
Κανάλι δίσκου	:	%N (όπου N=1, ...,8)

Αν κατά την εντολή PRINT ακολουθεί ένα από τα παραπάνω σύμβολα, τότε η έξοδος της πληροφορίας θα "περάσει" στο περιφερειακό, που ορίζεται από το σύμβολο αυτό.

Αν δεν υπάρχει κανένα σύμβολο περιφερειακό, τότε ο υπολογιστής δέχεται σαν περιφερειακό την τερματική οθόνη.

ΠΡΟΣΟΧΗ : Επειδή η εντολή PRINT, είναι από τις πιό χρησιμοποιούμενες εντολές στην BASIC, η SURER BASIC την αναγνωρίζει αν την συμβολίσουμε και με το λατινικό ερωτηματικό (?) αντί να γράψουμε PRINT.

π.χ. PRINT "NIKOS" ισοδυναμεί με την ?"NIKOS"

3.18.1. PRINT σε οθόνη

ΔΟΜΗ : PRINT <πληροφορία> ή ?<πληροφορία>

ΣΚΟΠΟΣ : Διοχέτευση μιάς πληροφορίας από το χώρο της BASIC στο τερματικό - οθόνη.

ΠΕΡΙΓΡΑΦΗ : Αν μετά την εντολή PRINT γράψουμε το όνομα κάποιας μεταβλητής, τότε στην οθόνη θα γραφεί το περιεχόμενο της μεταβλητής.

Αν όμως θέλουμε να γράψουμε μία πληροφορία στην οθόνη, χωρίς την χρήση μεταβλητής, τότε αν

α) Η πληροφορία είναι κάποιος αριθμός γράφουμε PRINT και τον αριθμό ή ? και τον αριθμό.

β) Η πληροφορία είναι αλφαριθμητική, γράφουμε PRINT και την πληροφορία μέσα σε αποστρόφους.

ΠΑΡΑΔΕΙΓΜΑΤΑ

α) PRINT 10
β) PRINT "NIKOS"

Με την εντολή PRINT μπορούμε να κάνουμε απ΄ευθείας και πράξεις στις πληροφορίες την στιγμή της εκτύπωσης.

ΠΑΡΑΔΕΙΓΜΑΤΑ

α) PRINT A-B
β) PRINT 15+3*18
γ) PRINT A$+B$+"*"

Γενικά με την εντολή PRINT μπορούμε να τυπώσουμε πολλές πληροφορίες στην ίδια γραμμή οθόνης, παραθέτοντας απλά τις πληροφορίες που θέλω.

Επίσης υπάρχουν ορισμένες συναρτήσεις εισόδου/εξόδου (I/O) που καθορίζουν τη θέση που θα γραφεί μία πληροφορία και είναι:

1. Ερωτηματικό (;)
2. Κόμμα (,)
3. TAB (X)
4. SPC (X)
5. ! XY

Λεπτομέρειες γιά την λειτουργία των συναρτήσεων αυτών δίδονται στο ειδικό κεφάλαιο 6 περί συναρτήσεων I/O.

ΠΑΡΑΔΕΙΓΜΑ

```
PRINT A$B$A;C TAB(12) D$"ΚΑΛΗΜΕΡΑ"
```

3.18.2. PRINT σε εκτυπωτή

ΔΟΜΗ : α) PRINT#5,<πληροφορία-1>...<πληροφορία-η>
γιά παράλληλο εκτυπωτή.

β) PRINT#6,<πληροφορία-1>...<πληροφορία-η>
γιά σειριακό εκτυπωτή.

ΣΚΟΠΟΣ : Διοχετεύει πληροφορίες από τον χώρο της BASIC, σε παράλληλο ή σειριακό εκτυπωτή ανάλογα.

ΠΕΡΙΓΡΑΦΗ : Μπορούμε να διοχετεύσουμε πληροφορίες στο περιφερειακό-εκτυπωτή αν αμέσως μετά την εντολή PRINT γραφεί το ειδικό σύμβολο #5 ή #6 που χαρακτηρίζει τον εκτυπωτή.

Οτι ισχύει γιά την εντολή PRINT στην οθόνη, ισχύει και γιά την εντολή PRINT στον εκτυπωτή, με μόνη διαφορά ότι, το σύνολο των πληροφοριών που θα εκτυπωθούν, ξεχωρίζουν από τον κορμό της εντολή μ΄ένα κόμμα (,).

ΠΑΡΑΔΕΙΓΜΑ

```
α) PRINT#5,10        ή ?#5, 10
β) PRINT#5,"NIKOS"   ή ?#5, "NIKOS"
γ) PRINT#5,A-B       ή ?#5, A-B
δ) PRINT#5,3*10+15   ή ?#5, 3*10+15
ε) PRINT#5,A$ B$     ή ?#5, A$+B$
στ)PRINT#5,A$B$A; C TAB(30)"ΚΑΛΗΜΕΡΑ"
   ή ?#5, A$BSA;C
```

Αντίστοιχα παραδείγματα μπορούν να δοθούν και για τον σειριακό εκτυπωτή με το ειδικό σύμβολο #6.

3.18.3. PRINT σε αρχείο δίσκου

ΔΟΜΗ : PRINT%N,<πληροφορία-1>,...,<πληροφορία-η>
όπου N αριθμός καναλιού επικοινωνίας.

ΣΚΟΠΟΣ : Διοχετεύει πληροφορίες από τον χώρο της BASIC στο τερματικό αρχείο δίσκου.

ΠΕΡΙΓΡΑΦΗ : Στο κεφάλαιο περί αρχείων μπορεί να βρεί όλες τις απαραίτητες πληροφορίες γιά τη δομή τους και τον τρόπο λειτουργίας.

Εδώ απλώς υπενθυμίζουμε, πως υπάρχουν 8 λογικά κανάλια επικοινωνίας της BASIC με αρχεία δίσκου, που αριθμούνται από το 1 μέχρι το 8.

Το σύμβολο επικοινωνίας της BASIC με τα αρχεία, μέσω καναλιών, είναι το % έτσι αν γράψουμε: PRINT %3, A$, σημαίνει: καταχώρησε μέσω του καναλιού 3 την πληροφορία A$, στο αρχείο που προηγούμενα έχουμε "ανοίξει" στο κανάλι 3.

Και εδώ υπάρχει η έννοια του INTEX και μάλιστα, σε κάθε ένα από τα 8 κανάλια υπάρχει και ο αντίστοιχος INTEX, που παίρνει την ονομασία του από τον αριθμό του καναλιού π.χ. INTEX(1), INTEX(5), κ.λ.π.

Εδώ ο INTEX δείχνει την διεύθυνση μέσα στο αντίστοιχο αρχείο, που θα καταχωρηθεί μία πληροφορία. Και στο τέλος κάθε πληροφορίας η BASIC καταχωρεί αυτόματα το σύμβολο CR, γιά να τις ξεχωρίζει μεταξύ τους, αργότερα στην ανάγνωση.

3.18.4. PRINT σε μνήμη

ΔΟΜΗ : PRINT#4, <πληροφορία>

ΣΚΟΠΟΣ : Διοχέτευση πληροφοριών στο MEMORY DEVICE.

ΠΕΡΙΓΡΑΦΗ : Οπως ήδη έχουμε αναφέρει μία περιοχή μνήμης, που είναι εκτός χώρου εργασίας της BASIC, μπορεί να θεωρηθεί σαν ένα περιφερειακό (έξω κόσμος), με το οποίο η BASIC ανταλλάσει πληροφορίες. Το περιφερειακό αυτό ονομάζεται MEMORY DEVICE.

Πληροφορίες από την BASIC μπορούν να τυπωθούν στο περιφερειακό αυτό, αν αμέσως μετά την εντολή PRINT γράψουμε το χαρακτηριστικό σύμβολο #4 (PRINT#4,A$).

Οι λόγοι που τυπώνουμε πληροφορίες σε μία περιοχή μνήμηςεκτός του χώρου της BASIC είναι οι εξής 2:

α) Χρήσιμες πληροφορίες που προκύπτουν από την εκτέλεση ενός προγράμματος, αν μπουν στο MEMORY DEVICE δεν καταστρέφονται και έτσι μπορούμε να τις διαβιβάσουμε σε ένα πρόγραμμα.

β) Πληροφορίες που προκύπτουν από την εκτέλεση ενός προγράμματος μπορούν να συσσωρευτούν σε περιοχή μνήμης, εκτός των συνηθισμένων καναλιών επικοινωνίας με δίσκους, και στη συνέχεια να διαβιβαστούν σε αρχεία σαν ολότητα, κερδίζοντας σε ευελιξία και ταχύτητα.

Για να μπορούμε να ελέγχουμε τη διεύθυνση μνήμης όπου η εντολή PRINT#4,A$ θα καταχωρήσει την πληροφορία π.χ. A$, υπάρχει ο "λογικός" δείκτης, ο INTEX (10), στον οποίο μπορούμε να δίνουμε διάφορες τιμές π.χ. INTEX<10>=42000 (βλέπε την εντολή INTEX).

Η εντολή PRINT#4 θα καταχωρήσει τις πληροφορίες στη διεύθυνση μνήμης, που δείχνει κάθε στιγμή ο INTEX 10.

Η τιμή του INTEX μεταβάλλεται συνέχεια, αυτόματα, με κάθε χαρακτήρα που καταχωρείται, ώστε πάντα να δείχνει σωστά την επόμενη θέση, που μπορεί να καταχωρηθεί ένας χαρακτήρας.

ΠΑΡΑΔΕΙΓΜΑ

```
        40000
---------------------(μνήμη) ( -=1 BYTE)
     INTEX(10)
```

Εστω ότι έχουμε δώσει στον INTEX 10 τη τιμή 40000 και άρα στο σχήμα δείχνει στο τελευταίο αστεράκι.

Αν τώρα γράψουμε την εντολή:

PRINT#4,"NIKOS"

τότε κάπου ενδιάμεσα όταν έχει καταχωρηθεί και το γράμμα Κ τα πράγματα θα είναι:

```
--------------NIK----(μνήμη)
     INTEX (10)       (-=1 BYTE)
```

δηλ. ο INTEX προχωράει και τώρα δείχνει στη θέση 40003 που θα γραφτεί σε λίγο το γράμμα Ο, κ.ο.κ.

Στο τέλος, όταν καταχωρηθεί όλη η πληροφορία, η κατάσταση είναι:

```
                D40006
----------NIKOS CR---(μνήμη)
     INTEX (10)       (-=1 BYTE)
```

Παρατηρούμε ότι στο τέλος της πληροφορίας η BASIC καταχώρησε και ένα επιπλέον σύμβολο (CR CARRIAGE RETURN 0DH σε HEXADECIMAL), που σκοπό έχει να διαχωρίζει τις πληροφορίες μεταξύ τους ώστε να είναι εύκολη η μετέπειτα ανάγνωσή τους.

Αυτό το σύμβολο είναι ένα BYTE (ένας χαρακτήρας).

Το ίδιο, όπως θα δούμε, συμβαίνει και με την καταχώρηση πληροφοριών σε δίσκους.

Οποιαδήποτε στιγμή μπορούμε να διαβάσουμε την τιμή που έχει ο INTEX του MEMORY DEVICE, αν γράψουμε την εντολή: PRINT INTEX (10), όπου τώρα τον αριθμό 10 τον έχουμε βάλει μέσα σε παρανθέσεις.

3.19. REPL

ΔΟΜΗ : REPL<όνομα μεταβλητής>,<χαρακτ=χαρακτ>

ΣΚΟΠΟΣ : Αντικαθιστά στο περιεχόμενο μιάς αλφαριθμητικής μεταβλητής, χαρακτήρες που επιλέγουμε, με άλλους.

ΠΕΡΙΓΡΑΦΗ : Η εντολή αυτή μας επιτρέπει την αντικατάσταση χαρακτήρων του περιεχομένου μιάς αλφαριθμητικής μεταβλητής με άλλους.

Αν στο περιεχόμενο μιάς αλφαριθμητικής μεταβλητής υπάρχει ένας χαρακτήρας η-φορές, και θέλουμε να τον αντικαταστήσουμε με κάποιον άλλον, τότε με την εντολή REPL θα αντικατασταθούν ταυτόχρονα και οι η-χαρακτήρες.

ΠΑΡΑΔΕΙΓΜΑ

REPL A$, E=I
όπου A$="ΒΕΚΕΝΤΕΟΣ"

Μετά την εκτέλεση της εντολής, έχουμε

A$="ΒΙΚΙΝΤΙΟΣ"

3.20. READ

ΔΟΜΗ : READ<μεταβλητή-1>,...,<μηταβλητή-η>

ΣΚΟΠΟΣ : Περνάει πληροφορίες που δίνονται στο πρόγραμμα με εντολή DATA, σε μεταβλητές.

ΠΕΡΙΓΡΑΦΗ : Οταν σε ένα πρόγραμμα BASIC έχουμε αποθηκεύσει πληροφορίες χρησιμοποιώντας την εντολή DATA, μπορούμε να περάσουμε τις πληροφορίες αυτές σε μεταβλητές χρησιμοποιώντας την εντολή READ.

Η σχέση "ποιά πληροφορία θα περάσει, σε ποιά μεταβλητή", ορίζεται από την σειρά παράθεσης των μεταβλητών στην εντολή READ και τη σειρά παράθεσης των πληροφοριών στην εντολή DATA, υπάρχει δηλαδή μία αμφιμονοσήμαντη αντιστοιχία μεταξύ μεταβλητών και πληροφοριών.

Αν σε ένα πρόγραμμα υπάρχουν πολλές εντολές READ, είναι σαν να υπάρχει μία εντολή READ και οι μεταβλητές έχουν σαν διάταξη, τη διάταξη της πρώτης κατά σειρά READ, ακολουθεί η διάταξη της δεύτερης READ κ.ο.κ.

Οπως λοιπόν γίνεται αντιληπτό, υπάρχει κάποιος εσωτερικός δείκτης DATA που κάθε φορά δείχνει μία πληροφορία DATA, και αυξάνει αυτός ο δείκτης σε κάθε εντολή READ.

Στην αρχή του προγράμματος, μόλις αρχίσει η εκτέλεσή του, ο δείκτης DATA δείχνει πληροφορία της πρώτης εντολής.

Μόλις, κατά την εκτέλεση, συναντήσει την εντολή READ περνάει η BASIC την πληροφορία στην πρώτη μεταβλητή, μετακινεί τον δείκτη DATA στην επόμενη πληροφορία, περνάει στην δεύτερη μεταβλητή την πληροφορία κ.ο.κ.

Θα πρέπει να προσέχουμε στην εντολή READ, ώστε σε μεταβλητες αριθμητικές, να αντιστοιχούν αριθμητικές μεταβλητές, αλφαριθμητικές πληροφορίες.

Εαν με την επαναλαμβανόμενη εκτέλεση εντολών READ εξαντλήσουμε όλες τις πληροφορίες DATA, τότε θα εμφανιστεί λάθος "OUT OF DATA", γι΄αυτό χρησιμοποιείται η εντολή RESTORE που επαναφέρει το δείκτη DATA στην αρχή των DATA.

ΠΑΡΑΔΕΙΓΜΑ

```
α) 10 READ A,L$
    - - - - - -
    - - - - - -
   30 READ M$,A1,B
    - - - - - - -
    - - - - - - -
   70 READ L1$
    - - - - -
    - - - - -
    - - - - -
   500 DATA 10,"NIKOS","ΒΙΚΕΝΤΙΟΣ",5.3,12,"ΓΙΑΝΝΗΣ"
```

Οι εντολές 10, 30 και 70 μπορούν να αντικαταστάθούν από την

```
10 READ A,L$,M$,A1,B,L1$
```

ενώ η εντολή 500 μπορεί να αντικατασταθεί από τις εντολές

```
500 DATA 10
501 DATA "NIKOS"
502 DATA "ΒΙΚΕΝΤΙΟΣ"
- - - - - - - - - - -
505 DATA "ΓΙΑΝΝΗΣ"
```

ή

```
500 DATA 10, "NIKOS"
501 DATA "ΒΙΚΕΝΤΙΟΣ",5.3,12
502 DATA "ΓΙΑΝΝΗΣ"            κ.ο.κ.
```

Οι μεταβλητές παίρνουν τις τιμές:

```
A    =  10
L$   = "NIKOS"
M$   = "ΒΙΚΕΝΤΙΟΣ"
A1   =  5.3
B    =  12
L1$  = "ΓΙΑΝΝΗΣ"
```

β)
```
10FOR I=1 TO 3:READ A$(1):RESTORE:NEXT I
- - - - - - - - - - - - - - - - - - - -
50 DATA "ΒΙΚΕΝΤΙΟΣ"
```

```
τότε A$(1)="ΒΙΚΕΝΤΙΟΣ"
     A$(2)="ΒΙΚΕΝΤΙΟΣ"
     A$(3)="ΒΙΚΕΝΤΙΟΣ"
```

3.21. RESTORE

ΔΟΜΗ : RESTORE

ΣΚΟΠΟΣ : Επαναφέρει τον δείκτη εντολών DATA, ώστε να δείχνει την πρώτη πληροφορία.

ΠΕΡΙΓΡΑΦΗ : Αν χρησιμοποιώντας την εντολή READ/DATA θέλουμε να ξαναρχίσουμε την ανάγνωση πληροφοριών που δίνονται σε εντολές DATA από την αρχή, χρησιμοποιούμε την εντολή RESTORE και ο δείκτης DATA επανέρχεται στην αρχή των DATA.

ΠΑΡΑΔΕΙΓΜΑ

```
10 READ A,B,C
20 PRINT A;B;C
30 RESTORE
40 READ E
50 PRINT E
60 DATA 10,15,20,25
RUN
10 15 20
10
```

3.22. SORT

ΔΟΜΗ : SORT

ΣΚΟΠΟΣ : Κατατάσσει σε αλβαβητική μορφή δεδομένα που βρίσκονται στην μνήμη.

Με την εντολή SORT κατατάσσονται σε αλφαβητική μορφή, αλφαριθμητικά δεδομένα, ή σε αύξουσα διάταξη αριθμητικά δεδομένα.

Τα δεδομένα (πληροφορίες) πρέπει να ευρίσκονται στη κεντρική μνήμη, και μάλιστα σε περιοχή έξω από τον χώρο της BASIC, και να έχουν σταθερό μήκος (RECORD LENGTH).

Για να εκτελεσθεί η εντολή SORT, πρέπει να δηλωθούν από πριν στην BASIC μέσω της εντολής POKE (βλέπε κεφάλαιο-2, COMMAND POKE, 2.10.1.2) τις εξής παραμέτρους:

α. Διεύθυνση μνήμης που αρχίζουν τα δεδομένα

```
POKE  15464, LOW
POKE  15465, HIGH
```

β. Αριθμός χαρακτήρων που λαμβάνουν μέρος στη διαδικασία SORT δηλαδή ως προς πόσους χαρακτήρες από το RECORD θέλουμε το SORTING

POKE 15467, NN

γ. Το μήκος του RECORD

POKE 14468, RL

Για έκδοση της γλώσσας V.3.4 και νεότερες δεν χρειάζεται ο καθορισμός των παραμέτρων

SORT (RAM,NR,AP.XAP,RL)

Η εντολή SORT για την VERSION V.3.4 και νεότερες έχει νέα δομή. Ετσι αντί ο χρήστης να γράφει τις μεταβλητές του με τις διάφορες POKE εντολές, τις δίνει απ' ευθείας με την εντολή σαν παραμέτρους.

Η νέα μορφή της εντολής είναι:

SORT (RAN,NR,N,RL) όπου

RAN είναι η διεύθυνση μνήμης από την οποία θα φορτωθούν οι πληροφορίες για διάταξη

NR είναι ο αριθμός των εγγραφών πληροφοριών που θα διαταχθούν.

N είναι ο πληθικός αριθμός των πρώτων από την αρχή χαρακτήρων κάθε πληροφορίας ως προς τους οποίους θα γίνει η διάταξη

RL είναι το μέγεθος της κάθε πληροφορίας (RECORD LENGTH)

ΠΑΡΑΔΕΙΓΜΑ

SORT (3400,100,5, 42)

Σημαίνει ότι από την 34000_{10}-49150_{10} περιοχή της μνήμης θα φορτωθούν οι πληροφορίες που θέλουμε να διαταχθούν (SORTING), 100 είναι το πλήθος των πληροφοριών που θα διαταχθούν, η διάταξη θα γίνει ως προς τους 5 πρώτους χαρακτήρες της κάθε πληροφορίας και κάθε πληροφορία έχει μήκος 42 BYTES.

Γενικά το SORT αρχείων δίσκου που έχουν φτιαχθεί με το GIGA-DBMS γίνεται μέσω των KEY FILES, χρησιμοποιώντας την UTILITY "SORT"

ΠΑΡΑΔΕΙΓΜΑ

```
 1  POKE 133,132:RUN5
 5  OPEN "PELAT1",1:B1=20
10  POKE 15464,208:POKE 15465,132
20  POKE 15467,50:POKE 15468,30
30  PUTM "PELAT1"[B1],500,3400:INTEX<9>=3400
40  SORT
50  PUTD "PELAT1"[B1],500,3400:CLOSE:END
```

Αποθηκεύουμε τα δεδομένα από το αρχείο PELAT1 στη μνήμη από τη διεύθυνση 34032_{10} και πάνω.

Εχουμε 34000=132*256+208 άρα το LOW είναι 208 και το HIGH 132.

Θέλουμε να σορτάρουμε τα δεδομένα, ως προς τους 5 πρώτους χαρακτήρες και το μήκος του κάθε RECORD δεδομένων είναι 30 χαρακτήρες.

Κάνουμε SORT των δεδομένων και τα επαναφέρουμε στο PELAT1.

3.23. STOP

ΔΟΜΗ : STOP

ΣΚΟΠΟΣ : Σταματάει την εκτέλεση ενός προγράμματος προτού τελειώσει.

ΠΕΡΙΓΡΑΦΗ : Η εντολή STOP σταματάει την εκτέλεση ενός προγράμματος και φέρνει τον Ηλεκτρονικό Υπολογιστή στη θέση MONITOR δηλαδή στην οθόνη υπάρχει η ένδειξη GIGATRONICS.

Συγκεκριμμένα, μόλις συναντήσει το πρόγραμμα την εντολή STOP στην οθόνη εμφανίζεται η ένδειξη

BREAK γρ.NNN (=αριθμός γραμμής προγράμματος)

GIGATRONICS

Αν θέλουμε το πρόγραμμα να συνεχίσει την εκτέλεση του χρησιμοποιούμε την εντολή CONT (βλέπε εντολή CONT στο ίδιο κεφάλαιο)

Η εντολή STOP είναι αντίστοιχη της εντολής CONTROL-C που δίνεται από το πληκτρολόγιο (βλέπε κεφάλαιο-1, παρ.1.4.1.) και χρησιμοποιείται κύρια γιά να σταματήσει την εκτέλεση κάποιου προγράμματος σε συγκεκριμμένη θέση του, όταν υποπτευόμαστε ότι υπάρχει κάποιο πρόβλημα και θέλουμε να διαβάσουμε τις τιμές διαφόρων μεταβλητών του.

ΠΡΟΣΟΧΗ : Αν το πρόγραμμα καθορίζει καί χρησιμοποιεί MEMORY DEVICE με την εντολή POKE 133,X, με την εντολή STOP ή CONTROL-C, η BASIC επανακτά όλη της την μνήμη, δηλαδή καταργεί το MEMORY DEVICE, οπότε υπάρχει κίνδυνος ν΄αλλοιωθούν τα δεδομένα που υπήρχαν πριν το STOP σ΄αυτό.

3.24. WORD

ΔΟΜΗ : WORD <όνομα μεταβλητής>,<αριθμός καναλιού>

ΣΚΟΠΟΣ : Ψάχνει να εντοπίσει σ΄ένα αρχείο δίσκου ή στο MEMORY DEVICE μία σειρά χαρακτήρων.

ΠΕΡΙΓΡΑΦΗ : Αν έχουμε ανοίξει ένα DATA αρχείο και θέλουμε να βρούμε αυτόματα που βρίσκεται μία πληροφορία σ΄αυτό, τότε μηδενίζουμε τον INTEX του καναλιού επικοινωνίας, στο οποίο ανοίξαμε το αρχείο, και χρησιμοποιούμε την εντολή WORD.

Τότε η BASIC θα σαρώσει το αρχείο από την αρχή του, ψάχνοντας να βρει την ζητούμενη πληροφορία. Αν τη βρει, τότε αυτόματα ο δείκτης INTEX θα δείχνει την εσωτερική διεύθυνση του αρχείου, όπου αρχίζει η πληροφορία.

Αν κατά την σάρωση του αρχείου, δεν βρεθεί η πληροφορία, τότε ο INTEX του καναλιού λαμβάνει την τιμή 1.0E9 δηλαδή ένα δισεκατομμύριο, μόλις συναντήσει το λογικό τέλος του αρχείου, που καθορίζεται με τον χαρακτήρα ASCII 255 (FFH), ή το φυσικό τέλος του αρχείου.

Για να κερδίσουμε χρόνο κατά την σάρωση, ή επειδή εξυπηρετεί συγκεκριμμένους σκοπούς σε κάποιο πρόγραμμα, αν ζητάμε κάποια πληροφορία που ξέρουμε ότι θα βρίσκεται από μία καθορισμένη διεύθυνση στο αρχείο και κάτω, τότε δίνουμε στον INTEX σαν τιμή, τη καθορισμένη διεύθυνση και όχι το μηδέν (0) και κατόπιν χρησιμοποιούμε την εντολή WORD.

ΠΡΟΣΟΧΗ : Πριν την χρήση της εντολής WORD πρέπει να καθορισθεί η τιμή του INTEX του καναλιού, στο οποίο αναφέρεται η WORD.

ΠΑΡΑΔΕΙΓΜΑ

```
INTEX<1>=0:WORD A$,1
```

Κατά τον ίδιο τρόπο, αν θέλουμε να εντοπίσουμε μία πληροφορία στο MEMORY DEVICE που καθορίσαμε από πριν ν΄αρχίζει από την διεύθυνση μνήμης 40000 10 τότε γράφουμε

```
INTEX<9>=40000:WORD A$,9
```

Αν φθάσουμε στο φυσικό τέρμα της μνήμης και δεν έχει βρεθεί η ζητούμενη πληροφορία, τότε ο INTEX (9) παίρνει τιμή μεγαλύτερη του 650000.

Το ίδιο θα συμβεί αν συναντήσει το λογικό τέλος του MEMORY DEVICE, που και εδώ είναι ASCII χαρακτήρας 255 (FFH)

ΠΑΡΑΔΕΙΓΜΑ

```
10 INTEX<9>=40000:WORD "1000",9
```

3.25. Συνδιασμοί εντολών για επίτευξη συγκεκριμένων σκοπών

Χρησιμοποιόντας συνδιασμούς εντολών, COMMANDS και συναρτήσεων της SUPER BASIC, μπορούμε να κατασκευάσουμε "νέες εντολές" που εξυπηρετούν συγκεκριμένους σκοπούς σε κάποιο πρόγραμμα.

Συγκριτικά θα δίδονται εντολές από μία STANDARD-BASIC και ακολούθως ο συνδιασμός των εντολών SUPER BASIC που δίνουν σαν αποτέλεσμα την λειτουργία των εντολών αυτών. Επειδή σε αρκετούς τέτοιους συνδιασμούς εντολών θα χρησιμοποιούνται και συναρτήσεις αλφαριθμητικών μεταβλητών, θα πρέπει ο αναγνώστης να μελετήσει πρώτα τις συναρτήσεις αυτές.

3.25.1. WHILE/WEND

Με τις εντολές αυτές δημιουργείται μία λογική LOOP που εκτελείται εφόσον υφίσταται κάποια συνθήκη.

Η αντιστοιχία των εντολών μ΄εντολές SUPER BASIC είναι:

```
100 FOR I=1 TO K
110 IF συνθήκη THEN I=K:GOTO 200
120 - - - - - - - - - - - - - - -
130 - - - - - - - - - - - - - - -
140 - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - -
200 NEXT I
```

3.25.2. PRINT USING

Με την εντολή αυτή εκτυπώνονται πληροφορίες βάσει κάποιου FORMAT.

α) αλφαριθμητικές μεταβλητές
   PRINT LEFT$(A$,η) όπου η=αριθμός χαρακτήρων

β) αριθμητικές μεταβλητές

   1. PRINT![XY]!M όπου X αριθμός ακεραίων ψηφίων
                        Ψ αριθμός δεκαδικών ψηφίων

   2. PRINT LEFT$ (MID$(STR$(μ),2),η) μόνο για μεταβλητές με θετική τιμή, όπου
      M=η μεταβλητή και
      η=αριθμός χαρακτήρων για τύπωμα.

3.25.3. GET

Με την εντολή αυτή μεταφέρεται μία πληροφορία από αρχείο δίσκου σε κάποια μεταβλητή, μέσα στο πρόγραμμα BASIC.

Στην SUPER BASIC εκτελείται ως εξής:

```
10 INTEX (*)=K:INPUT%*, A$  :A=VAL(A$)
```

όπου *=αριθμός καναλιού
K=διεύθυνση του αρχείου, που αρχίζει η πληροφορία
αν η πληροφορία είναι αριθμητική.

3.25.4. LSET και RSET

Βάζουμε στη μνήμη δεδομένα για να γραφούν μετά σε κάποιο αρχείο δίσκου.

Αν το πεδίο στο αρχείο είναι μικρότερο από το μήκος της πληροφορίας, με το LSET το πεδίο γεμίζει με τους χαρακτήρες ASCII που χωράει αρχίζοντας από αριστερά, ενώ το RSET αρχίζει από δεξιά.

Αντίστοιχα στην SUPER BASIC γίνεται

```
10 POKE 133,132:RUN 20
20 INTEX<9>=34000:PRINT#4, LEFT(A$,η)  για το LSET

10 POKE 133,132:RUN 20
20 INTEX<9>=34000:PRINT#4, RIGHT$(A$,η) για το RSET
```

3.25.5. PUT

Η εντολή αυτή παίρνει μία πληροφορία από την μνήμη και την γράφει σ΄ένα RANDOM αρχείο - δίσκου.

Αντίστοιχα στη SUPER BASIC γίνεται:

```
10 INTEX<9>=K:INPUT#4,A$
20 INTEX<*>=L:PRINT$*,A$
```

όπου

L=η διεύθυνση του αρχείου που θ΄αρχίσει η πληροφορία
K=η διεύθυνση της μνήμης που αρχίζει η πληροφορία
*=αριθμός καναλιού επικοινωνίας BASIC και αρχείο-δίσκου

3.25.6. WRITE

Τυπώνει στην οθόνη πληροφορίες

Αντίστοιχο της SUPER BASIC

PRINT#0 ή PRINT

3.25.7. FIELD

Καθορίζει το μήκος των πεδίων σ΄ένα αρχείο δεδομένων.

Αντίστοιχα στην SUPER BASIC γίνεται

A$=LEFT$(A$,η)

3.25.8. EOF

Δείχνει το τέλος του αρχείου δεδομένων

Αντίστοιχα στην SUPER BASIC την λειτουργία αυτή κάνει ο INTEX του καναλιού επικοινωνίας που λαμβάνει τη τιμή 1.0E9(=1 δισεκατομμύριο) όταν βρίσκεται στο τέλος λογικό ή φυσικό DATA αρχείου.

ΚΕΦΑΛΑΙΟ 4

ΣΥΝΑΡΤΗΣΕΙΣ ΓΙΑ ΤΟ I/O (Είσοδος/Έξοδος)

CURSOR : Οτιδήποτε και να γράψουμε στην οθόνη ή τον εκτυπωτή, θα γραφτεί αρχίζοντας εκεί που αναβοσβήνει το χαρακτηριστικό σημάδι στην οθόνη ή εκεί που είναι τοποθετημένη η κεφαλή εκτύπωσης, στον εκτυπωτή. Το σημείο αυτό καλείται CURSOR.

Σε κάθε ένα χαρακτήρα που γράφεται, ο CURSOR μετακινείται μία θέση προς τα δεξιά, υποδεικνύοντας έτσι την νέα θέση που θα γράφει ο επόμενος χαρακτήρας.

Οταν γραφεί στο τερματικό κάποια πληροφορία, ο CURSOR πηγαίνει στην αρχή της επόμενης γραμμής. Αυτό είναι αποτέλεσμα της λειτουργίας των εντολών CARRIAGE RETURN που σημαίνει το τέλος της πληροφορίας (σε γλώσσα μηχανής έχει κωδικό 0DH) και LINE FEED που σημαίνει αλλαγή γραμμής (σε γλώσσα μηχανής έχει κωδικό 0AH).

Υπάρχουν όμως διάφορες συναρτήσεις εκτύπωσης, που ενεργοποιούνται μόνο μετά την εντολή PRINT και μετακινούν τον CURSOR στην θέση που επιθυμούμε και αυτές δίνονται είτε με κάποιο όνομα και παραμέτρους είτε με κάποιο σύμβολο.

4.1. ΚΟΜΜΑ (,)

ΛΕΙΤΟΥΡΓΙΑ: Οταν στην εντολή PRINT, μετά την πληροφορία που θα τυπωθεί, υπάρχει το σύμβολο κόμμα (,), τότε ο CURSOR αφού τυπωθεί η πληροφορία, θα μετακινηθεί 1-4 θέσεις μπροστά, από την αρχή της προηγούμενης πληροφορίας, και περιμένει μέχρι να του δοθεί νέα εντολή PRINT.

4.2. ΕΛΛΗΝΙΚΟ ΕΡΩΤΗΜΑΤΙΚΟ (;)

ΛΕΙΤΟΥΡΓΙΑ: Οταν στην εντολή PRINT, μετά την πληροφορία που θα τυπωθεί, υπάρχει το σύμβολο (;), τότε ο CURSOR παραμένει στην θέση που έχει φθάσει.

4.3. ΣΧΗΜΑ ! [XY]

ΛΕΙΤΟΥΡΓΙΑ: Στην περίπτωση που τυπώνουμε αριθμητικές πληροφορίες, πολλές φορές υπάρχει η ανάγκη να περιέχεται ένας προκαθορισμένος αριθμός δεκαδικών και ακεραίων ψηφίων, με το τελευταίο δεκαδικό στρογγυλοποιημένο και να τυπώνονται στοιχισμένες από δεξιά, ώστε όταν εμφανίζονται αριθμητικές στήλες, τα ψηφία με την ίδια αριθμητική βαρύτητα να καταλαμβάνουν την ίδια στήλη.

Τότε δίνουμε στους αριθμούς το λεγόμενο FORMAT ή σχήμα. Για να δόσουμε λοιπόν ένα σχήμα, χρησιμοποιούμε την συνάρτηση ![XY], όπου το θευμαστικό σημαίνει σχήμα και μέσα στις αγγύλες υπάρχει ο ορισμός του σχήματος με την μορφή ενός διψήφιου αριθμού.

Ο πρώτος αριθμός του αριθμού, που καθορίζει το σχήμα, ορίζει το πλήθος των ακέραιων ψηφίων, ενώ ο δεύτερος αριθμός, το πλήθος των δεκαδικών ψηφίων.

Από την στιγμή που θα δηλωθεί κάποιο σχήμα με την εντολή:

PRINT![XY],(στην αριθμ.των 12 ψηφίων![χ,ψ])

και έπειτα αν ζητηθεί να τυπωθεί κάποια αριθμητική μεταβλητή, και αμέσως μετά την εντολή PRINT και πριν το όνομα της αριθμητικής μεταβλητής δοθεί το σύμβολο θαυμαστικό (!), τότε ο αριθμός θα τυπωθεί με το σχήμα.

ΠΑΡΑΔΕΙΓΜΑ

```
10 PRINT![6,2]!A$!M;R
```

Το σχήμα που ορίζεται είναι το 62 δηλαδή οι αριθμητικές μεταβλητές θα τυπώνονται με 6 ακεραια ψηφία και 2 δεκαδικά (σύνολο μαζί με την τελεία και το πρόσημο 10 χαρακτ.). Για να τυπωθεί όμως κάποιος αριθμός με σχήμα θα πρέπει να έχει πριν το όνομα της μεταβλητής, της οποίας είναι περιεχόμενο, το σύμβολο (!), ειδάλλως θα τυπωθεί χωρίς σχήμα.

Σο παράδειγμα η μεταβλητή M θα τυπωθεί με σχήμα, ενώ η μεταβλητή R θα τυπωθεί χωρίς σχήμα.

4.4. POS X,Y

ΛΕΙΤΟΥΡΓΙΑ: Μετακινεί το CURSOR στην οθόνη στη θέση συντατεγμένες (X,Y), δηλαδή στην γραμμή X και στη στήλη Y.

Δίνεται ότι η τερματική οθόνη έχει 24 γραμμές οριζόντιες που αριθμούνται 0,1,2,...,23 και 80 κάθετες στήλες που αριθμούνται 0,1,2,....,79.

ΠΑΡΑΔΕΙΓΜΑ

Αν θέλουμε ο CURSOR να βρεθεί κατευθείαν ατην γραμμή 17 και την στήλη 56 γράφουμε POS 17,56

4.5. SPC (X)

ΛΕΙΤΟΥΡΓΙΑ: Μετακινεί τον CURSOR από το τέλος κάποιας πληροφορίας που τυπώθηκε, ή αν δεν έχει τυπωθεί πληροφορία από την αρχή της γραμμής, τόσες θέσεις πρός τα δεξιά, όσες του λέει ο αριθμός X, και μετά να τυπωθεί η πληροφορία που ακολουθεί μετά την συνάρτηση SPC.

ΠΑΡΑΔΕΙΓΜΑ

PRINT "NIKOS"SPC(10)"BIKENTIOΣ"

Τότε θα τυπωθεί στην οθόνη

NIKOS 10 κενά BIKENTIOΣ

4.6. TAB (X)

ΛΕΙΤΟΥΡΓΙΑ: Οταν μετά την εντολή PRINT ακολουθεί τη συνάρτηση TAB(X) και φυσικά πριν το όνομα της μεταβλητής που θέλουμε να τυπώσουμε, τότε ο CURSOR θα πάει στην θέση X από το αριστερό άκρο (αρχή της γραμμής που βρισκόμαστε) του τερματικού (οθόνη ή εκτυπωτής) και μετά θα τυπώσει την πληροφορία.

Αν ο CURSOR ήδη βρίσκεται δεξιότερα από την θέση που καθορίζεται το X, τότε επειδή ο CURSOR δεν μπορεί να πάει αριστερά διότι θα έσβηναν οι πληροφορίες που τυπώθηκαν και προυπήρχαν, η συνάρτηση TAB(X) αγνοείται.

ΠΑΡΑΔΕΙΓΜΑ

PRINT TAB(10)A$ TAB(8)B$

Στο παράδειγμα, η A$ θα τυπωθεί από την 11 θέση της γραμμής, ενώ η B$ αμέσως μετά το τέλος της A$.

ΚΕΦΑΛΑΙΟ 5

ΣΥΝΑΡΤΗΣΕΙΣ ΑΛΑΦΑΡΙΘΜΗΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ

Στο κεφάλαιο αυτό αναλύονται όλες εκείνες οι συναρτήσεις που δομούν την SUPER BASIC, και αναφέρονται σε αλφαριθμητικές μεταβλητές.

Μετά την εκτέλεση κάποιας τέτοιας συνάρτησης πάνω σε αλφαριθμητικής μεταβλητή, το αποτέλεσμα είναι πάλι μία αλφαριθμητική μεταβλητή κατάλληλα όμως διαμορφωμένη.

Εξαίρεση αποτελεί, όπως θα δούμε, η συνάρτηση VAL που επιδρά σε αλφαριθμητική μεταβλητή και η STR$ που επιδρά σε αριθμητική μεταβλητή και δίνει αποτέλεσμα αλφαριθμητικό.

5.1. ASC (X$)

Δίνει την ASCII τιμή του πρώτου χαρακτήρα της μεταβλητής X$ στο δεκαδικό σύστημα.

ΠΑΡΑΔΕΙΓΜΑ

X$	ASC(X$)
"ΑΛΕΚΟΣ"	65
"0"	48

5.2. CHR$ (X)

Η συνάρτηση αυτή δίνει στην έξοδό της τον ASCII χαρακτήρα που έχει τιμή στο δεκαδικό σύστημα X.

Για το κατάλογο των ASCII χαρακτήρων βλέπε ΠΑΡΑΡΤΗΜΑ Β.

Με την συνάρτηση αυτή μπορούμε να εκμεταλευθούμε πλήρως τις δυνατότητες διαφόρων CONTROL χαρακτήρων, ώστε να πετυχαίνουμε ευελιξία και καλαισθησία.

ΠΑΡΑΔΕΙΓΜΑ

PRINT CHR$(20)CHR$(10)CHR$(15)
είναι ισοδύναμη με την PRINT POS 10,15

5.3. LEFT$(A$,I)

Η συνάρτηση αυτή απομονώνει τους Ι πρώτους αριστερούς χαρακτήρες της μεταβλητής Α$.

ΠΑΡΑΔΕΙΓΜΑ

A$	LEFTS/(A$,3)
"NIKOS"	"NIK"
"ΒΙΚΕΝΤΙΟΣ"	"ΒΙΚ"

5.4. LEN (X$)

Η συνάρτηση αυτή δίνει το πλήθος των χαρακτήρων που έχει η αλφαριθμητική μεταβλητή Χ$.

ΠΑΡΑΔΕΙΓΜΑ

X$.	LEN(X$)
"ΒΙΚΕΝΤΙΟΣ"	9
"NIKOS"	5

5.5. MID$(A$,I,K)

Η συνάρτηση αυτή απομονώνει τους Κ χαρακτήρες της μεταβλητής Α$ αχρίζοντας από τον χαρακτήρα που βρίσκεται στη θέση Ι.

Αν στην έκφραση της συνάρτησης λείπει ο δεύτερος δείκτης Κ, τότε απομονώνει όλους τους χαρακτήρες της Α$ που βρίσκεται στην θέση Ι και δεξιά αυτής.

ΠΑΡΑΔΕΙΓΜΑ

X$	MID$(X$,3,2)	MID$(X$,2)
"NIKOS"	"KO"	"IKOS"

5.6. RIGHT$ (A$,I)

Η συνάρτηση αυτή απομονώνει τους Ι χαρακτήρες της μεταβλητής Α$ αρχίζοντας από το δεξιό άκρο της μεταβλητής και πίσω.

ΠΑΡΑΔΕΙΓΜΑ

X$	RIGHT$(X$,5)
"ΒΙΚΕΝΤΙΟΣ"	"ΝΤΙΟΣ"

5.7. STR$ (X)

Η συνάρτηση αυτή μετατρέπει τον αριθμό Χ σε STRING. Πρέπει να τονισθεί ότι το STRING ξεκινάει μ΄ένα κενό χαρακτήρα αν ο αριθμός Χ είναι θετικός.

ΠΑΡΑΔΕΙΓΜΑ

X	STR$(X)
153	153
-28	-28

5.8. VAL (A$)

Η συνάρτηση αυτή μας δίνει το αριθμητικό μέρος της μεταβλητής Α$, αρχίζοντας από αριστερά.

ΠΑΡΑΔΕΙΓΜΑ

A$	VAL(A$)
"100ABC"	100
"3A"	3
"A100"	0

ΚΕΦΑΛΑΙΟ 6

ΜΑΘΗΜΑΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Οι μαθηματικές συναρτήσεις έχουν ονόματα, που αποτελούνται από 3 χαρακτήρες αλφαβητικούς.

Στη συνέχεια του ονόματός τους και μέσα σε παρανθέσεις ( ) γράφουμε την τιμή της αριθμητικής μεταβλητής της συνάρτησης ή το όνομα κάποιας αριθμητικής μεταβλητής, που έχει λάβει τιμή από πριν (ARGUMENT), ή ακόμη και μία μαθηματική έκφραση αριθμητικών μεταβλητών, που να περιέχει άλλες μαθηματικές συναρτήσεις (σύνθεση συναρτήσεων).

ΠΑΡΑΔΕΙΓΜΑΤΑ

α. COS(3.17), EXP(-10), SINC(1.57)
β. COS(A*B+(C-D)*A),EXP(A/B+C/D-A*B),SIN(A 2+B 2)
γ. COS(EXP(SQR(X))),ATN(SIN(EXP(A+B))*10)

Αξίζει να σημειωθεί ότι η ευελιξία του προγράμματος και ο τρόπος επεξεργασίας των αριθμητικών μεταβλητών στις μαθηματικές συναρτήσεις, έχουν σαν αποτέλεσμα την μεγάλη ακρίβεια και την βέλτιστη ταχύτητα υπολογισμού τους.

Συγκεκριμμένα, το εξαγόμενο των μαθηματικών συναρτήσεων έχει ακρίβεια 12 δεκαδικών ψηφίων (βλέπε παράρτημα με πίνακες υπολογισμού μαθηματικών συναρτησεων για διάφορες τιμές μεταβλητών).

Το αποτέλεσμα των μαθηματικών συναρτήσεων μπορούμε να το στείλουμε σ΄ένα περιφερειακό με την εντολή PRINT ή να το περάσουμε σε κάποια αριθμητική μεταβλητή ή ακόμη να το χρησιμοποιήσουμε απ΄ευθείας σε συγκρίσεις.

ΠΑΡΑΔΕΙΓΜΑ

α. PRINT COS(X),.PRINT 5, ATN(A*B+C)
β. IF ABS(X-Y)Z=EPS THEN...

Στη συνέχεια δίδονται οι βασικές μαθηματικές σσυναρτήσεις, που χρησιμοποιεί η SUPER BASIC, με την σχετική ανάλυσή τους, και ακολούθως οι μέθοδοι που μπορούμε να υπολογήσουμε τις τιμές άλλων μαθηματικών συναρτήσεων με την βοήθεια αυτών.

6.1. ABS (X)

Υπολογίζει την απόλυτη τιμή της πραγματικής αριθμητικής μεταβλητής X ως εξής:

```
         X,  X>0
ABS(X)=  0,  X=0
        -X,  X<0
```

6.2. ATN (X)

Υπολογίζει το τόξο εφαπτομένης που έχει τιμή X, κατά την έκφραση:

Το ξεφ X=ω<=> εφω=X

Το τόξο δίδεται εκπεφρασμένο σε ακτίνια (RADS) και αν θέλουμε να το μετατρέψουμε σε μοίρες χρησιμοποιούμε τον τύπο

```
   180*ATN(X)
M=------------
    3.14159
```

6.3. COS (X)

Υπολογίζει το συνημίτονο της αριθμητικής μεταβλητής X. Το τόξο X σίσεται σε ακτίνια (RADS), γιά να υπολογισθεί το συνημίτονό του.

Αν έχουμε τόξο σε μοίρες και θέλουμε να βρούμε το συνημίτονό του χρησιμοποιούμε την έκφραση

K=COS(3.14159*X/180)

6.4. EXP (X)

Υπολογίζει την εκθετική δύναμη με βάση το ... (βάση νεπερίων, λογαρίθμων) και εκθέτει τον X δηλαδή

K=...

6.5. INT (X)

Δίνει τον αμέσως μικρότερο ή ίσο ακέραιο αριθμό από την τιμή της μεταβλητής X.

ΠΑΡΑΔΕΙΓΜΑ

INT(3.5)=3
INT(-3.4)=-4

6.6. LOG (X)

Δίνει τον νεπέριο λογάριθμο του X (νεπέριος=φυσικός). Εαν θέλουμε να υπολογίσουμε τον δεκαδικό λογάριθμο, χρησιμοποιούμε τον τύπο

$$LOG_{10}\ X=\frac{LOG...X}{LOG_{10}..}$$

οπότε $LOG_{10}X=LOG(X).M$
(M=0.43429448190325182765...)

6.7. RND (X)

Δίνει ένα ψευδοτυχαίο αριθμό μεταξύ των τιμών 0 και 1 με περιοδικότητα ανάλογη της τάξεως του μέγιστου αριθμού που "χωραει" σε μία αριθμητική μεταβλητή.

ΠΑΡΑΔΕΙΓΜΑ

X	RND(X)
1	5122
1	65337

6.8. SIN (X)

Υπολογίζει το ημίτονο του τόξου X

Το τόξο X δίδεται σε ακτίνια (RADS).

Εαν θέλουμε να υπολογίσουμε το ημίτονο κάποιου τόξου που δίδεται σε μοίρες, τότε χρησιμοποιούμε την έκφραση

K=SIN(M*3.14159/180) ή

K=SIN(0.0174532*M)

6.9. SGN (X)

Δίνει το πρόσημο του αριθμού X βάσει του ορισμού

```
        1, X>0
SGN(X)= 0, X=0
       -1, X<0
```

6.10. SQR (X)

Υπολογίζει την τετραγωνική ρίζα του X, ακολουθόντας την επαναληπτική μέθοδο του NEWTON

.....

6.11. TAN (X)

Υπολογίζει την εφαπτομένη του τόξου X, όπου το X δίδεται εκπεφρασμένο σε ακτίνια (RADS).

Εαν θέλουμε να υπολογίσουμε την εφαπτομένη κάποιου τόξου εκπεφρασμένου σε μοίρες τότε χρησιμοποιούμε την έκφραση

K=TAN(M*3.14159/180) ή
K=TAN(0.0174532*M)

Μετά τον καθορισμό των βασικών μαθηματικών συνρτήσεων που δομούν την SUPER-BASIC, δίδονται ορισμένες άλλες μαθηματικές συναρτήσεις που μπορούν να υπολογισθούν συνδιάζοντας τις προηγούμενες.

6.12. Υπολογισμός της συνάρτησης συνεφαπτομένη σφ(X)

σφ(X)=K=1/TAN(X) ή
σφ(X)=K=COS(X)/SIN(X) X=0,π/2

6.13. Υπολογισμός της συνάρτησης τέμνουσα τεμ(X)

$$\text{τεμν}(X)=1/\text{CIS}(X),\ X=K\Pi+\frac{\Pi}{2}\ (\Pi=3.14159),K\in Z$$

6.14. Υπολογισμός της συνάρτησης συντέμνουσας στεμ(X)

στεμ(X)=1/SIN(X), X=KΠ(Π=3.14159, K∈Z

6.15. Υπολογισμός της υπερεκθετικής α...

.....
α.=K=EXR=X*LOG(A), (A>0,X..R)

6.16. Υπολογισμός υπερβολικού ημιτόνου SINH(X)

SINH(X)=(EXP(-X))/2

6.17. Υπολογισμός υπερβολικού συνημιτόνου COSH(X)

COSH(X)=(EXP(X)+EXP(-X))/2

6.18. Υπολογισμός υπερβολικής εφαπτομένης TANH(X)

TANH(X)=(EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))

6.19. Υπολογισμός τόξου υπερβολικού ημιτόνου ARCSINH(X)

ARCSINH(X)=LOG(X=SQR(X*X+1))

6.20. Υπολογισμός τόξου υπερβολικού συνημιτόνου ARCCOSH(X)

ARCCOSH(X)=LOG(X+SQR(X*X-1))

ΚΕΦΑΛΑΙΟ 7

Α Ρ Χ Ε Ι Α

Γιά να υπάρχει συντονισμός και δομή στα μαγνητικά μέσα (δίσκους-δισκέττες), πρέπει αυτά να χωρισθούν σε λογικά μέρη, που θα περιέχουν ομοειδείς πληροφορίες. Τα λογικά αυτά μέρη καλούνται αρχεία (FILES).

Γιά να είναι εύκολο να ξεχωρίζουν τα διάφορα αρχεία μεταξύ τους, πρέπει να υπάρχει κάποιος πίνακας στην αρχή του μαγνητικού μέσου, που θα έχει σαν βασικά στοιχεία, το όνομα του αρχείου, την αρχή του αρχείου πάνω στο δίσκο και το φυσικό τέλος του αρχείου, ενώ σαν δευτερεύοντα στοιχεία, οτιδήποτε άλλα, π.χ. την φύση του αρχείου, κάποιο κλειδί (PASSWORD), το λογικό τέλος του αρχείου κ.λ.π.

Ο πίνακας αυτός δημιουργείται αυτόματα από το λειτουργικό σύστημα του υπολογιστή που καλείται DIRECTORY.

Συνεπώς κάθε αρχείο διακρίνεται με ένα όνομα, που αποτελείται το πολύ από 6 αλφαριθμητικούς κεφαλαίους χαρακτήρες και υποχρεωτικά ο πρώτος χαρακτήρας θα είναι αλφαβητικός.

Το όνομα του αρχείου καλείται FILE-NAME.

Απαραίτητη προυπόθεση είναι, όταν επικοινωνούμε με κάποιο αρχείο με τις COMMANDS RUN, LOAD, SAVE ή την εντολή OPEN, το όνομα του αρχείου που ακολουθεί στο όνομα των COMMANDS ή της εντολής OPEN να περικλείεται σε διπλές άνω αποστρόφους (") π.χ. RUN "FILE-NAME".

Τα αρχεία ανάλογα με το είδος πληροφοριών που περιέχουν, διακρίνονται σε:

α. DATA αρχεία
β. BASIC αρχεία
γ. OTHER αρχεία

και ανάλογα της προστασίας που τα χαρακτηρίζει (βλέπε στη συνέχεια τα προστατευμένα αρχεία "PROTECTED FILES"), διακρίνονται σε:

1. READ WHITE (RW)
2. WRITE (W)
3. READ (R)
4. NONE (N)

7.1. ΠΡΟΣΤΑΤΕΥΜΕΝΑ ΑΡΧΕΙΑ - ΚΛΕΙΔΙΑ (PASSWORDS)

Πολλές φορές θέλουμε κάποιο πρόγραμμα που "φυλάγεται" σε κάποιο αρχείο να είναι προστατευμένο, ώστε να μην μπορεί κάποιος άλλος χρήστης να το ανοίξει και να διαβάσει τις εντολές του ή να το τρέξει με την εντολή RUN.

Τότε κατά την δημιουργία του αρχείου, χρησιμοποιούμε κάποιο περιορισμό, ένα κλειδί (PASSWORD), που είναι μία 4 ψηφία λέξη με αλφαριθμητικούς χαρακτήρες, και ο πρώτος χαρακτήρας υποχρεωτικά αλφαβητικός.

Μόνο αν γνωρίζουμε το κλειδί αυτό, μπορούμε να εκτελέσουμε την εργασία (πρόγραμμα) που έχουμε κλειδώσει.

Αν ένα αρχείο χαρακτηρίζεται εκτός από το όνομα του και από κάποιο κλειδί, τότε τα λέμε προστατευμένα αρχεία (PROTESTED FILES).

Ανάλογα με το είδος προστασίας που έχουμε ορίσει γιά κάποιο αρχείο, διακρίνουμε διαγορετικά επίπεδα.

7.1.1. READ-WHITE (RN)

Το αρχείο δεν είναι προστατευμένο, άρα μπορούμε να γράψουμε και να διαβάσουμε πληροφορίες χωρίς την χρήση κλειδιού.

7.1.2. WHITE (W)

Το αρχείο είναι προστατευμένο, όσο αφορά το γράψιμο πληροφοριών, δηλαδή γιά να κάνουμε SAVE, πρέπει να δόσουμε και το κατάλληλο κλειδί. Επίσης το αρχείο αυτό δεν μπορούμε να το φορτώσουμε στην μνήμη και να εκτελέσουμε την COMMAND LIST, αν δεν γνωρλιζουμε το κλειδί.

Αν το επιχειρήσουμε, τότε ο υπολογιστής θα απαντήσει με (ΣΝ ΛΑΘΟΣ).

Ενα WRITE προστατευμένο αρχείο, μπορούμε να το τρέξουμε χωρίς να χρειάζεται το κλειδί, απλώς με την εντολή RUN και το όνομά του.

7.1.3. READ (R)

Το αρχείο είναι προστατευμένο.

7.1.4. NONE (N)

Το αρχείο είναι πλήρως προστατευμένο και δεν μπορούμε να το εκτελέσουμε με την εντολή RUN, ούτε να το διαβάσουμε με την αλληλουχία των εντολών LOAD και LIST.

Για να εκτελεσθεί, απαιτείται μαζί με το όνομά του, να δοθεί και το κλειδί του.

7.2. ΑΡΧΕΙΑ ΔΕΔΟΜΕΝΩΝ (DATA)

Τα αρχεία DATA έχουν σαν σκοπό να περιέχουν γιά πληροφορίες διάφορα δεδομένα που εξυπηρετούν ορισμένες εφαρμογές είτε εμπορικές είτε επιστημονικές.

Γιά να μπορέσουμε να επικοινωνήσουμε με ένα αρχείο DATA, πρέπει πρώτα να ανοίξουμε με την εντολή OPEN σε κάποιο κανάλι επικοινωνίας (βλέπε εντολές OPEN και INTEX στο κεφάλαιο 3).

Με τα DATA αρχεία μπορεί ο χρήστης να επικοινωνεί μόνο με τις εντολές INPUT και PRINT άμεσα, ενώ έμμεσα με τις εντολές, LIST, PUTM, PYTD.

Απαγορεύεται η χρήση των εντολών RUN, LOAD, SAVE, σε DATA αρχεία.

Μία ειδική κατηγορία DATA αρχείων που χρησιμοποιείται αποκλειστικά από DBMS (DATA BASE MANAGEMENT SYSTEM) της GIGATRONICS είναι τα κύρια αρχεία (MASTER FILES) και τα αρχεία κλειδιά (KEY-FILES).

Ενα κύριο αρχείο περιέχει ομάδες στοιχείων, που έχουν μία συγκεκριμένη δομή δηλαδή τα ίδια πεδία και με την ίδια σημασία τά αντίστοιχα πεδία των διαφορετικών ομάδων.

Οι ομάδες καλούνται ITEMS και τα πεδία RECORDS.

Ενα κλειδί αρχείο, που πάντοτε αναφέρεται σε κάποιο κύριο αρχείο, περιέχει σαν στοιχεία, τα περιεχόμενα κάποιου συγκεκριμμένου πεδίου των ομάδων του κυρίου αρχείου στο οποίο αναφέρεται, και την αρχή της ομάδας του κύριου αρχείου, που περιέχει το αντίστοιχο περιεχόμενο του πεδίου.

ΠΑΡΑΔΕΙΓΜΑ

Κύριο αρχείο με τα εξής πεδία:

```
                          ------------
Αρχή ομάδας -1    200     !ΚΩΔΙΚΟΣ   !
                          ------------
                          ----------------
                          !ΟΝΟΜΑ         !
                          ----------------
                          ------------------------
                          !ΕΠΙΘΕΤΟ               !
                          ------------------------

Αρχή ομάδας -2    250          .
                               .
                               .
                               .
                               .
                               .
                               .
                          ------------
Αρχή ομάδας N 200+N*50    !ΚΩΔΙΚΟΣ   !
                          ------------
                          ----------------
                          !ΟΝΟΜΑ         !
                          ----------------
                          ------------------------
                          !ΕΠΙΘΕΤΟ               !
                          ------------------------
```

Κλειδί αρχείο με στοιχεία:

```
-------------------
!ΚΩΔΙΚΟΣ          !
-------------------      , αρχή ομάδας 1 (200)
-------------------
!ΚΩΔΙΚΟΣ          !
-------------------      , αρχή ομάδας 2 (250)

                .
                .
                .
                .
                .
-------------------
!ΚΩΔΙΚΟΣ          !      , αρχή ομάδας N (200+N*50)
-------------------
```

Στο παράδειγμα υποθέτουμε ότι στο κύριο αρχείο, το κάθε ITEM έχει μήκος 50 χαρακτήρων και αποτελείται από 3 RECORDS. Η αρχή του κάθε ITEM δεν είναι παρά εσωτερική διεύθυνση της αρχής του ITEM στο αρχείο.

Τότε το κλειδί αρχείο καθορίζεται από τον χρήστη να περιέχει σαν στοιχεία, τα περιεχόμενα του πρώτου πεδίου καθενός ITEM.

7.2.1. ΔΟΜΗ ενός κύριου αρχείου DATA

Ενα κύριο αρχείο DATA έχει σαν εσωτερική δομή την ακόλουθη:

Στη διεύθυνση 0 - 9	περιέχει την αρχή (εσωτερική διεύθυνση του αρχείου) των καθ΄εαυτού πληροφοριών.
Στη διεύθυνση 10-19	περιέχει το λογικό τέλος του κύριου αρχείου.
Στη διεύθυνση 20	περιέχει (βλέπε UTILITY SGREEN) που αναφέρεται στο κύριο αρχείο.

Επειδή το SCREEN δεν έχει σταθερό μήκος, αλλά εξαρτάται από τον αριθμό, το μέγεθος και την μορφή των πεδίων, που το δομούν, γι΄αυτό δεν δίνονται στο εξής εσωτερικές διευθύνσεις του αρχείου, αλλά η σειρά προτεραιότητας των στοιχείων που καταγράφονται.

Συνεπώς μετά το SCREEN καταγράφεται:

1. Το μέγεθος του ITEM.

2. Ο αριθμός των RECORDS.

3. Το μέγεθος του κάθε RECORD με την σειρά.

4. Η γραμμή του SCREEN που βρίσκεται το κάθε RECORD.

5. Από πιό BYTE της γραμμής του SCREEN αρχίζει το κάθε RECORD.

6. Το πλήθος των αρχείων κλειδιών που αναφέρονται στο κύριο αρχείο.

7. Ο αριθμός του RECORD γιά το αντίστοιχο του αρχείο - κλειδί.

8. Ο αριθμός του RECORD που θέλουμε να είναι μοναδικό (αν υπάρχει)

9. Ο αριθμός των DISPLAYS RECORDS.

10. Οι αριθμοί των RECORDS που θέλουμε να είναι DISPLAY.

11. Τα ονόματα των αρχείων κλειδιών, με τα οποία συνδέονται τα DISPLAYS RECORDS.

12. Ο αριθμός των RECORDS που καθορίζονται από κάποιο υπολογισμό (COMPUTER-FIELDS).

13. Οι αριθμοί των RECORDS που είναι COMPUTED.

14. Ο τρόπος υπολογισμού των COMPUTED RECORDS.

Γιά τα COMPUTED FIELDS προηγείται το νούμερο του RECORD, ακολουθεί ο τρόπος υπολογισμού αυτού, και επαναλαμβάνεται τόσες φορές, όσες είναι και ο αριθμός των COMPUTED FIELDS.

Μόλις τελειώσουν τα στοιχεία αυτά, αρχίζει η καταγραφή των ITEMS.

7.2.2. ΔΟΜΗ ενός DATA αρχείου-κλειδιού

Ενα DATA αρχείο-κλειδί έχει σαν εσωτερική δομή την ακόλουθη.

Στη διεύθυνση 0 - 9 γράφεται το λογικό τέλος του.

Στη διεύθυνση 10-19 γράφεται το λογικό τέλος του κύριου αρχείου, στο οποίο αναφέρεται.

Στη διέυθυνση 20 και κάτω γράφονται τα ITEMS.

Κάθε ITEM έχει 2 RECORDS.

Το πρώτο περιέχει το πειεχόμενο του RECORD που επιλέχθηκε από το κύριο αρχείο.

Το δεύτερο περιέχει την αρχή του ITEM που περιέχει την αντίστοιχη πληροφορία του πρώτου RECORD, στο κύριο αρχείο.

7.3. BASIC ΑΡΧΕΙΑ

Τα αρχεία BASIC χρησιμεύουν γιά να αποθηκεύουμε και να καλούμε προγράμματα γραμμένα σε γλώσσα BASIC.

Οταν θέλουμε να αποθηκεύσουμε ένα πρόγραμμα σε ένα αρχείο με όνομα π.χ. "TEST" τότε χρησιμοποιούμε την εντολή "SAVE", π.χ: SAVE "TEST".

Μόλις ο υπολογιστής ολοκληρώσει την μεταφορά του προγράμματος στο αρχείο "TEST" θα γράψει στην οθόνη:

SAVED "TEST", δηλαδή "Το φύλαξα στο αρχείο TEST"

Για να φορτώσουμε στη μνήμη ένα πρόγραμμα που το έχουμε φυλάξει στο αρχείο "TEST" χρησιμοποιούμε την εντολή: LOAD

Γράφουμε τότε: LOAD "TEST"

Για να εκτελέσουμε κάποιο πρόγραμμα που το έχουμε φυλάξει στο αρχείο "TEST" γράφουμε:

RUN "TEST"

7.4. OTHER ARXEIA (O)

Τα αρχεία αυτά χρησιμεύουν γιά να καταχωρούνται πληροφορίες, που δεν έχουν τη μορφή μεταβλητών BASIC ούτε και είναι προγράμματα.

Εται ένα πρόγραμμα γραμμένο σε MACHINE CODE (τη γλώσσα που καταλαβαίνει άμεσα ο MICROPROCESSOR) μπορεί να αποθηκευτεί μόνο σε ένα αρχείο, που είναι OTHER.

Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε την εντολή PUTM και PUTD.

ΠΑΡΑΡΤΗΜΑ Α!

ΠΙΝΑΚΑΣ ΛΑΘΩΝ

Α. ΔΙΣΚΕΤΤΑ (FLOPPY DRIVE)

ΚΩΔΙΚΟΣ ΛΑΘΟΥΣ	ΑΝΑΛΥΣΗ ΛΑΘΟΥΣ
1.	DRIVE NOT READY πόρτα ανοικτή, δεν υπάρχει δισκέττα στο DRIVE.
2.	SEEK ERROR:κεφαλή σε λάθος TRACK ((σως υπάρχει λάθος στο DRIVE, πρόβλημα HARDWARE.
3.	Ακυρο όνομα DRIVE (π.χ. DSK "T")
4.	Δεν βρίσκεται το TRACK 0 ((σως δεν έχει γίνει INITIALISE στο TRACK 0).
5.	Δεν βρίσκεται INTEX HOLE (HARDWARE λάθος).
6.	Δισκέττα WRITE PROTECTED (προσπαθείς να γράψεις σε δισκέττα προστατευμένη).
7.	Τα DATA που μόλις έγραψα στη δισκέττα δεν διαβάζονται σωστά, ένα προς ένα, ((σως υπάρχει λάθος στο μαγνητικό μέσο ή στη μνήμη).
8.	Λείπουν HEADERS στο TRACK (INITIALISE το TRACK)
9.	FE.(FRAMING ERROR). Τα BYTES δεν έχουν σωστό FRAMING.
10.	OV. (HEADERS OVERRUN). Πρόβλημα στο ACIA (;)
11.	FE.(FRAMING ERROR).
12.	PE:(PURITY ERROR).

13. PE+FE (PURITY ERROR+FRAMING ERROR).

14. PE+OV (PURITY ERROR+HEADERS OVERRUN).

15. PE+OV+FE (PURITY ERROR+HEADERS ONERRUN +FRAMING ERROR)

DATA FIELD:

16. Λείπουν DATA (INITIALISE TRACK).

17. FE. (FRAMING ERROR): Τα BYTES δεν είναι σωστά πλαισιωμένα, κάνε INITIALISE το TRACK αν το λάθος επιμένει τότε η δισκέττα είναι χαλασμένη.

18. OV (HEADERS OVERRUN).

19. FE+OV (FRAMING ERROR+HEADERS OVERRUN).

24. CHECKSUM λάθος. Κάνε INITIALISE το TRACK.

OM. OUT OF MEMORY (γέμισμα μνήμης).

ΠΑΡΑΡΤΗΜΑ Β

ΚΑΤΑΛΟΓΟΣ UTILITIES ΤΗΣ SUPER BASIC

H-DOS Φορτώνει το λειτουργικό σύστημα και τη γλώσσα της μνήμης από το μαγνητικό μέσο.

CREATE Δημιουργεί αρχεία στα μαγνητικά μέσα.

DELETE Διαγράφει αρχεία από τα μαγνητικά μέσα.

RENAME Αλλάζει το όνομα αρχείων στο ευρετήριο.

COPIER Σύνθετο πρόγραμμα,που αναφέρεται μόνο σε δισκέττες και κάνει:

1. INITIALISE (εκχώρηση αρχικής τιμής)
2. Αντιγράφει (BACK-UP) μία δισκέττα σε άλλη
3. Αντιγράφει μέρος ή το σύνολο μιάς δισκέττας σε κάποια άλλη (δίνονται όρια).

PACKER Αποδίδει στο σύστημα το χώρο των αρχείων (από το DELETE), που έχουν διαγραφεί, (μόνο γιά FLOPPY).

LIST Τυπώνει ή εμφανίζει στην οθόνη προγράμματα ή αρχεία (RUN-LIST).

INIT-H INITIALISE σε δίσκους (HARD DISCS).

COPYG Μεταφέρει αρχεία δομής GIGA DATA BASE σε άλλα αρχεία, προσδιορίζοντας ταυτόχρονα την αρχή και το τέλος τους.

SORT Ταξινομεί αρχεία στους δίσκους και στην μνήμη.

GINTEX Ευρετηριάζει αρχεία.

GIGA Το κυρίως μέρος του GIGA D.B.M.S. με το οποίο:

1. Εισάγουμε πληροφορίες (KEY-IN).

2. Διαβάζουμε πληροφορίες (INQUIRY).

3. Διορθώνουμε πληροφορίες.

4. Διαγράφουμε εγγραφές.

SCREEN Μορφοποιεί την οθόνη και δομεί τα αρχεία που αναφέρονται σ΄αυτήν.

GREP Εμφανίζει στην οθόνη ή εκτυπώνει αναφορά με πληροφορίες, που αναφέρονται στην μορφοποίηση, που έχει γίνει .

COPYF Αντιγράφει αρχεία ή προγράμματα .

MULTI Επιτρέπει την ταυτόχρονη χρήση τερματικών (TIME SHARING ROUTINE).

CAT Εμφανίζει στην οθόνη ή εκτυπώνει το ευρετήριο των αρχείων - προγραμμάτων των μαγνητικών μέσων .

REDIT Αλλάζει μαζικά λέξεις, φράσεις ή χαρακτήρες, ενός προγράμματος, σε ότι ζητηθεί, π.χ. PRINT στο δίσκο, PRINT στο χαρτί .

BACK-UP Αντιγράφει μέρος ή το σύνολο στοιχείων, που βρίσκονται στο μαγνητικό δίσκο, σε άλλο δίσκο (μόνο γιά HARD-DISC).

GPRINT Δομημένη εμφάνιση αρχείου στο χαρτί ή στην οθόνη, με επιλογή σειράς πεδίου, συνόλων πληροφοριών πεδίου κ.λ.π., σε οριζόντια ή κάθετη διάταξη.

KEIM Επεξεργασία κειμένου (WORD PROCESSING: δόμηση κειμένων γιά εμφάνιση ή εκτύπωση, σύμφωνα με τους κανόνες της Ελληνικής Γραμματικής).