# QL

## Οδηγός Χρήσεως

- Εισαγωγή
- Οδηγός Αρχάριου
- Έννοιες
- Δεσμευμένες Λέξεις

# QL

## Οδηγός Χρήσεως

Μετάφραση & Επιμέλεια: Χαρ. Φραγκάκις, Καθηγητής Α.Π.Θ.



**MPS**
ΠΛΗΡΟΦΟΡΙΚΗ Β. ΕΛΛΑΔΟΣ
ΠΟΛΥΤΕΧΝΕΙΟΥ 47, ΤΗΛ. 540 246
546 25 - ΘΕΣΣΑΛΟΝΙΚΗ

# QL

## Οδηγός Χρήσεως

- Εισαγωγή
- Οδηγός Αρχάριου
- Έννοιες
- Δεσμευμένες Λέξεις

# ΠΕΡΙΕΧΟΜΕΝΑ

ΔΕΣΜΕΥΜΕΝΕΣ ΛΕΞΕΙΣ (KEYWORDS)

# ΠΡΟΣΟΧΗ!

## ΓΙΑ ΤΟ ΕΛΛΗΝΙΚΟ ROM ΤΟΥ QL

---

Ο ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΗΣ QL ΟΤΑΝ ΧΡΗΣΙΜΟΠΟΙΕΙ ΤΗΝ ΕΛΛΗΝΙΚΗ ROM ΕΧΕΙ ΕΛΛΗΝΙΚΟΥΣ ΚΑΙ ΛΑΤΙΝΙΚΟΥΣ ΧΑΡΑΚΤΗΡΕΣ. ΓΙΑ ΝΑ ΑΛΛΑΞΟΥΜΕ SET ΧΑΡΑΚΤΗΡΩΝ ΑΠΟ ΑΓΓΛΙΚΑ ΣΤΑ ΕΛΛΗΝΙΚΑ ΚΑΙ ΑΝΤΙΣΤΡΟΦΑ, ΠΑΤΟΥΜΕ ΜΑΖΙ ΤΑ ΠΛΗΚΤΡΑ

CTRL + ALT + CAPS LOCK

ΣΤΗΝ ΕΛΛΗΝΙΚΗ ΕΚΔΟΣΗ ΤΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ (ARCHIVE, ABACUS, QUILL ΚΑΙ EASEL) ΤΑ ΓΡΑΜΜΑΤΑ ΓΙΑ ΤΙΣ ΔΙΑΦΟΡΕΣ ΕΠΙΛΟΓΕΣ ΕΙΝΑΙ ΣΤΑ **ΕΛΛΗΝΙΚΑ**. ΓΙΑΥΤΟ ΜΕΤΑ ΤΗ ΦΟΡΤΩΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΑΝ ΠΡΩΤΗ ΕΝΕΡΓΕΙΑ ΠΗΓΑΙΝΟΥΜΕ ΣΤΟ ΕΛΛΗΝΙΚΟ ΑΛΦΑΒΗΤΟ ΠΑΤΩΝΤΑΣ

CTRL + ALT + CAPS LOCK

# QL

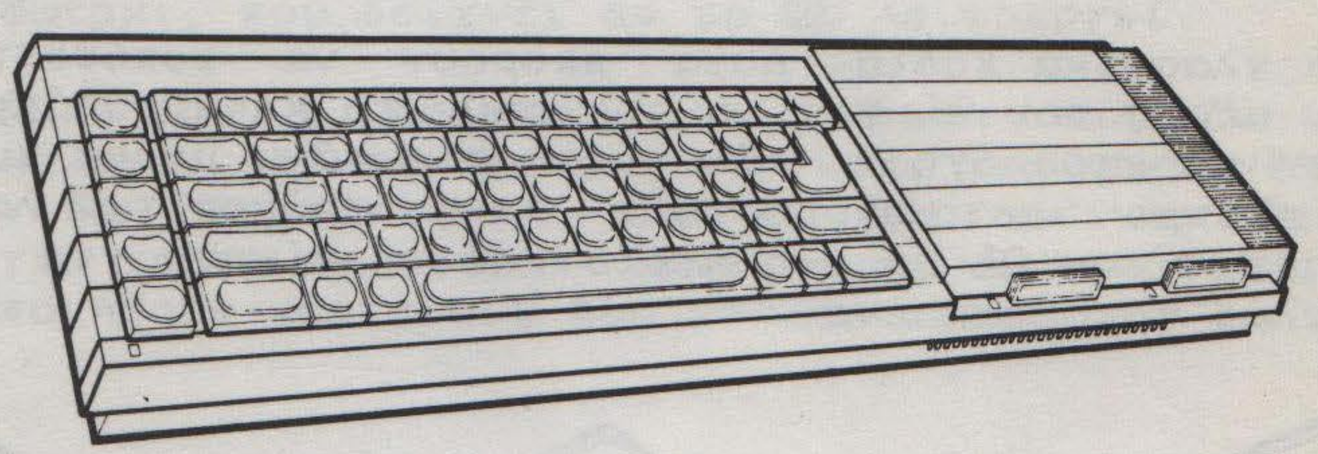
## Εισαγωγή

ΚΕΦΑΛΑΙΟ 1

# ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΥΠΟΛΟΓΙΣΤΗ QL

'Οταν ανοίξετε το κουτί με τον υπολογιστή QL θα βρείτε:

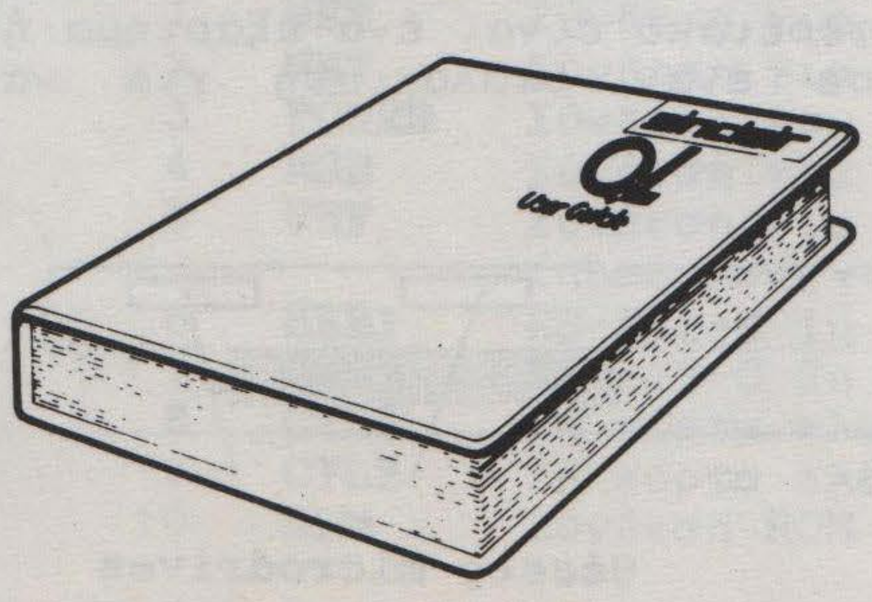
- Το βιβλίο "Οδηγός Χρήστη" του QL.
- Τον υπολογιστή QL.
- Ενα μετασχηματιστή.
- Ενα καλώδιο κεραίας:



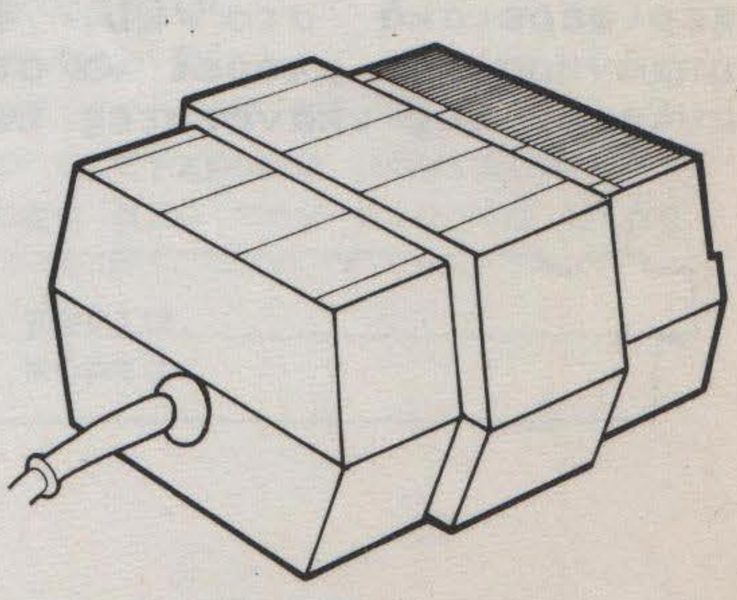
Το καλώδιο έχει περίπου 2 μέτρα μήκος και έχει διαφορετικές συνδέσεις στο κάθε άκρο. Χρησιμοποιείται για να συνδέσετε το QL σας με την υποδοχή κεραίας της τηλεόρασής σας.

- Ενα καλώδιο δικτύου:

Αυτό είναι περίπου 2 μέτρα μακρύ και έχει τον ίδιο τύπο σύνδεσης στο κάθε άκρο. Χρησιμοποιείται για να συνδέσετε το QL σας με τα άλλα QL ή SPECTRUM ώστε τα δεδομένα και τα μηνύματα να μπορούν να ανταλλάσσονται ανάμεσά τους.

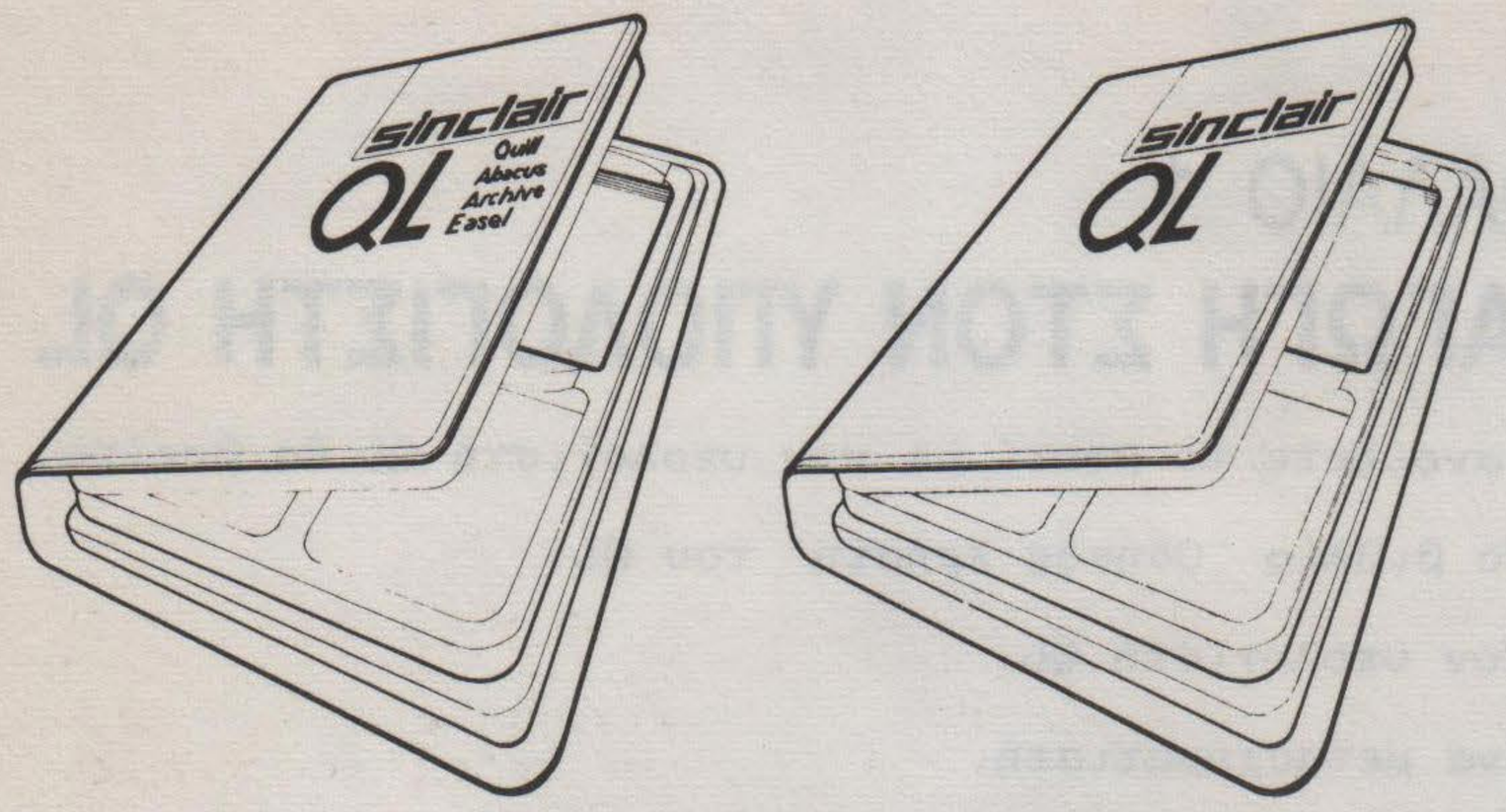



Ο Οδηγός χρήσης του QL



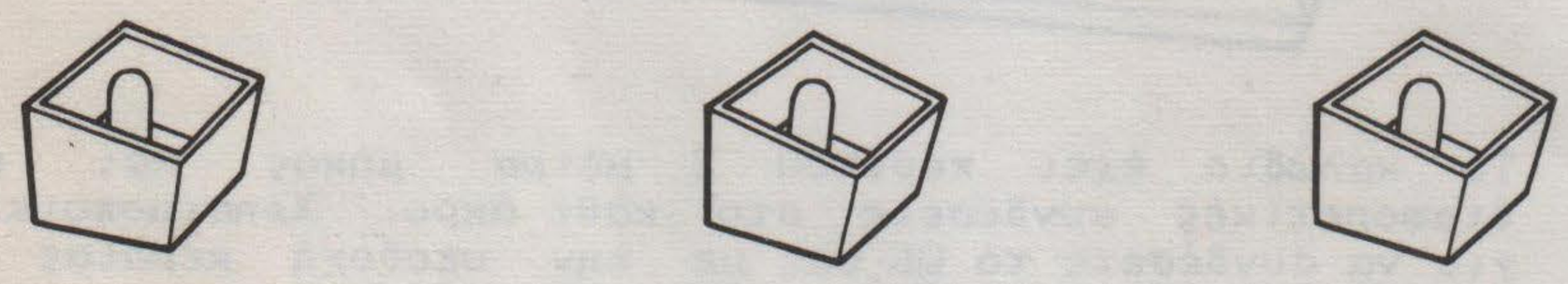
Ο μετασχηματιστής

- Τέσσερις μικροκασέτες οι οποίες περιέχουν: το QL Abacus, το QL Archive, το QL Easel, το QL Quill.




θήκες μικροκασετών

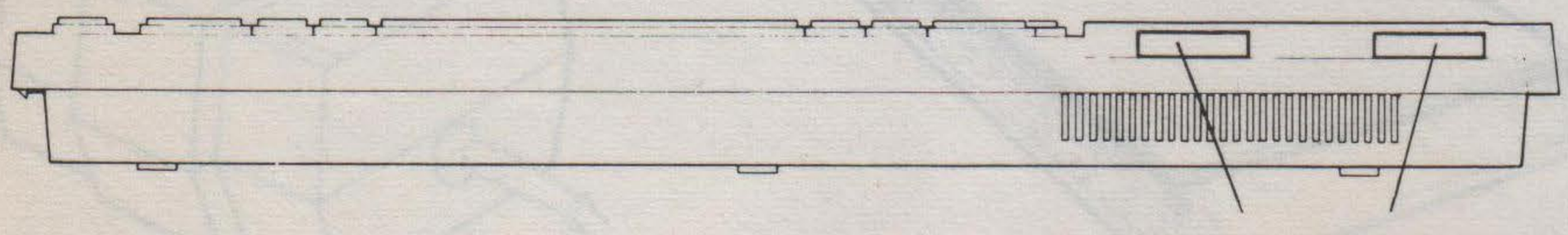
- Τρία πλαστικά πόδια: Αυτά μπορούν να τοποθετηθούν στο κάτω μέρος του QL για να σηκώσουν το πληκτρολόγιο για πιο άνετη πληκτρολόγηση. Οι μύτες στο πάνω μέρος των ποδιών θα πρέπει να εφαρμοστούν στις τρύπες των λαστιχένιων πελμάτων του QL με μία περιστροφική κίνηση για να είναι ασφαλώς τοποθετημένα.



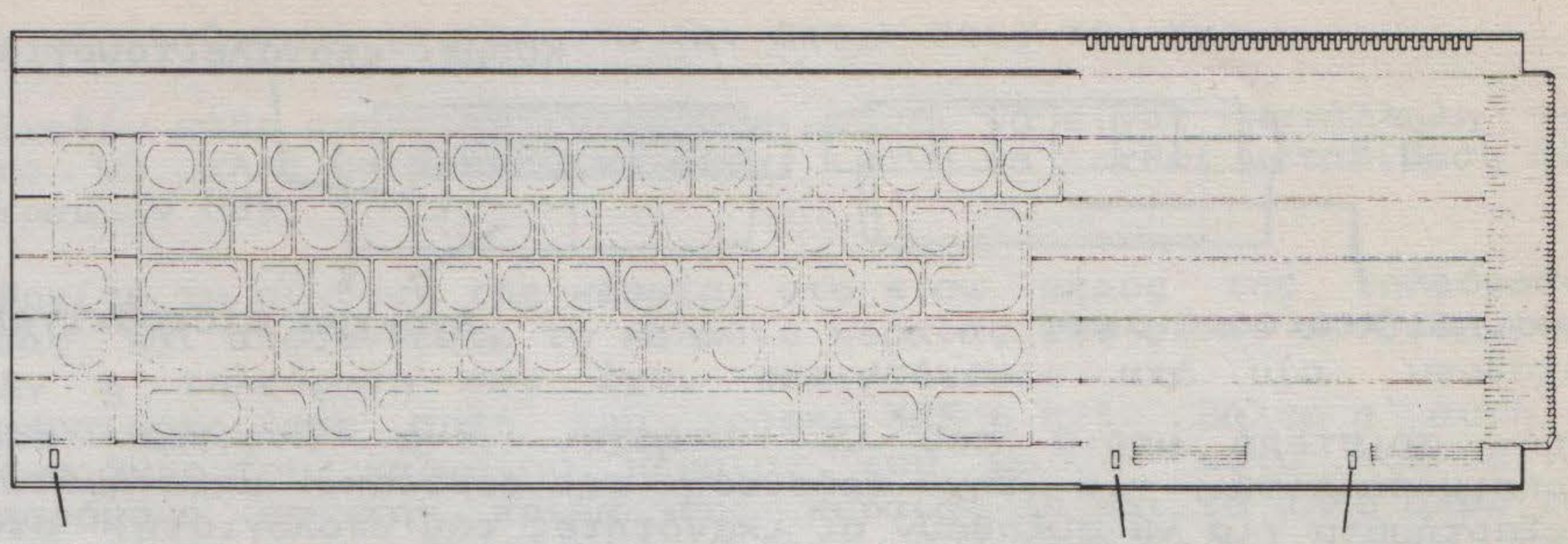
Τρία πλαστικά πόδια

## ΠΕΡΙΓΡΑΦΗ ΤΟΥ QL

Στο πίσω μέρος και στις πλευρές του υπολογιστή υπάρχει μία σειρά από συνδέσεις. Οι περισσότερες από αυτές δεν είναι πάντα απαραίτητες, χρειάζονται για να ενώνετε διάφορα περιφερειακά στο QL, (ένα περιφερειακό είναι ένα εξάρτημα ή μηχάνημα που μπορεί να συνδεθεί σε έναν υπολογιστή για να αυξήσει τις ικανότητες του).



Θέσεις microdrives

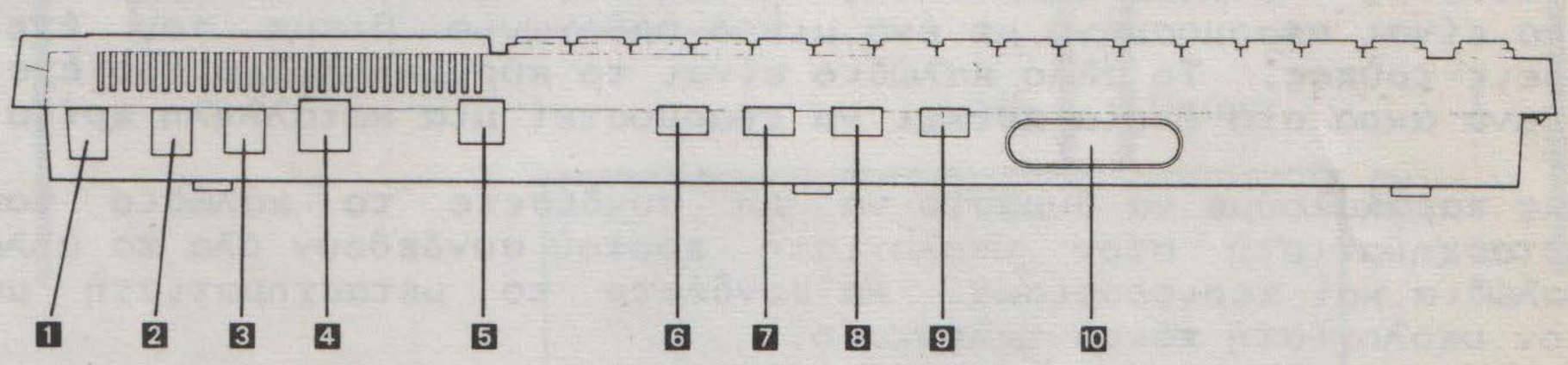


Φωτάκι λειτουργίας

Φωτάκια microdrives

Στην μπροστινή δεξιά μεριά του υπολογιστή υπάρχουν δύο ανοίγματα. Αυτά είναι τα δύο Microdrives του QL στα οποία τοποθετούνται μικροκασέτες. Οι μικροκασέτες χρησιμοποιούνται για την αποθήκευση προγραμμάτων και δεδομένων στο QL. Δίπλα σε κάθε άνοιγμα υπάρχει ένα φωτάκι. Οταν το φωτάκι ανάβει, το Microdrive βρίσκεται σε λειτουργία και η μικροκασέτα δε θα πρέπει να μετακινηθεί. Στο αριστερό εμπρός μέρος υπάρχει ακόμα ένα φωτάκι, που δείχνει αν το QL λειτουργεί.

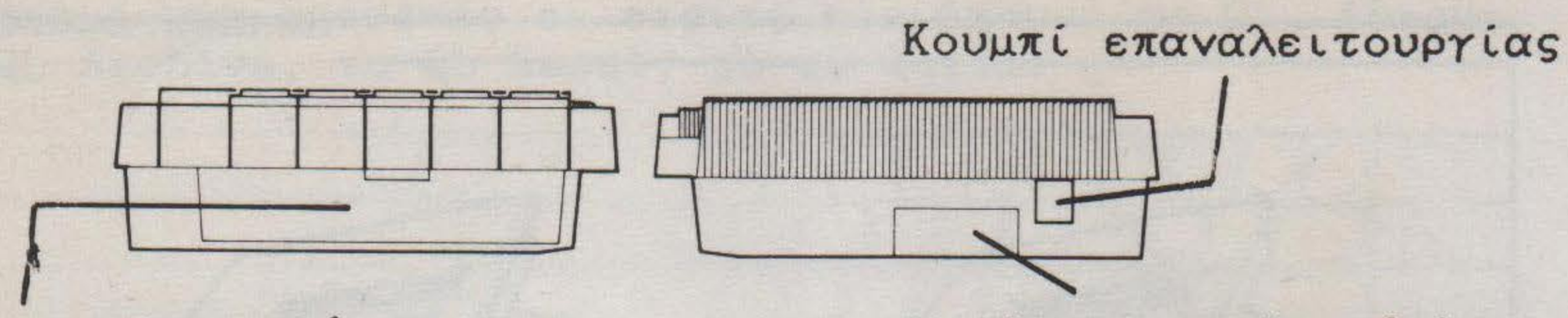
Στο δεξιό άκρο του QL υπάρχει μία σύνδεση καλυμένη με ένα πλαστικό κάλυμα. Αυτή χρησιμοποιείται για να συνδεθούν έξι επί πλέον QL Microdrives. Τα ZX Microdrives δεν είναι κατάλληλα για χρήση με το QL. Κενές όμως μικροκασέτες μπορούν να χρησιμοποιηθούν και στους δύο υπολογιστές.




Στο πίσω μέρος του υπολογιστή υπάρχουν υποδοχές για τη σύνδεση των:

| | | |
|---|---|---|
| 1 | **NET** | Σύνδεση για το δίκτυο του QL |
| 2 | **NET** | Σύνδεση για το δίκτυο του QL |
| 3 | **POWER** | Σύνδεση του μετασχηματιστή με το QL |
| 4 | **RGB** | Σύνδεση για μονόχρωμο ή έγχρωμο μόνιτορ |
| 5 | **UHF** | Σύνδεση για την υποδοχή από την κεραία μιας τηλεόρασης |
| 6 | **SER1** | RS-232-C 1η σειριακή πόρτα |
| 7 | **SER2** | RS-232-C 2η σειριακή πόρτα |
| 8 | **CTL1** | 1η πόρτα ελέγχου |
| 9 | **CTL2** | 2η πόρτα ελέγχου |
| 10 | **ROM** | Σύνδεση ROM στο QL |

Τα ZX ROM Cartridges δεν είναι συμβατά με τα QL ROM cartridges και δεν μπορούν να χρησιμοποιηθούν στο QL.




Στην αριστερή μεριά του QL υπάρχει ένα άνοιγμα που χρησιμοποιείται για την πρόσθεση περιφερειακών μηχανημάτων (εξαρτήματα για να αυξηθούν οι ικανότητες του υπολογιστή) στο QL είτε μεμονωμένα - απευθείας στο QL - ή πολλαπλά χρησιμοποιώντας τη μονάδα (Module) επέκτασης του QL.

Το κουμπί επανατοποθέτησης (RESET) του QL είναι στη δεξιά άκρη του υπολογιστή δίπλα στο άνοιγμα της επέκτασης των Microdrives. Χρησιμοποιείται για να "επανατοποθετηθεί" το σύστημα στην κατάσταση που ήταν ακριβώς όταν είχε τεθεί σε λειτουργία. Το πάτημα του RESET οποιαδήποτε στιγμή θα προκαλέσει την εξαφάνιση όλων των προγραμμάτων στον υπολογιστή και μπορεί μερικές φορές να καταστρέψει τα δεδομένα που έχουν γραφτεί στις μικροκασέτες. Χρησιμοποιείτε το RESET με προσοχή, να βγάζετε πάντα τις μικροκασέτες πριν το πατήσετε

**Διάταξη των εξαρτημάτων**

**Ο μετασχηματιστής**

Για να λειτουργήσει ο υπολογιστής, πρέπει να γίνουν διάφορες συνδέσεις: Ο μετασχηματιστής του QL σας έχει δύο καλώδια. Το ένα είναι εφαρμοσμένο με ένα μικρό ορθογώνιο βίσμα που έχει τρεις τρύπες. Το άλλο καλώδιο είναι το κύριο καλώδιο που έχει γυμνά άκρα στα οποία πρέπει να εφαρμοστεί μια κατάλληλη πρίζα.

Σας παρακαλούμε να θυμάστε να μη συνδέσετε το καλώδιο του μετασχηματιστή στον υπολογιστή προτού συνδεθούν όλα τα άλλα καλώδια και περιφερειακά. Να συνδέετε το μετασχηματιστή με τον υπολογιστή πάντα τελευταίο.

**Το monitor**

'Οταν ο μετασχηματιστής συνδεθεί το QL θα λειτουργεί, αλλά δε θα μπορείτε να δείτε τι κάνει. Γι' αυτό θα χρειαστείτε κάποιο είδος οθόνης. Το QL μπορεί να χρησιμοποιήσει τηλεόραση ή μόνιτορ.

'Ενα μόνιτορ είναι μία εξειδικευμένη συσκευή που έχει μία οθόνη σαν της τηλεόρασης, αλλά δίνει ένα απ' ευθείας αποτέλεσμα και δε μπορεί να δεχτεί σήματα τηλεοπτικού σταθμού. Μια τηλεόραση δε θα μπορέσει να εμφανίσει τα μικρότερα στοιχεία του QL ικανοποιητικά και μόνο ένα περιορισμένο μέρος πληροφοριών θα μπορέσουν να εμφανίζονται κάθε φορά. 'Ενα μόνιτορ έχει καλύτερη διακριτικότητα και μπορεί να εμφανίσει περισσότερο κείμενο, αλλά θα είναι πιθανά πιο ακριβό από την αντίστοιχη τηλεόραση.

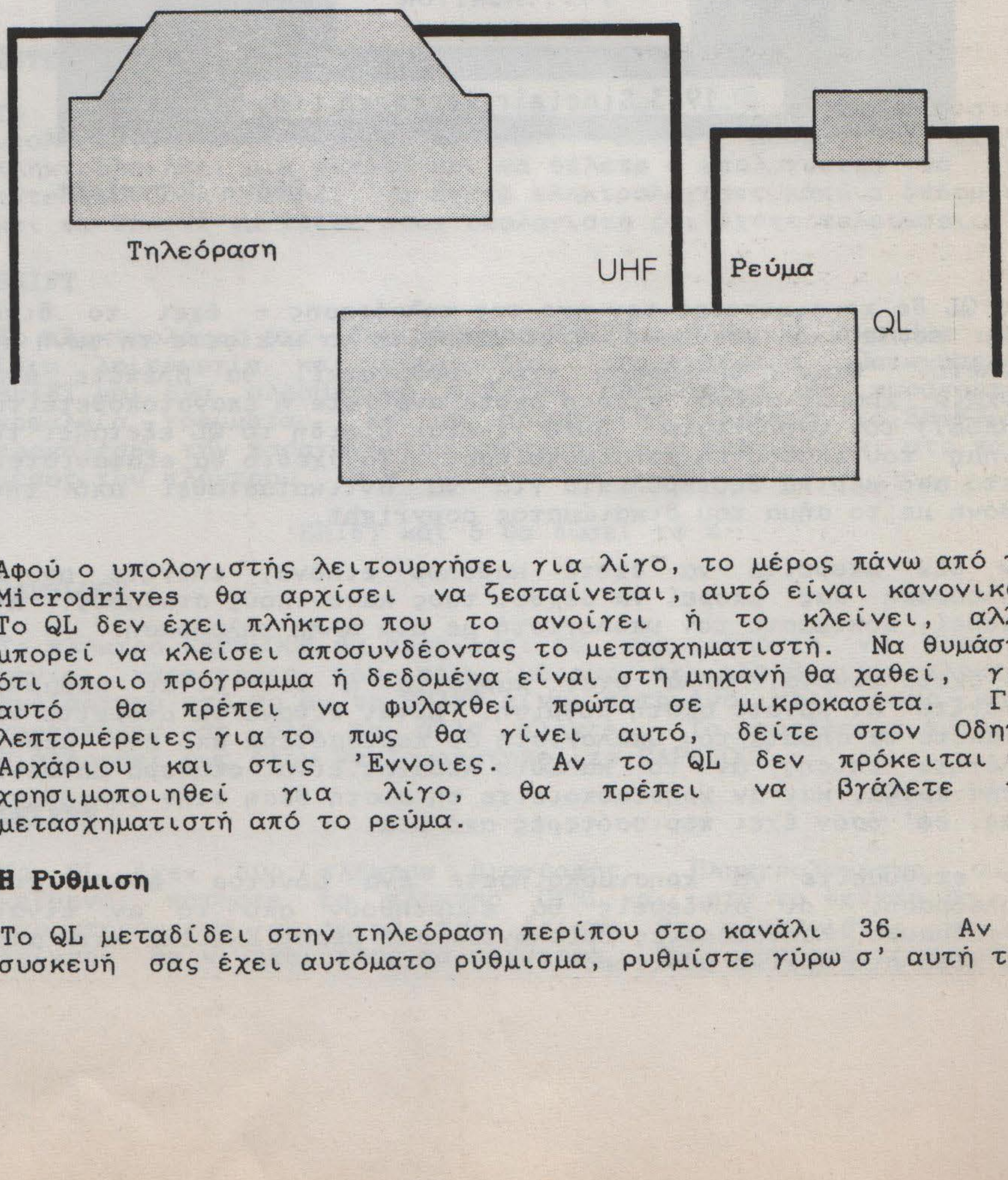
Για να γίνει χρήση της έγχρωμης οθόνης του QL, φυσικά θα είναι απαραίτητα ένα έγχρωμο μόνιτορ ή μία τηλεόραση, αλλά ο υπολογιστής μπορεί να δουλέψει τέλεια και σε άσπρο και μαύρο,

παρουσιάζοντας τα χρώματα σαν αποχρώσεις του γκρι.

Σχεδόν κάθε σημερινή τηλεόραση μπορεί να είναι κατάλληλη για το QL αλλά θα πρέπει να είναι ικανή να δεχθεί μεταβίβαση 625 γραμμών UHF.

Βρείτε τη σύνδεση της κεραίας στο πίσω μέρος της τηλεόρασης σας και αποσυνδέστε το καλώδιο κεραίας που είναι συνδεδεμένο. Αν η τηλεόραση σας έχει περισσότερες από μία υποδοχές χρησιμοποιήστε αυτή που γράφει UHF ή 625. Βάλτε σ' αυτή το ένα άκρο του καλωδίου κεραίας του QL - αυτό που είναι παρόμοιο με την πρίζα της κεραίας - και το άλλο άκρο του καλωδίου στην θέση που γράφει UHF στο πίσω μέρος του υπολογιστή.

Βάλτε το μετασχηματιστή στην ανάλογη κύρια υποδοχή και ανάψτε τον υπολογιστή. Βγάλτε όλες τις μικροκασέτες από τα Microdrives και σπρώξτε το μικρό βίσμα του μετασχηματιστή στην τριπλή πρίζα που γράφει POWER στο πίσω μέρος του QL. Το φωτάκι λειτουργίας κάτω από το πλήκτρο F5 θα πρέπει τώρα να ανάψει και η σύνδεση σας θα πρέπει να εμφανίζεται όπως παρακάτω:

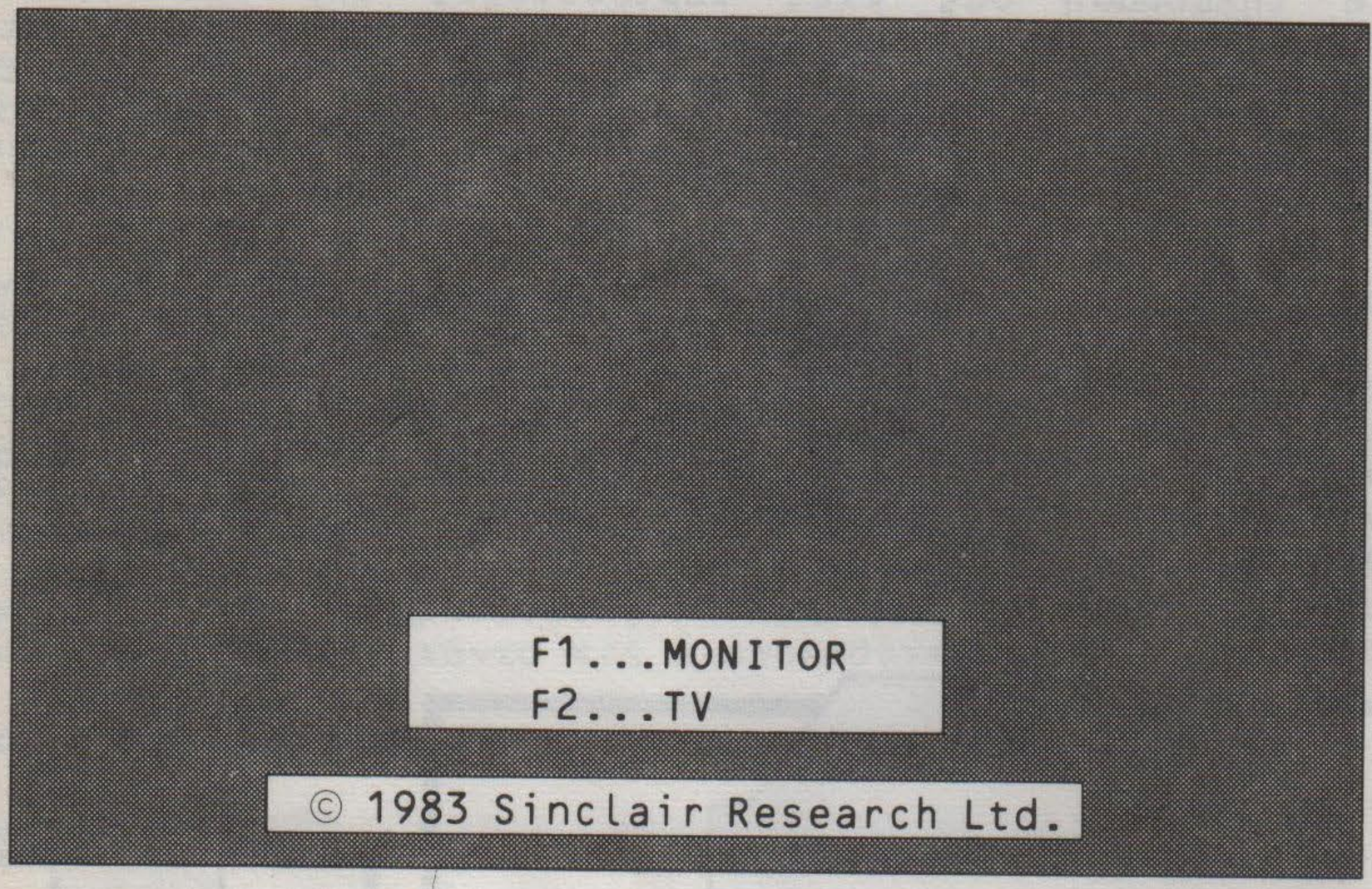



Αφού ο υπολογιστής λειτουργήσει για λίγο, το μέρος πάνω από τα Microdrives θα αρχίσει να ζεσταίνεται, αυτό είναι κανονικό. Το QL δεν έχει πλήκτρο που το ανοίγει ή το κλείνει, αλλά μπορεί να κλείσει αποσυνδέοντας το μετασχηματιστή. Να θυμάστε ότι όποιο πρόγραμμα ή δεδομένα είναι στη μηχανή θα χαθεί, γι' αυτό θα πρέπει να φυλαχθεί πρώτα σε μικροκασέτα. Για λεπτομέρειες για το πως θα γίνει αυτό, δείτε στον Οδηγό Αρχάριου και στις 'Εννοιες. Αν το QL δεν πρόκειται να χρησιμοποιηθεί για λίγο, θα πρέπει να βγάλετε το μετασχηματιστή από το ρεύμα.

## Η Ρύθμιση

Το QL μεταδίδει στην τηλεόραση περίπου στο κανάλι 36. Αν η συσκευή σας έχει αυτόματο ρύθμισμα, ρυθμίστε γύρω σ' αυτή την

περιοχή. Αν η τηλεόραση σας έχει κουμπιά που πατιούνται, διαλέξτε ένα αχρησιμοποίητο κουμπί και ρυθμίστε αυτό με τον υπολογιστή. Για να βρείτε πως θα γίνει, μπορεί να χρειασθεί να συμβουλευτείτε τον πωλητή σας ή τις οδηγίες της τηλεόρασης. 'Οταν τη ρυθμίσετε σωστά θα πρέπει να δείτε την οθόνη με το δικαίωμα copyright.




Οθόνη Copyright

Το QL δε χρησιμοποιεί τον ήχο της τηλεόρασης - έχει το δικό του εσωτερικό μεγάφωνο έτσι μπορείτε να κλείσετε τη φωνή αν θέλετε. 'Οταν η τηλεόραση έχει ρυθμιστεί, θα βλέπετε ένα τυχαία χρωματισμένο σχέδιο όποτε ανοίγετε ή επανατοποθετείτε (RESET) τον υπολογιστή. Αυτό γίνεται επειδή το QL εξετάζει τη μνήμη του προτού τη χρησιμοποιήσει. Το σχέδιο θα εξαφανιστεί μετά από μερικά δευτερόλεπτα για να αντικατασταθεί από την οθόνη με το σήμα του δικαιώματος copyright.

Αν δεν μπορείτε να έχετε καθόλου εικόνα, ελέγξτε αν η τηλεόραση σας μπορεί να δεχθεί τους κανονικούς σταθμούς. Αν μπορεί, δοκιμάστε τον υπολογιστή με μία άλλη τηλεόραση.

Αν έχετε εικόνα, αλλά έχει παράσιτα ή δεν είναι καθαρή, ελέγξτε αν κάνατε σωστή ρύθμιση. Είναι πιθανό να μπορείτε να πιάσετε τα σήματα του υπολογιστή σε περισσότερα από ένα μέρη. Ελέγξτε επίσης αν το καλώδιο κεραίας είναι σταθερά βαλμένο στην πρίζα, και αν χρησιμοποιείτε τη σωστή θέση στην τηλεόρασή σας, εφ' όσον έχει περισσότερες από μία.

Αν επιθυμείτε να χρησιμοποιήσετε ένα μόνιτορ αντί για τηλεόραση, οι συνδέσεις θα εξαρτηθούν από το αν είναι ασπρόμαυρο, λεπτομέρειες βρίσκονται στο μέρος ΕΝΝΟΙΕΣ κάτω από το κεφάλαιο αρχικό μόνιτορ.

Το QL χρειάζεται να γνωρίζει αν χρησιμοποιείτε τον υπολογιστή με μόνιτορ ή με τηλεόραση. Πατήστε:

F1 για το μόνιτορ, ή

F2 για την τηλεόραση

Το Microdrive 1 θα λειτουργήσει για λίγο. Το QL ψάχνει για να φορτώσει τα προγράμματα. Αυτό μπορεί να αγνοηθεί για την ώρα. Τελικά, ο υπολογιστής θα αρχίσει να λειτουργεί και θα εμφανίσει το δρομέα του, ένα φωτισμένο χρωματιστό τετράγωνο. 'Οταν ο δρομέας είναι ορατός το QL είναι έτοιμο να δεχθεί εντολές ή δεδομένα.

## Η χρήση του QL

### Το πληκτρολόγιο

Αντίθετα από τους προηγούμενους υπολογιστές Sinclair, δεν υπάρχει είσοδος δεσμευμένων λέξεων (keywords) με ένα μοναδικό πλήκτρο στον QL. Παρ' όλα αυτά, διάφορα πλήκτρα και ομάδες πλήκτρων έχουν μία ειδική έννοια:

### ENTER

Το πλήκτρο ENTER χρησιμοποείται για να δηλώσετε στον υπολογιστή ότι θέλετε να κάνει κάτι. Μπορεί να έχετε πληκτρολογήσει μια εντολή και να θέλετε ο υπολογιστής να την εκτελέσει, ή μπορεί να έχετε πληκτρολογήσει κάποια δεδομένα και να θέλετε να πείτε στον υπολογιστή ότι έχετε τελειώσει.

### SHIFT

Το πληκτρολόγιο έχει δύο πλήκτρα SHIFT. 'Εχουν ακριβώς την ίδια λειτουργία και είναι δύο, για ευχέρεια. Πατώντας το SHIFT και ένα αλφαβητικό πλήκτρο ταυτόχρονα θα παράγονται κεφαλαία γράμματα. Σε μη αλφαβητικά πλήκτρα το SHIFT θα προκαλέσει την παραγωγή του χαρακτήρα που εμφανίζεται στο πάνω μέρος του πλήκτρου. Π.χ.

SHIFT και 5 θα δώσει το %

### CAPSLOCK

Το πλήκτρο CAPSLOCK θα αναγκάσει το πληκτρολόγιο να παράγει πάντα χαρακτήρες του πάνω μέρους σε αλφαβητικά πλήκτρα ανεξάρτητα από το αν το SHIFT είναι πατημένο, αλλά τους κάτω χαρακτήρες σε όλα τα άλλα πλήκτα. Το αποτέλεσμα αυτό θα διατηρηθεί έως ότου να ξαναπατηθεί το CAPSLOCK.

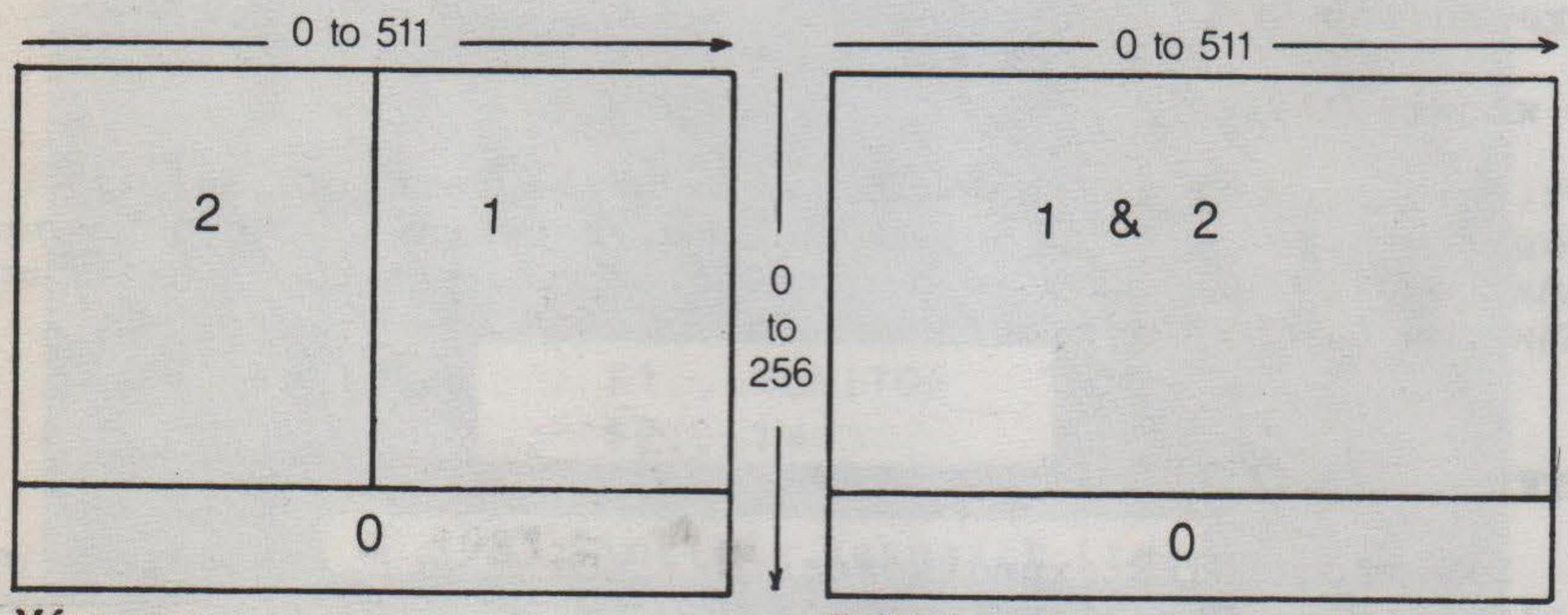
### DELETE

Το QL έχει δύο πλήκτρα διαγραφής. Πληκτρολογήστε λίγο κείμενο, πατήστε το πλήκτρο CTRL και μετά το δείκτη <-. Ο χαρακτήρας στα αριστερά του δείκτη θα εξαφανιστεί και ο δείκτης θα κινηθεί προς τα αριστερά. Τώρα πατήστε το πλήκτρο

<- μόνο του μερικές φορές για να τοποθετήσετε το δείκτη στο κέντρο του κειμένου που έχετε πληκτρολογήσει. Πατήστε το πλήκτρο CTRL ξανά, και αυτή τη φορά πατήστε το δείκτη ->. Ο δείκτης δε θα κινηθεί, ο χαρακτήρας πάνω στον οποίο ήταν θα διαγραφεί και ότι υπάρχει στα δεξιά θα κινηθεί για να γεμίσει το κενό.

**Η οθόνη**

Η οθόνη του QL μπορεί να διαιρεθεί σε διαφορετικές περιοχές ή "παράθυρα" κατά βούληση. Μόλις έχετε ανάψει ή επανατοποθετήσει τον υπολογιστή, και έχετε πατήσει το F1 ή το F2, η οθόνη θα δείχνει έτσι:




Μόνιτορ Τηλεόραση

Το μακρύ λεπτό παράθυρο στο κάτω μέρος χρησιμοποιείται για να εμφανίζει όλες τις εντολές που πληκτρολογείτε στον υπολογιστή.

'Οταν ο δρομέας είναι ορατός το QL είναι έτοιμο να δεχθεί εντολές ή δεδομένα. Αν ο υπολογιστής είναι απασχολημένος τότε ο δρομέας δεν είναι ορατός. Ο δρομέας θα κινείται κατά μήκος της γραμμής δείχνοντας που θα εμφανιστεί ο επόμενος χαρακτήρας που θα πληκτρολογηθεί.

Αν ποτέ ο υπολογιστής αποτύχει να αντιδράσει σωστά ή θέλετε να εξαναγκάσετε το πρόγραμμα να σταματήσει, πατήστε το πλήκτρο CTRL και συγχρόνως μετά τη μπάρα διαστήματος.

CTRL και SPACE

Το σύστημα τότε θα πρέπει να εμφανίσει το δρομέα του. Αν αυτό δε λειτουργήσει, βγάλτε από τη θέση τους όποιες μικροκασέτες υπάρχουν και πατήστε το RESET.

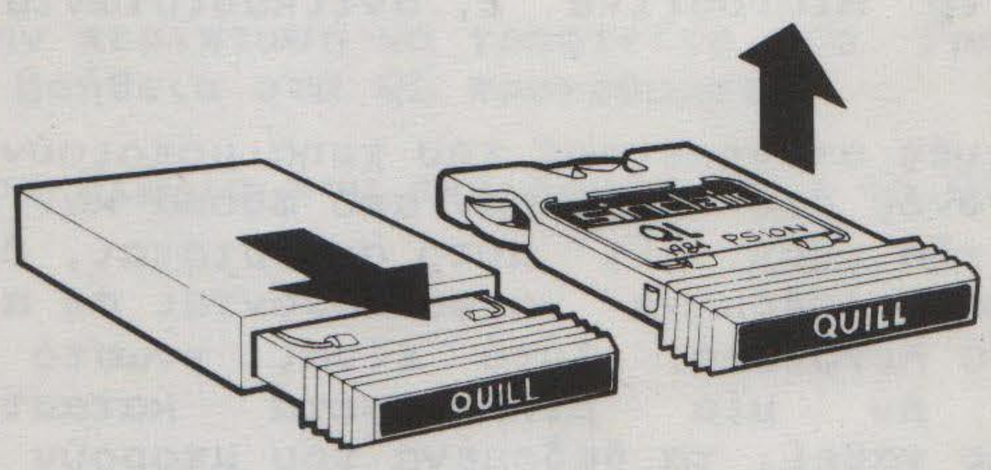
Εκτός αν ξέρετε BASIC, το παράθυρο εντολών θα περιέχει πιθανά τυχαίους χαρακτήρες, σαν αποτέλεσμα του πειραματισμού με το πληκτρολόγιο. Πατήστε το πλήκτρο ENTER για να δείτε τι θα κάνει ο υπολογιστής. Το πιο πιθανό είναι να εκτυπώσει στο παράθυρο εντολών, "bad line", για να δείξει ότι δεν καταλαβαίνει. 'Οτι έχετε πληκτρολογήσει θα επαναληφθεί, με το δρομέα τοποθετημένο μετά από αυτό, για να σας δώσει την ευκαιρία να το αλλάξετε και να προσπαθήσετε ξανά. Αν θελετε μπορείτε να το σβήσετε, χρησιμοποιόντας τα πλήκτρα διαγραφής.

Δεν μπορείτε ποτέ να χαλάσετε τον υπολογιστή, πληκτρολογώντας στο πληκτρολόγιο. Το χειρότερο που μπορείτε να κάνετε είναι να σβήσετε μία μικροκασέτα. Γι' αυτό πειραματιστείτε, είναι ο καλύτερος τρόπος για να μάθετε.

**Τα Microdrives**

Τα δύο Microdrives του QL ονομάζονται mdv1 (το αριστερό) και mdv2 (το δεξιό).

Οι μικροκασέτες πρέπει να τοποθετηθούν σωστά στο Microdrive. Κρατήστε τη μικροκασέτα από το τυλιγμένο πλαστικό χερούλι και βγάλτε την από το προστατευτικό της κάλυμα. Οι μικροκασέτες των προγραμμάτων του QL έχουν ένα όνομα σε ετικέτα, κρατήστε τη μικροκασέτα έτσι ώστε αυτό να είναι προς τα πάνω. Οι κενές μικροκασέτες είναι εφοδιασμένες με ξεχωριστές ετικέτες για κόλημα, και θα πρέπει να κρατιούνται με την εσοχή της ετικέτας προς τα πάνω.




Οι μικροκασέτες θα πρέπει να χρησιμοποιούνται πάντα με προσοχή. Δε θα πρέπει ΠΟΤΕ να ανοίγετε ή να κλείνετε τον QL, ή να τον ξανατοποθετείτε όταν υπάρχει μία μικροκασέτα σε οποιοδήποτε Microdrive. Προσέχετε όταν βάζετε ή βγάζετε τις μικροκασέτες, περιμένετε ως ότου τα φωτάκια των Microdrives έχουν σβήσει πριν αγγίξετε τη μικροκασέτα και να είστε προσεκτικοί μα και σταθεροί. Μην αγγίξετε ποτέ την ταινία στη μικροκασέτα και βάζετε πάντα τη μικροκασέτα στο προστατευτικό της κάλυμα όταν είναι έξω από το Microdrive.

Πριν να μπορέσει να χρησιμοποιηθεί μία άδεια μικροκασέτα, πρέπει να περάσει από μία φάση που λέγεται φορμάρισμα (formatting). Αυτό θα σβήσει οτιδήποτε έχει αποθηκευτεί πιο μπροστά, γι αυτό να βεβαιώνεστε πάντα για το περιεχόμενο των μικροκασετών και να βάζετε τις ανάλογες ετικέτες με τα ονόματα τους στις μικροκασέτες.

Πρώτα να βρείτε ένα όνομα για τη μικροκασέτα χρησιμοποιώντας όχι περισσότερους από δέκα χαρακτήρες. Με το QL αναμένο και ορθό το δρομέα που αναβοσβήνει τοποθετήστε τη μικροκασέτα που θα φορμαριστεί στο Microdrive 1 (δηλαδή αριστερά). Ας πούμε το όνομα της μικροκασέτας είναι "data". Τότε θα πληκτρολογούσατε:

FORMAT mdv1_

Μη συγχέετε το σύμβολο υπογράμμισης _ που χρησιμοποιείται παραπάνω με το σύμβολο μείον - επειδή και τα δύο είναι στο ίδιο πλήκτρο, ακριβώς πάνω και προς τα δεξιά του πλήκτρου P. Το σύμβολο υπογράμμισης είναι το πάνω σ' αυτό το πλήκτρο, και έτσι πρέπει να πατήσετε ένα από τα πλήκτρα SHIFT καθώς θα πατάτε το πλήκτρο του μείον.

Μετά πατήστε το πλήκτρο ENTER και θα ανάψει το αριστερό φωτάκι του Microdrive για τριάντα δευτερόλεπτα περίπου. Το QL θα εμφανίσει ένα μήνυμα στην οθόνη που θα σας λέει πόσος χώρος είναι διαθέσιμος σε εκείνη τη μικροκασέτα. Προσωρινά μπορείτε να το αγνοήσετε αυτό εξηγείται στο μέρος Δεσμευμένες λέξεις (κάτω από τη FORMAT).

Είναι καλή εξάσκηση να φορμάρετε μία τελείως καινούρια μικροκασέτα πολλές φορές, αυτό θα βοηθήσει την ταινία να κυλήσει απαλά, και μπορεί να βρείτε ότι περισσότεροι τομείς θα είναι διαθέσιμοι μετά από πολλά φορμαρίσματα.

Στο παραπάνω παράδειγμα, η μικροκασέτα θα μπορούσε επίσης να φορμαριστεί στο Microdrive 2, αντικαθιστώντας το mdv1 με το mdv2.

'Ολες οι συσκευές αποθήκευσης που χρησιμοποιούν ένα μαγνητικό μέσο είναι γεγονός ότι υποφέρουν από φθορά και οι μικροκασέτες δεν αποτελούν εξαίρεση. Γι' αυτό συνιστάται, όλα τα σπουδαία προγράμματα και δεδομένα να φυλάσσονται σε αντίγραφα σε δύο μικροκασέτες το λιγότερο. Αυτό είναι γνωστό σαν αντίγραφα υποστήριξης. Αν μία μικροκασέτα καταστραφεί και το περιεχόμενό της χαθεί, τα δεδομένα του μπορούν να ανακτηθούν από τη σχετική βοηθητική μικροκασέτα. Αν προσθέτετε συνεχώς δεδομένα σε μία μικροκασέτα, θα πρέπει να αντιγράφονται συχνά. 'Οτι έχει γίνει αφότου έγινε η τελευταία αντιγραφή θα χαθεί αν η κύρια μικροκασέτα καταστραφεί.

Οι μικροκασέτες μπορούν να αντιγραφούν χρησιμοποιώντας την εντολή COPY της SuperBASIC τα σχετικά μ' αυτό υπάρχουν στον Οδηγό του Αρχάριου και στο τμήμα Δεσμευμένες Λέξεις αυτού του βιβλίου. Οι μικροκασέτες των προγραμμάτων του QL (QL Quill, QL Abacus, QL Archive και QL Easel) έχουν ένα ενσωματωμένο πρόγραμμα αντιγραφής. (Σχετικά μ' αυτό δείτε παρακάτω στο κεφάλαιο "Εισαγωγή στα QL Προγράμματα"). Τα δεδομένα που φυλάχθηκαν μέσα από τα QL προγράμματα μπορούν να αντιγραφούν απλά γράφοντάς τα δύο φορές, σε ξεχωριστές μικροκασέτες.

**Αρχίζοντας εργασία**

Βασικά υπάρχουν δύο τρόποι με τους οποίους μπορείτε να χρησιμοποιήσετε τον υπολογιστή: είτε χρησιμοποιώντας έτοιμα προγράμματα, σαν αυτά που είναι εφοδιασμένο το QL, είτε γράφοντας τα δικά σας προγράμματα στη γλώσσα SuperBASIC του QL.

Αν είστε καινούργιος στους υπολογιστές θα πρέπει να διαβάσετε πρώτα τον Οδηγό του Αρχάριου.

Για να χρησιμοποιήσετε τα QL προγράμματα, διαβάστε πρώτα το κεφάλαιο 2 αυτής της εισαγωγής και ειδικά το σχετικό μέρος για

το πρόγραμμα που σας ενδιαφέρει περισσότερο.

Αν είστε εξοικειωμένοι με τον προγραμματισμό BASIC σε άλλους υπολογιστές θα μπορούσατε να πάτε στο κεφάλαιο 8 "Από τη BASIC στη SuperBASIC" στον Οδηγό του Αρχάριου. Το κεφάλαιο αυτό περιγράφει τις κύριες διαφορές ανάμεσα στη BASIC που μπορεί να ξέρετε και τη SuperBASIC που χρησιμοποιείται στο QL. Εναλλακτικά, αν αισθάνεστε σίγουροι για τον εαυτό σας και έχετε ένα πρόγραμμα που θα θέλατε να δοκιμάσετε, μπορείτε απλά να το εισάγετε στο QL και να μάθετε με την εμπειρία. Το μέρος Δεσμευμένες Λέξεις αυτού του βιβλίου καταγράφει τις λέξεις BASIC που το QL αναγνωρίζει, και μαζί με το μέρος 'Εννοιες θα βοηθήσει στα προβλήματα που μπορεί να συναντήσετε.

**Αν έχετε ένα πρόβλημα**

Αν έχετε ένα πρόβλημα χρησιμοποιώντας το QL σας ή τα QL προγράμματα τότε:

1. Ανατρέξτε στο σχετικό μέρος στον Οδηγό Χρήσης του QL.

2. Εξετάστε την περίπτωση να γραφτείτε στο Γραφείο Χρηστών του QL για βοήθεια στα QL προγράμματα.

3. Ανατρέξτε σε βιβλία που εκδίδονται για το QL.

ΚΕΦΑΛΑΙΟ 2

# ΕΙΣΑΓΩΓΗ ΣΤΑ ΠΡΟΓΡΑΜΜΑΤΑ ΤΟΥ QL

Η εισαγωγή αυτή εξηγεί μερικά από τα βασικά στοιχεία για τα προγράμματα που δίνονται μαζί με το Sinclair QL. Περιγράφει τα χαρακτηριστικά που είναι κοινά και για τα τέσσερα προγράμματα.

Τα ιδιαίτερα εγχειρίδια περιγράφουν καθένα από τα τέσσερα προγράμματα με περισσότερες λεπτομέρειες, δίνοντας περισσότερη σημασία σ' αυτά τα χαρακτηριστικά που είναι ειδικά για το καθένα. Μην τα διαβάσετε απλώς - δοκιμάστε τα παραδείγματα και πειραματιστείτε με κάθε νέα ιδέα. Ο μόνος τρόπος για να καταλάβετε οποιοδήποτε υπολογιστή και τα προγράμματα που είναι γραμμένα γι' αυτόν είναι να τα χρησιμοποιήσετε. Μη φοβάστε να δοκιμάσετε οτιδήποτε - δεν μπορείτε ποτέ να χαλάσετε τον υπολογιστή πληκτρολογώντας οτιδήποτε στο πληκτρολόγιο.

Βεβαιωθείτε ότι δοκιμάζετε όλα τα παραδείγματα που απεικονίζουν και εξηγούν πολλά πράγματα που δεν περιγράφονται πουθενά αλλού.

Οι πιο τεχνικές πλευρές της Εισόδου/Εξόδου και ανταλλαγής πληροφοριών ανάμεσα στα τέσσερα προγράμματα περιγράφονται σε ξεχωριστό παράρτημα.

**Γενικά για τα προγράμματα**

Τα QL Quill, Abacus, Archive και Easel είναι τέσσερεις ιδιαίτερα δυνατές εφαρμογές προγράμματα που έχουν σχεδιαστεί προσεκτικά για να είναι εύκολα στη χρήση.

Σε κάθε πρόγραμμα οι επιλογές που θα χρειαστείτε πιο συχνά είναι διαθέσιμες αμέσως. Αυτές οι επιλογές καταγράφονται στην οθόνη και κάθε εκλογή που κάνετε, επιβεβαιώνεται αμέσως.

Σε πολλές περιπτώσεις, τα προγράμματα θα προτείνουν μία κατάλληλη απάντηση όταν ρωτούν για πληροφορίες. Αν επιθυμείτε να δεχθείτε την πρόταση απλά πατήστε το ENTER, αλλιώς, αρχίστε να πληκτρολογείτε τη δική σας απάντηση και η πρόταση του προγράμματος θα εξαφανιστεί.

'Οποτε το πρόγραμμα σας περιμένει να πληκτρολογήσετε έναν αριθμό, έχετε την επιλογή να εισάγετε μία αριθμητική έκφραση αντί για τον αριθμό. Για παράδειγμα, θα μπορούσατε να εισάγετε 1562 + 379 αντί του 1941, ώστε να μη χρειάζεται να το υπολογίσετε προηγούμενα. Μία πιο λεπτομερής περιγραφή των αριθμητικών ικανοτήτων περιλαμβάνεται στο παράρτημα.

Οι τέσσερεις εφαρμογές έχουν σχεδιαστεί σαν ένα ολοκληρωμένο πακέτο, δηλαδή:

* 'Εχουν κοινή εμφάνιση
* Παρόμοιες ενέργειες ελέγχονται με παρόμοιο τρόπο
* 'Εχουν κοινές εντολές
* Πληροφορίες μπορούν να μεταφερθούν ανάμεσα τους πολύ εύκολα.

**Η εμφάνιση**

'Ολα τα QL προγράμματα χρησιμοποιούν το πάνω μέρος της οθόνης, που ονομάζεται περιοχή ελέγχου, για την καταγραφή των επιλογών σας, λέγοντας σας τι αναμένεται και επιβεβαιώνοντας τις εκλογές που κάνετε. Αυτή η περιοχή της οθόνης χρησιμοποιεί ένα ομοιότυπο σχέδιο για αυτές τις επιλογές που είναι ίδιες σε όλα τα προγράμματα.

Η κεντρική περιοχή της οθόνης δείχνει τις πληροφορίες με τις οποίες εργάζεστε - για παράδειγμα, το κείμενο ενός εγγράφου, τα περιεχόμενα ενός ευρετηρίου καρτών, μία γραφική παράσταση ή μία οικονομική πρόβλεψη. Αυτές οι πληροφορίες εμφανίζονται με τον πιο κατάλληλο τρόπο για τη συγκεκριμένη εφαρμογή.

Οι λίγες γραμμές στο κάτω μέρος της οθόνης περιέχουν τη γραμμή εισόδου - όπου, για παράδειγμα, εμφανίζονται οι εντολές που πληκτρολογείτε. Επίσης περιλαμβάνει την περιοχή κατάστασης που αναφέρει την τωρινή κατάσταση της εργασίας σας. Αυτή περιέχει διάφορες πληροφορίες όπως το όνομα των δεδομένων ή του εγγράφου πάνω στο οποίο εργάζεσθε και πόση είναι ακόμη η ελεύθερη μνήμη.

**Φορτώνοντας**

Δεν θα πρέπει ποτέ να χρησιμοποιήσετε πρωτότυπα προγράμματα μικροκασέτες, παρά μόνο για να κάνετε ένα αντίγραφο σε μία νέα μικροκασέτα. Χρησιμοποιήστε για καθημερινή χρήση μόνο το αντίγραφο. Πληροφορίες για να κάνετε ένα αντίγραφο περιέχονται παρακάτω σ' αυτή την εισαγωγή στην παράγραφο με τον τίτλο "Microdrives".

'Ολα τα QL προγράμματα φορτώνονται παρόμοια. Υπάρχουν δύο τρόποι για να γίνει αυτό.

Χωρίς μικροκασέτες στα Microdrives, πατήστε το RESET ή κλείστε το QL και μετά ξανανοίξτε το. Βάλτε το αντίγραφο της μικροκασέτας του προγράμματος στην αριστερή μονάδα των Microdrives, και μετά πατήστε το F1 ή το F2 όπως σας ζητά το QL. Το Microdrive 1 θα λειτουργήσει αυτόματα, και μετά από μία μικρή παύση, θα εμφανιστεί μία έκθεση τίτλων στην οθόνη για να επιβεβαιώσει ότι το πρόγραμμα φορτώνεται. 'Οταν το πρόγραμμα είναι στον υπολογιστή, θα αρχίσει μόνο του. Αφήστε τη μικροκασέτα με το πρόγραμμα στη θέση της όλη την ώρα που τρέχει το πρόγραμμα, μια και θα χρειαστεί να φορτώνει επιπλέον πληροφορίες κάθε στιγμή.

'Οταν έχετε προχωρήσει αρκετά στις εργασίες των προγραμμάτων, για παράδειγμα, χρησιμοποιώντας έναν εκτυπωτή ή ένα δίκτυο, θα βρείτε ότι μερικές φορές διάφορες ενέργειες θα χρειαστούν να γίνουν στον υπολογιστή πριν το πρόγραμμα τρέξει. Δεν μπορείτε να κλείσετε ή να επανατοποθετήσετε τον QL αυτή τη στιγμή γιατί η εργασία σας δεν θα τελείωνε. 'Ετσι θα πρέπει να φορτώσετε το πρόγραμμα ξανά στην αριστερή μονάδα, χρησιμοποιώντας την εντολή:

```
lrun mdv1_boot
```

Μετά πατήστε το πλήκτρο ENTER και το φόρτωμα θα προχωρήσει όπως πριν.

Πλήκτρα λειτουργιών

Τρία από τα πέντε πλήκτρα λειτουργίας χρησιμοποιούνται με τον ίδιο τρόπο σε καθένα από τα QL προγράμματα. Αυτά είναι:

| ΠΛΗΚΤΡΟ | ΕΝΕΡΓΕΙΑ |
|---|---|
| F1 | Επίκληση για βοήθεια |
| F2 | Μετακίνηση ή διόρθωμα της πάνω περιοχής μηνυμάτων |
| F3 | Κλήση των εντολών για επιλογή |

Τα υπόλοιπα δύο πλήκτρα λειτουργίας χρησιμοποιούνται για ενέργειες που είναι ειδικές για καθένα από τα προγράμματα.

**Βοήθεια**

Η πρώτη επιλογή, που εμφανίζεται στο πάνω αριστερό μέρος της περιοχής ελέγχου σε όλα τα QL προγράμματα, δείχνει ότι μπορείτε να ζητήσετε βοήθεια πατώντας το πλήκτρο λειτουργίας 1 (F1). Ανεξάρτητα από οποιεσδήποτε άλλες αλλαγές στην εμφάνιση της περιοχής ελέγχου, η επιλογή βοήθειας θα φαίνεται πάντα. Αυτό σημαίνει πώς ότι και να κάνετε υπάρχει πάντα ευκολία βοήθειας.

'Οταν πατήσετε το F1 θα γίνει μία μικρή παύση πριν η οθόνη αλλάξει για να δείξει πληροφορίες σχετικές με αυτό που κάνετε.

Επίσης, θα δείτε συνήθως ότι υπάρχει ένας κατάλογος από θέματα που εμφανίζονται κατά μήκος του κάτω μέρους της οθόνης για τα οποία είναι διαθέσιμη περισσότερη βοήθεια. Μπορείτε να ζητήσετε περισσότερες πληροφορίες για κάθε ένα από αυτά τα θέματα πληκτρολογώντας το όνομα του και πατώντας το ENTER. Δε χρειάζεται να πληκτρολογήσετε ολόκληρο το όνομα, απλώς πληκτρολογήστε τα λίγα πρώτα γράμματα που είναι αρκετά για να το ξεχωρίσει το πρόγραμμα από οποιοδήποτε άλλο από τα θέματα στον κατάλογο.

Αφού πληκτρολογήσετε το θέμα της εκλογής σας και πατήσετε το ENTER, εμφανίζονται περισσότερες πληροφορίες για το θέμα εκείνο και μπορεί επίσης να εμφανιστεί ακόμα, ένας κατάλογος από υπομενού. Μπορείτε να διαλέξετε ένα από αυτά τα υπομενού πληκτρολογώντας τα πρώτα γράμματα του ονόματος του και

πατώντας το ENTER, όπως περιγράφτηκε παραπάνω. Μπορείτε να συνεχίσετε αυτή τη διαδικασία ώσπου να σας ειπωθεί ότι δεν υπάρχει άλλη διαθέσιμη πληροφορία.

Μπορείτε πάντα να πάτε πίσω στην προηγούμενη οθόνη πληροφοριών απλά πατώντας το ENTER, χωρίς να πληκτρολογήσετε κάποιο προηγούμενο κείμενο. Πατώντας πάλι το ENTER θα σας γυρίσει πίσω στο σημείο όπου ζητήσατε βοήθεια. Τότε θα έχετε αφήσει τη βοήθεια και θα γυρίσετε στην ίδια ακριβώς κατάσταση πριν ζητήσετε βοήθεια.

**Διαφυγή**

Πατώντας το ESC θα αφήσει τη βοήθεια αμέσως, από οποιοδήποτε επίπεδο, και θα ξαναγυρίσετε στην ίδια ακριβώς κατάσταση πριν να ζητήσετε βοήθεια.

Η βοήθεια είναι πάντα διαθέσιμη σε καθένα από τα QL προγράμματα εφόσον η μικροκασέτα προγράμματος είναι στη θέση 1. 'Οποτε δεν είστε σίγουροι για το τι θα πρέπει να κάνετε, πατήστε απλώς το πλήκτρο βοήθειας (F1) - ακόμα κι αν είστε στο μέσο της πληκτρολόγησης αριθμών ή κειμένου σαν μέρος μιας εντολής. Δε θα αρχίσετε πάντα στο ίδιο σημείο στη βοήθεια, αλλά θα σας δειχθούν οι πιο σχετικές πληροφορίες σε αυτό που κάνετε όταν πατήσετε το πλήκτρο βοήθειας. 'Οταν έχετε βρει την πληροφορία που χρειάζεστε και αφήσετε τη βοήθεια (πατώντας το ESC ή το ENTER) το πρόγραμμα θα σας γυρίζει πάντα στο ίδιο ακριβώς σημείο από το οποίο αρχίσατε.

Το κείμενο που δείχνεται από τη βοήθεια φυλάσσεται στη μικροκασέτα προγράμματος και διαβάζεται στο QL όποτε χρειάζεται. Γι' αυτό, η μικροκασέτα προγράμματος θα πρέπει να είναι συνέχεια στη θέση 1 όταν χρησιμοποιείτε το πρόγραμμα.

**Τα Μηνύματα**

Η περιοχή ελέγχου, εκτός από το ότι δείχνει τις επιλογές σας, φωτίζει την εκλογή σας και, όταν είναι αναγκαίο, σας προτείνει τι να κάνετε. Μπορείτε να σταματήσετε την επίδειξη της περιοχής ελέγχου πατώντας το πλήκτρο λειτουργίας 2 (F2). 'Οταν το κάνετε αυτό, το κενό που παραμένει στην οθόνη προστίθεται στην κεντρική περιοχή, ώστε να έχετε περισσότερο χώρο για να εμφανίσετε την εργασία σας.

Μπορείτε να επανακτήσετε την εμφάνιση της περιοχής ελέγχου σε οποιαδήποτε στιγμή πατώντας ξανά το F2.

Στα QL προγράμματα, μπορείτε πάντα να χρησιμοποιήσετε το πλήκτρο ESC για να ακυρώσετε την τωρινή ενέργεια κίνησης, ή να αφήσετε μια συγκεκριμένη ακολουθία ενεργειών. 'Εχουμε ήδη δει πως μπορείτε να χρησιμοποιήσετε το πλήκτρο ESC για να αφήσετε τη βοήθεια, από οποιοδήποτε επίπεδο. Μπορείτε επίσης να χρησιμοποιήσετε το ESC για να ακυρώσετε αριθμούς ή κείμενο που έχετε πληκτρολογήσει στη σειρά εισαγωγής, ή να αποβάλλετε μια ημιτελή εντολή.

**Διορθωτής γραμμής**

Μπορείτε να χρησιμοποιείτε το διορθωτή γραμμής για να αλλάξετε

ή να διορθώσετε μια σειρά κειμένου που έχετε πληκτρολογήσει. Όλα τα QL προγράμματα χρησιμοποιούν τον ίδιο διορθωτή γραμμής, αλλά κάθε πρόγραμμα το χρησιμοποιεί με τον πιο κατάλληλο τρόπο για εκείνη την εφαρμογή. Στο QL Quill, που είναι το ίδιο ένας πολύ ειδικευμένος διορθωτής χρησιμοποιείτε το διορθωτή γραμμής για συγκεκριμένους εξειδικευμένους σκοπούς, όπως για τη διόρθωση του κειμένου που χρησιμοποιήθηκε στις εντολές. Το QL Archive χρησιμοποιεί το διορθωτή γραμμών εκτεταμένα για τη διόρθωση προγραμμάτων βάσης δεδομένων.

Δεν μπορείτε, γενικά, να διορθώσετε κείμενο που εμφανίζεται από το πρόγραμμα (όπως τη λέξη εντολής που εμφανίζεται όταν επιλέγετε μία εντολή). Μπορείτε να διορθώνετε ότι έχετε πληκτρολογήσει εσείς οι ίδιοι, ή το κείμενο που δίνει το πρόγραμμα σαν μια προτεινόμενη απάντηση σε μία επιλογή.

Ο διορθωτής γραμμής χρησιμοποιεί τα τέσσερα πλήκτρα με τα βέλη (του δρομέα) μαζί με τα πλήκτρα CTRL και SHIFT.

| Πλήκτρα | Ενέργεια |
|---|---|
| <- | Κινεί ένα χαρακτήρα προς τα αριστερά |
| -> | Κινεί ένα χαρακτήρα προς τα δεξιά |
| ↑ | Κινεί προς τα πάνω μία σειρά |
| ↓ | Κινεί προς τα κάτω μία σειρά |
| SHIFT & <- | Κινεί μία λέξη προς τα αριστερά |
| SHIFT & -> | Κινεί μία λέξη προς τα δεξιά |
| SHIFT & ↑ | Κινεί προς τα πάνω μία παράγραφο |
| SHIFT & ↓ | Κινεί προς τα κάτω μία παράγραφο |
| CTRL & <- | Διαγράφει το χαρακτήρα στα αριστερά του δρομέα |
| CTRL & -> | Διαγράφει τον χαρακτήρα πάνω στον οποίο είναι ο δρομέας |
| CTRL & ↑ | Διαγράφει τη γραμμή στα αριστερά του δρομέα |
| CTRL & ↓ | Διαγράφει τη σειρά στα δεξιά του δρομέα, περιλαμβανομένου του χαρακτήρα πάνω στον οποίο βρίσκεται ο δρομέας |
| SHIFT & CTRL & <- | Διαγράφει τη λέξη στα αριστερά του δρομέα |
| SHIFT & CTRL & -> | Διαγράφει τη λέξη στα δεξιά του δρομέα |

Το σύμβολο & που χρησιμοποιείται παραπάνω δηλώνει ότι το πρώτο πλήκτρο θα πρέπει να είναι πατημένο καθώς πατιέται το δεύτερο.

'Οταν το SHIFT και το CTRL χρησιμοποιούνται μαζί, να τα πατάτε και τα δύο πριν πατήσετε το πλήκτρο του δρομέα. Ο πειραματισμός είναι ο καλύτερος τρόπος για να εξοικειωθείτε με το διορθωτή γραμμής: εισάγετε κάποιο κείμενο στο Quill και δοκιμάστε τους διάφορους συνδυασμούς.

## — Τα Microdrives

Πριν χρησιμοποιήσετε κάποιο από τα QL προγράμματα θα πρέπει να κάνετε τουλάχιστον ένα αντίγραφο σε μία νέα μικροκασέτα και να χρησιμοποιείτε μόνο αυτό το αντίγραφο. Κρατείστε το πρωτότυπο πρόγραμμα στη μικροκασέτα ασφαλές, και χρησιμοποιήστε το μόνο για να κάνετε άλλα αντίγραφα. 'Ετσι αν συμβούν κάποια ατυχήματα δε θα προκαλέσουν τελείως χάσιμο των προγραμμάτων σας.

Κάθε QL μικροκασέτα προγράμματος έχει ένα πρόγραμμα (ρουτίνα) αντιγραφής που χρησιμοποιείται, ως ακολούθως:

Τοποθετήστε το πρωτότυπο στη θέση 2 (δεξιά θέση). Τοποθετήστε τη κενή μικροκασέτα, ή μία που να περιέχει στοιχεία που δεν επιθυμείτε να κρατήσετε στη θέση 1.

Πληκτρολογήστε αυτή την εντολή:

```
"lrun" mdv2_clone
```

Πατήστε το πλήκτρο ENTER. Το QL θα εμφανίσει αυτό το μήνυμα:

```
FORMAT mdv1_type space to continue
```

'Οταν πατήσετε το πλήκτρο διαστήματος, οτιδήποτε υπάρχει στη μικροκασέτα της θέσης 1 θα σβηστεί, γι' αυτό σιγουρευτείτε ότι δε θέλετε να κρατήσετε κάτι σε αυτή τη μικροκασέτα προτού προχωρήσετε.

Πατήστε το πλήκτρο διαστήματος. Ο υπολογιστής θα φορμάρει τη μικροκασέτα και μετά θα αντιγράψει το πρόγραμμα σε κομμάτια, εμφανίζοντας το όνομα του καθενός κομματιού.

Περιμένετε έως ότου σβήσουν και τα δύο φωτάκια των microdrives πριν μετακινήσετε το πρωτότυπο.

### Χρήση των microdrives

Η μικροκασέτα προγράμματος πάντα απασχολεί το αριστερό microdrive (θέση 1). Δε θα πρέπει να μετακινήσετε τη μικροκασέτα από αυτή τη θέση ως ότου να τελειώσετε τη χρήση του προγράμματος, εκτός από τις περιπτώσεις που ειδικά σημειώνονται στα ιδιαίτερα κομμάτια του προγράμματος.

Χρησιμοποιήστε μία μικροκασέτα στη θέση 2 ή σε πρόσθετες microdrives - για την αποθήκευση πληροφοριών π.χ. Quill έγγραφα, αρχείο δεδομένων του Archive κλπ.

### Ονόματα Αρχείων

Τα microdrives μπορούν να χρησιμοποιηθούν για να αποθηκεύσουμε

πληροφορίες μέσα από ένα πρόγραμμα για μελλοντική χρήση. Αυτές οι πληροφορίες είναι γνωστές σαν ένα αρχείο στο οποίο πρέπει να του δοθεί ένα όνομα αρχείου για να το ξεχωρίζετε από άλλα αρχεία στη ίδια μικροκασέτα. Πρέπει να σκεφτείτε ένα όνομα αρχείου όχι μεγαλύτερο από οκτώ χαρακτήρες, χωρίς κενό σ' αυτό. θα είναι χρήσιμο αργότερα αν το όνομα κατά κάποιο τρόπο σημειώνει τα περιεχόμενα του αρχείου για παράδειγμα το "sales" είναι ένα καλύτερο όνομα για ένα αρχείο με αριθμούς πωλήσεων παρά το "fred". Οι λειτουργίες αποθήκευση και φόρτωμα αρχείου θα χρησιμοποιήσουν το microdrive 2 εκτός αν τους δοθεί ένας διαφορετικός αριθμός θέσης.

Ο απλούστερος τρόπος για να απαντήσετε σε μία απαίτηση για ένα όνομα αρχείου, είναι απλά να πληκτρολογήσετε το όνομα μόνο του, για παράδειγμα:

sales

Αν θέλατε να χρησιμοποιήσετε τη μονάδα 3, θα πληκτρολογούσατε:

mdv3_sales

Σημειώστε το χαρακτήρα υπογράμμισης που χωρίζει τα μέρη του ονόματος αρχείου.

Υπάρχει ένα τρίτο ουσιαστικό μέρος στα ονόματα αρχείων που δε βλέπετε συχνά, γιατί προστίθεται αυτόματα από το πρόγραμμα. Αυτό είναι μια επέκταση, μήκους τριών χαρακτήρων, που αναγνωρίζει ποιο πρόγραμμα φύλαξε το αρχείο. Αυτό είναι απαραίτητο για να αποφύγετε να φορτώσετε ένα ακατάλληλο αρχείο μέσα σε ένα πρόγραμμα. Οι επεκτάσεις που χρησιμοποιούνται είναι:

| | |
|---|---|
| QUILL | _doc |
| ABACUS | _aba |
| EASEL | _grf |
| ARCHIVE | _prg για προγράμματα |
| ARCHIVE | _dbf για αρχείο δεδομένων |

Αν θέλετε να μεταφέρετε πληροφορίες ανάμεσα σε προγράμματα, ένα ειδικό αρχείο ενεργοποιείται με την επέκταση _exp (για έξοδο). 'Ολα τα προγράμματα θα αναγνωρίσουν αυτά τα αρχεία. Περισσότερες πληροφορίες σε αυτή τη διάρκεια περιέχονται στο Συμπλήρωμα στη παράγραφο Πληροφορίες.

**Καταγράφοντας αρχεία σε μία μικροκασέτα**

Σε όλα τα προγράμματα, εκτός από το QL Archive, μπορείτε να απαιτήσετε έναν πίνακα περιεχομένων των ονομάτων όλων των αρχείων που υπάρχουν σε μία μικροκασέτα, οποιαδήποτε στιγμή μία εντολή χρειάζεται ένα όνομα αρχείου. Αυτό είναι πολύ χρήσιμο αν δεν είστε σίγουροι ότι θυμάστε το ακριβές όνομα που δώσατε στο αρχείο όταν το αποθηκεύσατε πρώτη φορά.

Κάθε φορά που το πρόγραμμα σας περιμένει να πληκτρολογήσετε ένα τέτοιο όνομα, έχετε τις ακόλουθες επιλογές:

- πατάτε το ENTER για να δεχθείτε οποιοδήποτε όνομα προτείνει το πρόγραμμα

- πληκτρολογείτε το όνομα αρχείου, και μετά πατάτε το ENTER

- πατήστε το ? για έναν πίνακα περιεχομένων των αρχείων της μικροκασέτας στη θέση 2.

Αν πληκτρολογήστε ένα ερωτηματικό αντί για ένα όνομα αρχείου, το πρόγραμμα σας δίνει τη λέξη "mdv2_" υποδεικνύοντας ότι θα έπρεπε να εμφανίζει τα αρχεία της μικροκασέτας στη θέση 2. Μπορείτε να δεχθείτε αυτή την πρόταση πατώντας το ENTER, ή μπορείτε να διορθώσετε την ένδειξη της μονάδας να αναφερθεί σε μία διαφορετική θέση (π.χ. mdv1_) και μετά πατήστε το ENTER - για έναν πίνακα περιεχομένων της άλλης μικροκασέτας.

'Οταν ο πίνακας περιεχομένων είναι τελειωμένος, το πρόγραμμα ξανά σας ζητά να πληκτρολογήσετε το όνομα αρχείου.

Το ARCHIVE δε χρησιμοποιεί αυτήν τη μέθοδο - που δεν είναι κατάλληλη στη γλώσσα προγραμματισμού που χρησιμοποιεί το ARCHIVE. Στο ARCHIVE υπάρχει η εντολή - dir - που εμφανίζει τα αρχεία. Σας επιτρέπει να πληκτρολογήσετε "mdv1", "mdv2", κτλ, για να προσδιορίσετε τη θέση της μικροκασέτας για την οποία χρειάζεται ο κατάλογος των αρχείων.

**ΤΟ ΔΙΚΤΥΟ**

Μπορούν να σταλούν δεδομένα μέσα από το δίκτυο του QL σε άλλα QL, χρησιμοποιώντας τις εντολές φόρτωσης και αποθήκευσης. Πριν φορτώσετε ένα QL πρόγραμμα, κάθε μικρουπολογιστής που πρόκειται να πάρει μέρος στο δίκτυο θα πρέπει να του δοθεί ένας μοναδικός αριθμός. Ανοίξτε τον μικρουπολογιστή, αλλά μην εισάγετε μία μικροκασέτα προγράμματος, πατήστε το F1 για το μόνιτορ ή το F2 για την τηλεόραση όταν ερωτηθείτε. Αν ο επιλεγμένος αριθμός σταθμού είναι 5, θα πρέπει να πληκτρολογήσετε την ακόλουθη εντολή:

net

Μετά πατήστε το πλήκτρο ENTER. Μπορείτε τώρα να τοποθετήσετε τη μικροκασέτα προγράμματος στην αριστερή θέση μικροκασέτας και να το φορτώσετε χρησιμοποιώντας την εντολή:

"1run"mdv1_boot

ακολουθούμενη από το ENTER ως συνήθως.
'Οταν το πρόγραμμα τρέχει, μπορεί να δεχθείτε δεδομένα που να στέλνονται μέσα από το δίκτυο επιλέγοντας την εντολή LOAD με τον κανονικό τρόπο. Αν τα δεδομένα έχουν σταλεί από το σταθμό 12, θα εισάγετε τα ακόλουθα αντί από ένα όνομα αρχείου:

_neti_12

Αυτό πρέπει να γίνει πριν ο σταθμός 12 αρχίσει να στέλνει.

Για να στείλετε δεδομένα, διαλέξτε την εντολή SAVE. Υποθέτοντας ότι στέλνατε στο σταθμό 23, θα εισάγετε τα ακόλουθα αντί για ένα όνομα αρχείου:

_neto_23

Ο σταθμός 23 θα πρέπει να είναι έτοιμος να δεχτεί πριν πατήσετε το ENTER.

# QL

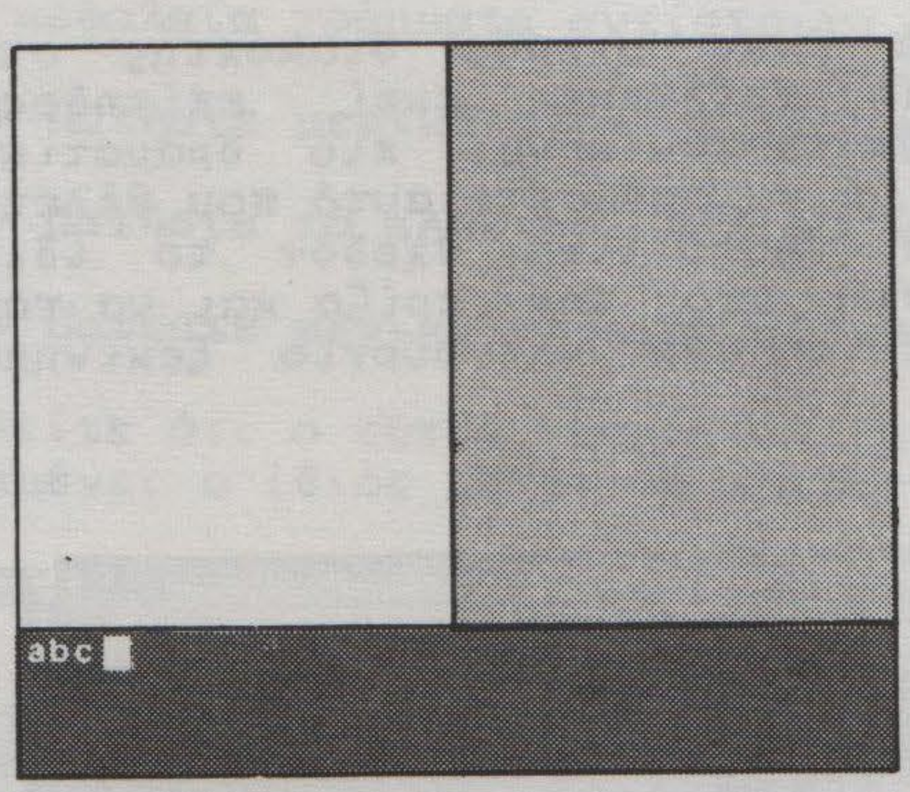
## Οδηγός Αρχαρίου

# ΚΕΦΑΛΑΙΟ 1

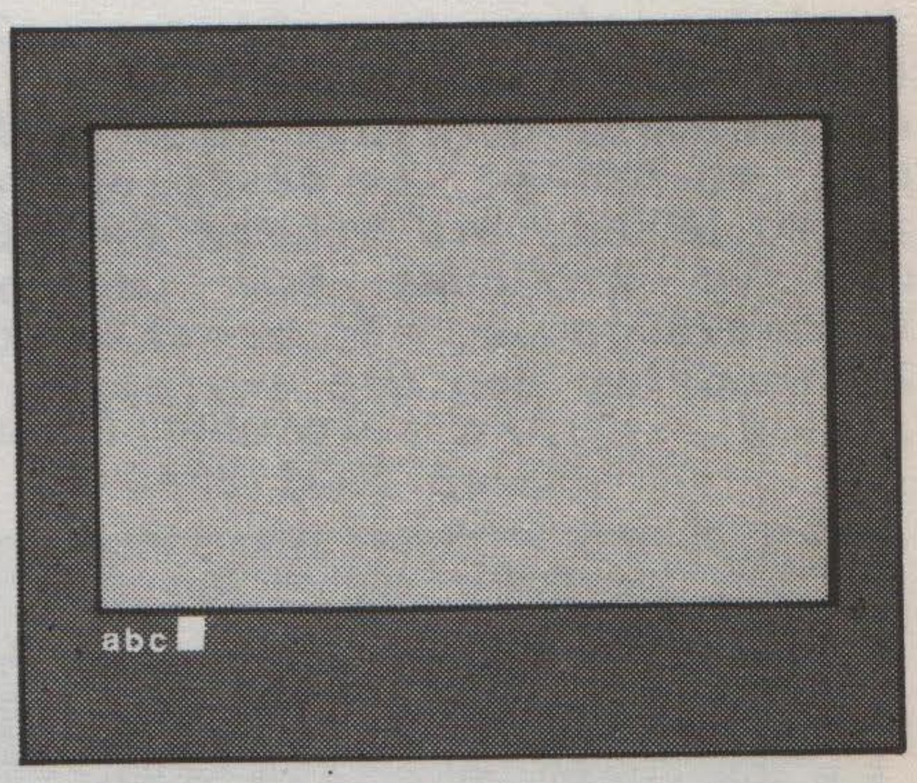
# ΑΡΧΙΖΟΝΤΑΣ ΥΠΟΛΟΓΙΣΜΟΥΣ

## Η οθόνη

Το QL σας πρέπει να είναι συνδεδεμένο με ένα μόνιτορ ή με μία τηλεόραση και να είναι συνδεδεμένο στο ρεύμα. Πατήστε μερικά πλήκτρα, π.χ. a, b, c και η οθόνη πρέπει να είναι όπως φαίνεται στις παρακάτω εικόνες. Το μικρό τετραγωνάκι που αναβοσβήνει λέγεται δρομέας.




Μόνιτορ




Τηλεόραση

Αν η οθόνη σας δεν είναι έτσι τότε διαβάστε το κεφάλαιο με τίτλο "ΕΙΣΑΓΩΓΗ". Αυτό θα σας βοηθήσει να λύσετε όλες σας τις απορίες.

## Το πληκτρολόγιο

Το QL είναι ένας ενέλικτος και ισχυρός υπολογιστής κι έτσι υπάρχουν μερικά χαρακτηριστικά στο πληκτρολόγιο τα οποία δε χρειάζεστε ακόμα. Προς το παρόν θα εξηγήσουμε αυτά μόνο που χρειάζεστε σ' αυτό και τα επόμενα έξι κεφάλαια.

## BREAK (Διακοπή)

Αυτό σας χρησιμεύει στο να "διαφεύγετε" από καταστάσεις που δε σας αρέσουν. π.χ.

- μία γραμμή που αποφασίσατε να εγκαταλείψετε

- κάποιο λάθος που δεν καταλαβαίνετε

- ένα εκτελούμενο πρόγραμμα που έπαψε να σας ενδιαφέρει

- οποιοδήποτε άλλο πρόβλημα

Επειδή το BREAK είναι τόσο ισχυρό, είναι φτιαγμένο έτσι που να είναι δύσκολο να πατηθεί κατά λάθος.

κρατήστε πατημένο το CTRL και πατήστε το SPACE

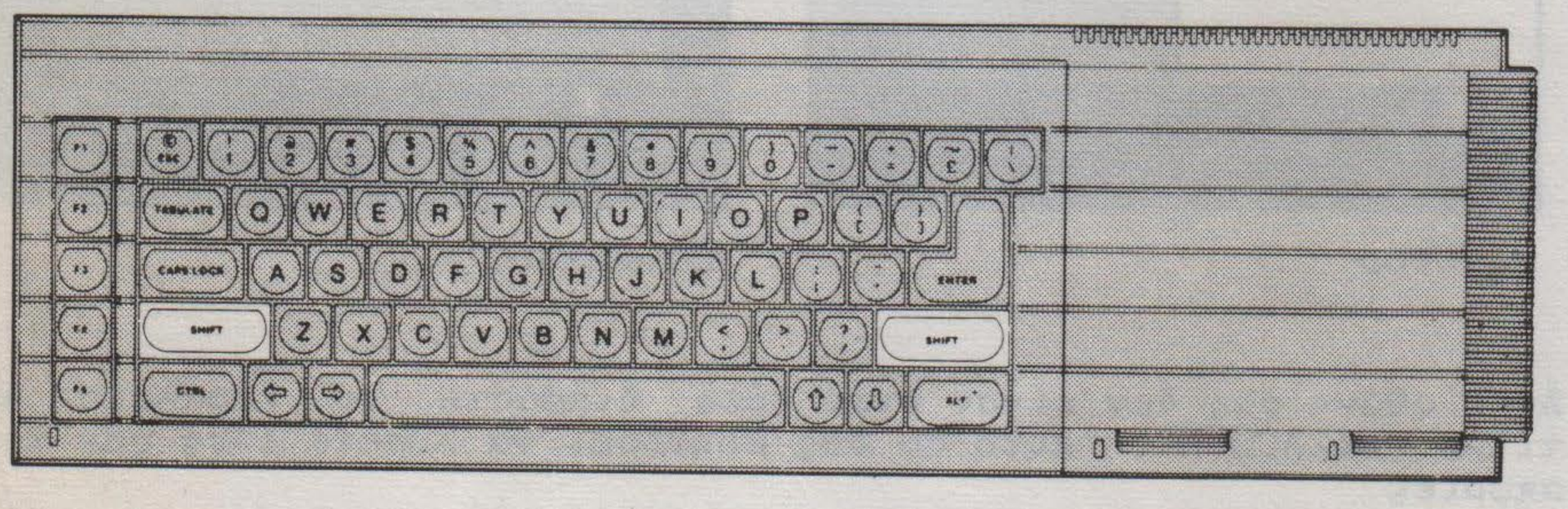
Αν δεν έχει προστεθεί ή αφαιρεθεί τίποτα σ' ένα πρόγραμμα το οποίο σταμάτησε με BREAK τότε μπορεί να ξαναρχίσει πληκτρολογώντας:

CONTINUE

Bad line

**RESET (Επαναφορά)**

Αυτό δεν είναι ένα πλήκτρο αλλά ένας μικρός διακόπτης στη δεξιά πλευρά του QL. Τοποθετήθηκε εκεί επίτηδες, απομακρυσμένο, γιατί τα αποτελέσματά του είναι πιο δραματικά από το BREAK. Αν δεν μπορείτε να κατορθώσετε αυτό που θέλετε με το BREAK τότε πιέστε το RESET. Αυτό είναι σχεδόν το ίδιο σαν να βγάζουμε τον υπολογιστή από την πρίζα και να τον ξανασυνδέσουμε. Το αποτέλεσμα είναι ένα καινούργιο ξεκίνημα λειτουργίας του από την αρχή.



**SHIFT**

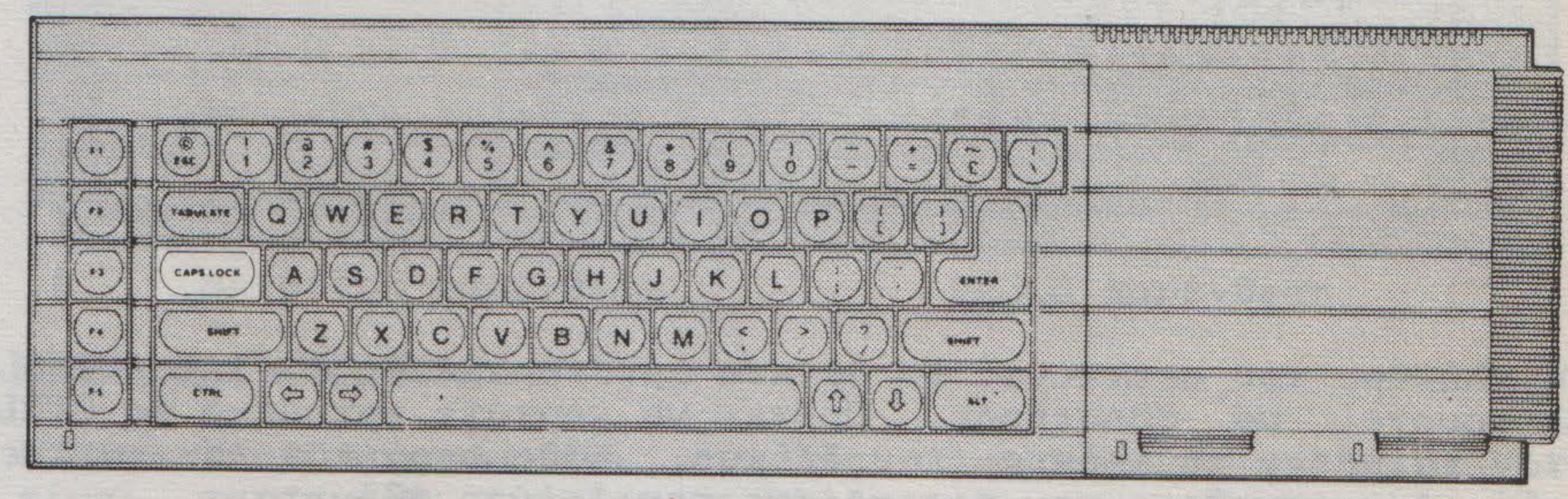
Υπάρχουν 2 πλήκτρα SHIFT επειδή χρησιμοποιούνται συχνά και πρέπει να είναι διαθέσιμα στο κάθε χέρι.

+ Κρατήστε πατημένο ένα πλήκτρο SHIFT και πατήστε μερικά πλήκτρα γραμμάτων θα πάρετε κεφαλαία γράμματα.

+ Κρατήστε πατημένο ένα πλήκτρο SHIFT και πατήστε άλλο πλήκτρο, όχι γράμμα. θα πάρετε το σύμβολο που βρίσκεται στο πάνω μέρος του πλήκτρου αυτού.

+ Χωρίς το SHIFT παίρνετε μικρά γράμματα ή σύμβολα που βρίσκονται στο κάτω μέρος του πλήκτρου.

**CAPITALS LOCK**



Το πλήκτρο αυτό δουλεύει σαν διακόπτης. Πιέστε το μόνο μία φορά, τότε τα πλήκτρα των γραμμάτων θα κλειδωθούν στα μικρά ή στα κεφαλαία γράμματα αντίστοιχα.

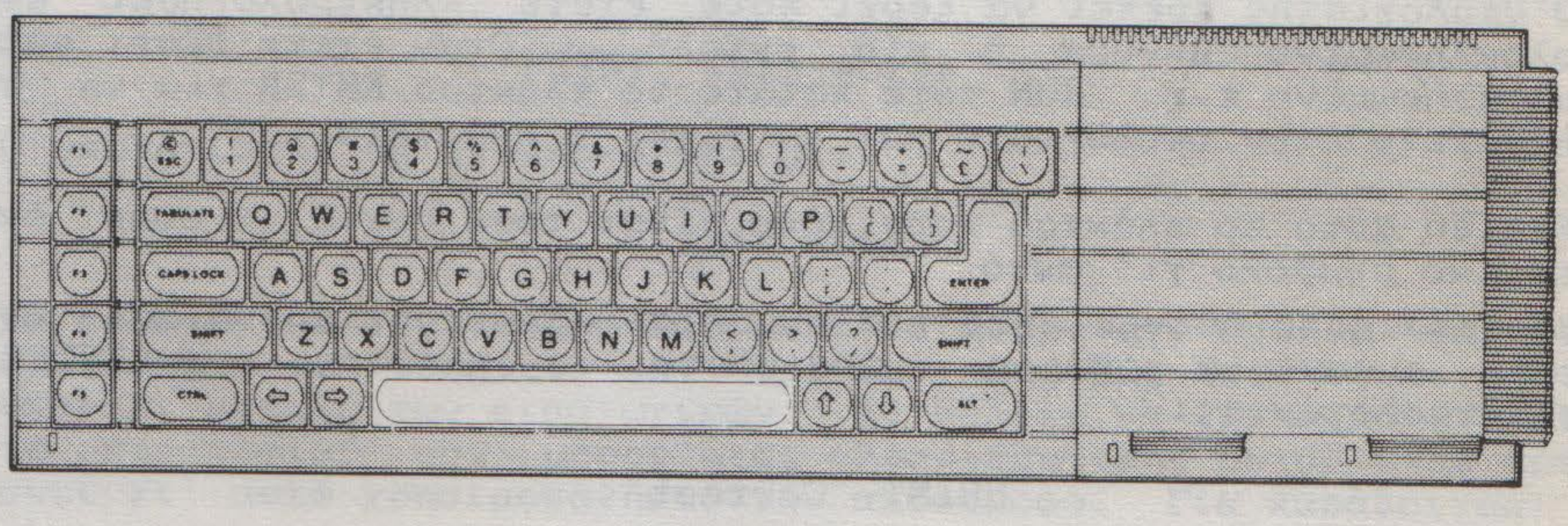
+ Πατήστε μερικά πλήκτρα γραμμάτων

+ Πατήστε το πλήκτρο CAPS LOCK μια φορά

+ Πατήστε μερικά πλήκτρα γραμμάτων.

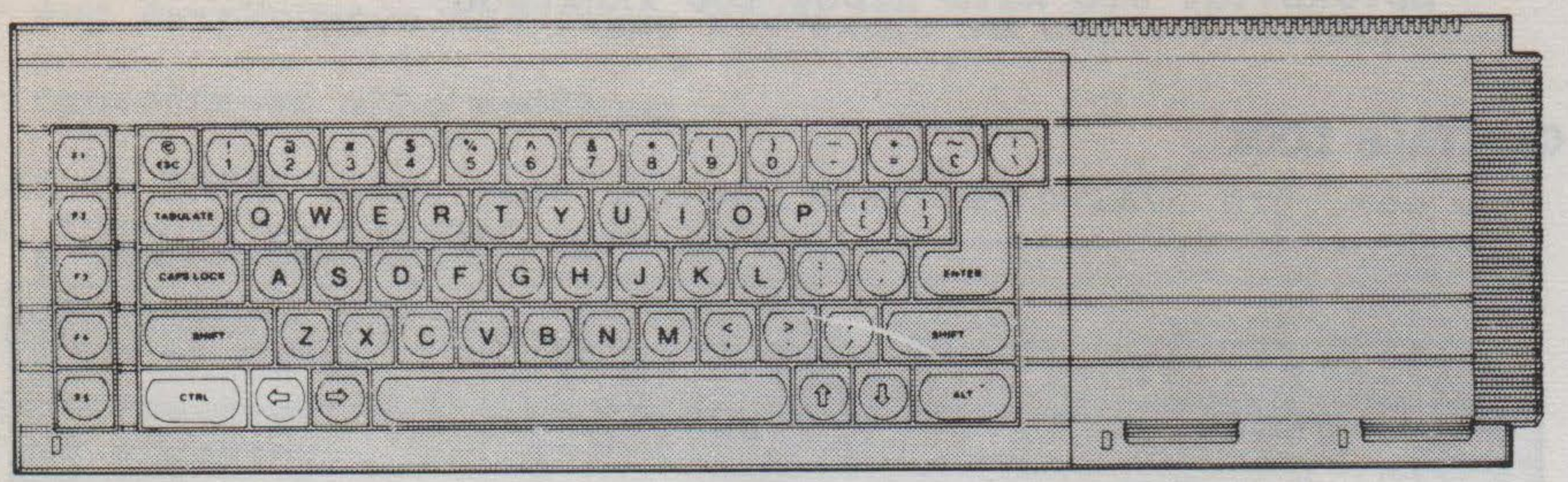
Θα δείτε ότι ο τύπος γραμμάτων (μικρά ή κεφαλαία) αλλάζει και παραμένει ο ίδιος μέχρι να πατήσετε το CAPS LOCK ξανά.

**SPACE BAR**



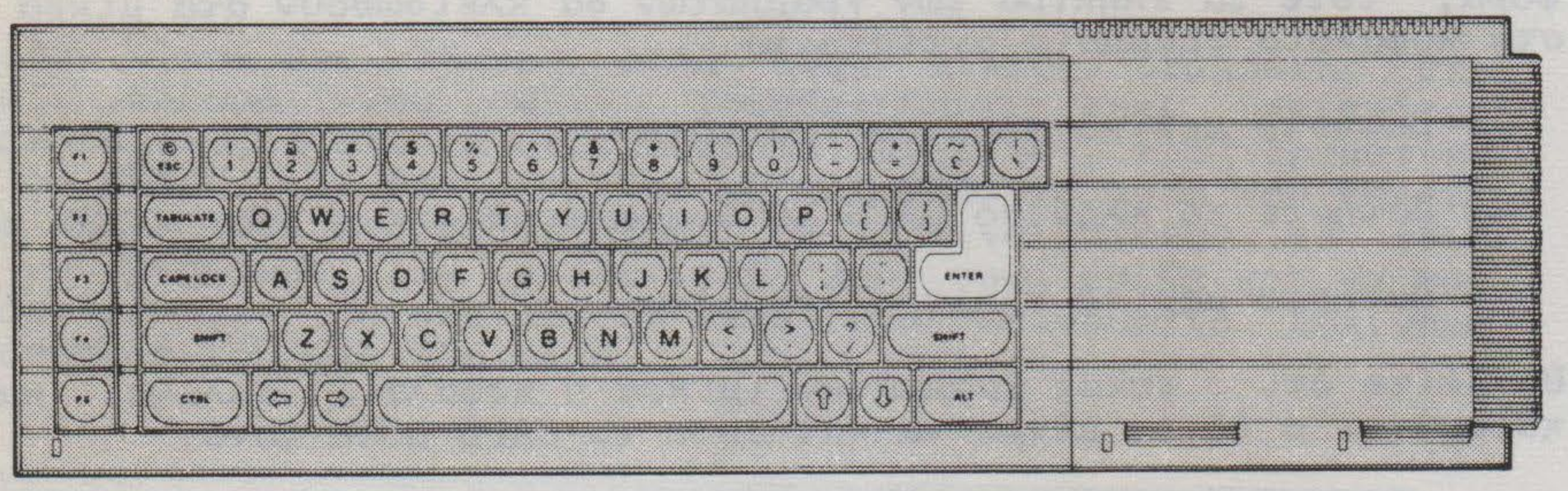
Το μακρύ πλήκτρο στο κάτω μέρος του πληκτρολογίου δίνει κενά διαστήματα. Είναι ένα πολύ σημαντικό πλήκτρο στη SuperBASIC όπως θα δείτε στο κεφάλαιο 2.

**ΣΒΥΣΙΜΟ**



Το πλήκτρο που μετακινεί το δείκτη αριστερά μαζί με το CTRL λειτουργεί σα σβηστήρας. Πρέπει να κρατάτε πατημένο το CTRL όταν πατήσετε το πλήκτρο του δείκτη. Κάθε φορά που τα πατάτε μαζί, ο χαρακτήρας που προηγείται του δείκτη σβήνεται.

**ENTER**



Ο υπολογιστής πρέπει να ξέρει πότε έχετε πληκτρολογήσει ένα ολοκληρωμένο μήνυμα ή μία εντολή. 'Οταν έχετε δώσει κάτι ολοκληρωμένο π.χ. RUN τότε πατάτε το πλήκτρο ENTER για να το εισάγετε στον υπολογιστή και να γίνει η ενέργεια.

Επειδή αυτό το πλήκτρο χρησιμοποιείται τόσο συχνά, έχουμε ένα ειδικό σύμβολο γι' αυτό

θα χρησιμοποιήσουμε αυτό για ευκολία, καλύτερη παρουσίαση και για οικονομία χώρου. Δοκιμάστε το πλήκτρο <- (ENTER) πληκτρολογώντας:

```
PRINT "Correct"<-
```

Αν δεν κάνατε λάθη ο υπολογιστής θα απαντήσει:

```
Correct
```

**ΑΛΛΑ ΠΛΗΚΤΡΑ ΤΟΥ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ΓΙΑ ΑΜΕΣΗ ΧΡΗΣΗ**

| | | | |
|---|---|---|---|
| * | πολλαπλασιασμός | + | πρόσθεση |
| / | διαίρεση | - | αφαίρεση |

| | | | |
|---|---|---|---|
| _ | υπογράμμιση | | ίσον (χρησιμοποιείται στην εντολή LET) |
| " | εισαγωγικά | ! | θαυμαστικό |
| , | κόμμα | ' | απόστροφος |
| ; | άνω τελεία | & | και |
| : | άνω κάτω τελεία | . | δεκαδικό σημείο ή τελεία |
| \ | ανάποδη | $ | δολλάριο |
| ( | αριστερή παρένθεση | ) | δεξιά παρένθεση |

**ΜΙΚΡΑ ΚΑΙ ΚΕΦΑΛΑΙΑ**

Η SuperBASIC αναγνωρίζει εντολές (Δεσμευμένες Λέξεις) είτε είναι σε μικρά είτε σε κεφαλαία γράμματα. Για παράδειγμα η εντολή της SuperBASIC που καθαρίζει την οθόνη είναι η CLS και μπορεί να πληκτρολογηθεί:

```
CLS<-
cls<-
clS<-
```

Αυτά είναι όλα σωστά και έχουν το ίδιο αποτέλεσμα. Μερικές λέξεις κλειδιά απεικονίζονται κατά ένα μέρος σε κεφαλαία για να δείξουν τον επιτρεπτό συγκοπτόμενο τύπο. 'Οταν μία δεσμευμένη λέξη δεν μπορεί να συγκοπεί, τότε απεικονίζεται ολόκληρη με κεφαλαία.

**Η ΧΡΗΣΗ ΕΙΣΑΓΩΓΙΚΩΝ**

Η συνηθισμένη χρήστη των εισαγωγικών είναι για να καθορίσουν μία λέξη ή πρόταση - μία αλφαριθμητική μεταβλητή. Δοκιμάστε:

```
PRINT "This works" <-
```

Ο υπολογιστής θα ανταποκριθεί με:

```
This works
```

Τα εισαγωγικά δεν τυπώνονται αλλά δείχνουν ότι κάποιο κείμενο πρόκειται να τυπωθεί και ορίζουν ακριβώς ποιο είναι - ότι υπάρχει μεταξύ των εισαγωγικών. Αν θέλετε να χρησιμοποιήσετε το εισαγωγικό σαν μέρος μιας αλφαριθμητικής μεταβλητής τότε αντί γι' αυτό χρησιμοποιήστε την απόστροφο. Για παράδειγμα:

```
PRINT'The quote symbol is" '
```

θα δουλέψει και θα εμφανίσει:

```
The quote symbol is"
```

**ΣΥΝΗΘΗ ΛΑΘΗ ΠΛΗΚΤΡΟΛΟΓΗΣΗΣ**

**Το μηδέν και το γράμμα 'O'**

Το 0 είναι μαζί με τους άλλους αριθμούς στο πάνω μέρος του

πληκτρολογίου και είναι λίγο πιο στενό. Το 'O' είναι μαζί με τα άλλα γράμματα. Προσέχετε να χρησιμοποιείτε πάντα το σωστό σύμβολο.

**Το ένα και το γράμμα 'I'**

Με τον ίδιο τρόπο αποφύγεται τη σύγχιση μεταξύ του 1 στους αριθμούς και του 'I' ανάμεσα στα γράμματα.

**ΚΡΑΤΑΤΕ ΤΟ SHIFT ΠΑΤΗΜΕΝΟ**

'Οταν χρησιμοποιείτε ένα πλήκτρο SHIFT κρατήστε το πατημένο όσο πληκτρολογείτε ώστε το SHIFT να κάνει επαφή πριν από το άλλο πλήκτρο.

Ο ίδιος κανόνας ισχύει και για τα πλήκτρα CTRL και ALT που χρησιμοποιούνται σε συνδυασμό με άλλα πλήκτρα αλλά αυτά δε σας χρειάζονται προς το παρόν.

**ΑΠΟΘΗΚΕΥΜΕΝΟ ΠΡΟΓΡΑΜΜΑ**

Πληκτρολογήστε τις 2 απλές εντολές:

```
CLS<-
PRINT'Hello'<-
```

Αυτές γενικά αποτελούν ένα πρόγραμμα. 'Ομως το αποθηκευμένο πρόγραμμα είναι το σημαντικό στον προγραμματισμό. Οι παραπάνω εντολές εκτελούνται αμέσως μόλις πατήσετε <- (ENTER).

Τώρα πληκτρολογήστε το πρόγραμμα με αριθμούς γραμμών:

```
100 CLS<-
110 PRINT 'HELLO'<-
```

Αυτή τη φορά τίποτα δε συμβαίνει εξωτερικά εκτός από το ότι το πρόγραμμα εμφανίζεται στο πάνω μέρος της οθόνης. Αυτό σημαίνει ότι είναι δεκτό σαν σωστό γραμματικά και συντακτικά και ότι είναι σύμφωνο με τους κανόνες της SuperBASIC, αλλά δεν έχει εκτελεστεί ακόμη, απλώς έχει αποθηκευτεί. Για να το κάνετε να δουλέψει πληκτρολογήστε:

```
RUN<-
```

Η διαφορά μεταξύ των απ' ευθείας εντολών για άμεση εκτέλεση και μιας αποθηκευμένης ακολουθίας εντολών θα συζητηθεί στο επόμενο κεφάλαιο. Προς το παρόν μπορείτε να πειραματιστείτε με τις παραπάνω ιδέες και με δύο ακόμη:

```
LIST<-
```

έχει σαν αποτέλεσμα ένα, αποθηκευμένο εσωτερικά, πρόγραμμα να εμφανιστεί στην οθόνη ή κάπου αλλού.

```
NEW<-
```

έχει σαν αποτέλεσμα ένα αποθηκευμένο εσωτερικά πρόγραμμα, να σβηστεί έτσι ώστε να μπορείτε να πληκτρολογήσετε ένα νέο.

## ΤΕΣΤ ΓΝΩΣΕΩΝ ΓΙΑ ΤΟ ΚΕΦΑΛΑΙΟ 1

Μπορείτε να έχετε ένα αποτέλεσμα 16 βαθμών το πολύ από το ακόλουθο τέστ. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Σε ποιες περιπτώσεις θα χρησιμοποιούσατε το BREAK;
2. Που είναι το κουμπί RESET;
3. Ποιό είναι το αποτέλεσμα του κουμπιού RESET;
4. Πέστε δύο διαφορές μεταξύ του SHIFT και του CAPS LOCK.
5. Πως μπορείτε να σβήσετε ένα λάθος χαρακτήρα που μόλις γράψατε;
6. Ποιος είναι ο σκοπός του πλήκτρου ENTER;
7. Ποιό σύμβολο χρησιμοποιούμε για το πλήκτρο ENTER;

   Ποιό είναι το αποτέλεσμα των εντολών στις ερωτήσεις 8 ως 11;

8. CLS<-
9. RUN<-
10. LIST<-
11. NEW<-
12. 'Εχουν οι δεσμευμένες λέξεις το σωστό αποτέλεσμα αν τις πληκτρολογήσετε;
13. Ποια είναι η σημασία του τμήματος των δεσμευμένων λέξεων τις οποίες το QL απεικονίζει με κεφαλαία γράμματα;

## ΑΠΑΝΤΗΣΕΙΣ ΣΤΟ ΤΕΣΤ ΤΟΥ ΚΕΦΑΛΑΙΟΥ 1

[1] Χρησιμοποιήστε το BREAK για να εγκαταλείψετε ένα τρέχον πρόγραμμα επειδή:

- κάτι είναι λάθος ή δεν το καταλαβαίνετε
- δεν έχει πια ενδιαφέρον
- οποιοδήποτε άλλο πρόγραμμα

[2] Το κουμπί RESET βρίσκεται στη δεξιά πλευρά του υπολογιστή.

[3] Το αποτέλεσμα του κουμπιού RESET είναι σαν να βγάζουμε τον υπολογιστή από τη πρίζα και να τον συνδέσουμε ξανά.

[4] Το πλήκτρο SHIFT:

a. έχει αποτέλεσμα μόνο όταν πατιέται συνεχώς ενώ το

CAPS LOCK έχει αποτέλεσμα που παραμένει και αφού το πιέσετε

b. το πλήκτρο SHIFT επηρεάζει όλα τα πλήκτρα γραμμάτων, ψηφίων και συμβόλων ενώ το πλήκτρο CAPS LOCK επηρεάζει μόνο τα γράμματα.

[5] Τα πλήκτρα CTRL<- σβήνουν το χαρακτήρα που προηγείται του δείκτη.

[6] Το πλήκτρο <- (ENTER) έχει σαν αποτέλεσμα την εισαγωγή ενός μηνύματος ή εντολής στον υπολογιστή για εκτέλεση.

[7] Χρησιμοποιούμε το <- αντί για το πλήκτρο ENTER.

[8] CLS<- έχει σαν αποτέλεσμα να καθαριστεί ένα μέρος της οθόνης.

[9] RUN<- έχει σαν αποτέλεσμα την εκτέλεση ενός αποθηκευμένου προγράμματος.

[10] LIST<- έχει σαν αποτέλεσμα ένα αποθηκευμένο πρόγραμμα να απεικονιστεί στην οθόνη.

[11] NEW<- καθαρίζει τη βασική μνήμη ώστε να είναι έτοιμη για ένα νέο πρόγραμμα.

[12] Οι δεσμευμένες λέξεις της SuperBASIC αναγνωρίζονται είτε σε μικρά είτε σε κεφαλαία γράμματα.

[13] Το τμήμα της δεσμευμένης λέξης που απεικονίζεται σε κεφαλαία, είναι ο επιτρεπτός συγκοπτόμενος τύπος.

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

- 14 ως 16 είναι πολύ καλά. Συνεχίστε να διαβάζετε.

- 12 ως 13 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 1.

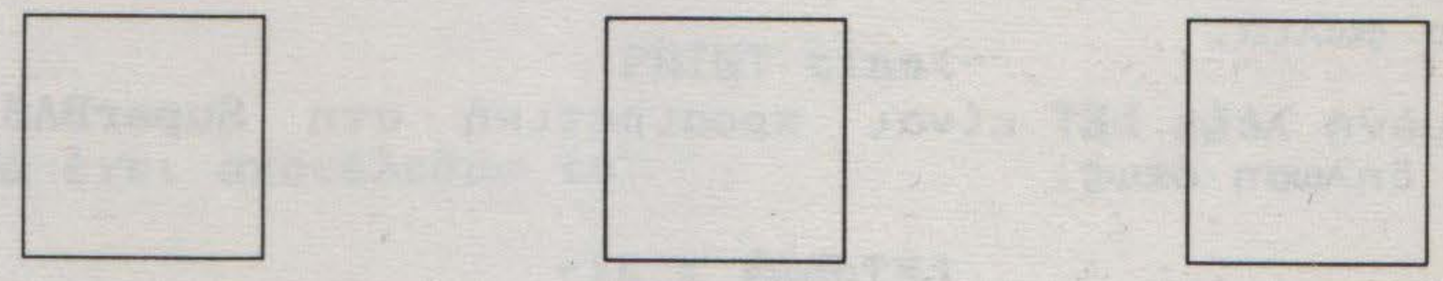
- 10 ή 11 είναι μέτρια, αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 1 και επαναλάβετε το τέστ.

- Κάτω από 10. Πρέπει να μελετήσετε πάλι προσεκτικά το κεφάλαιο 1 και να επαναλάβετε το τέστ.

# ΚΕΦΑΛΑΙΟ 2
# ΔΙΑΤΑΖΟΝΤΑΣ ΤΟΝ ΥΠΟΛΟΓΙΣΤΗ

**Αριθμοί, ονόματα και φωλιές**

Οι υπολογιστές χρειάζεται να αποθηκεύουν δεδομένα, όπως αριθμούς. Η αποθήκευση μπορεί να παρομοιαστεί με φωλιές.



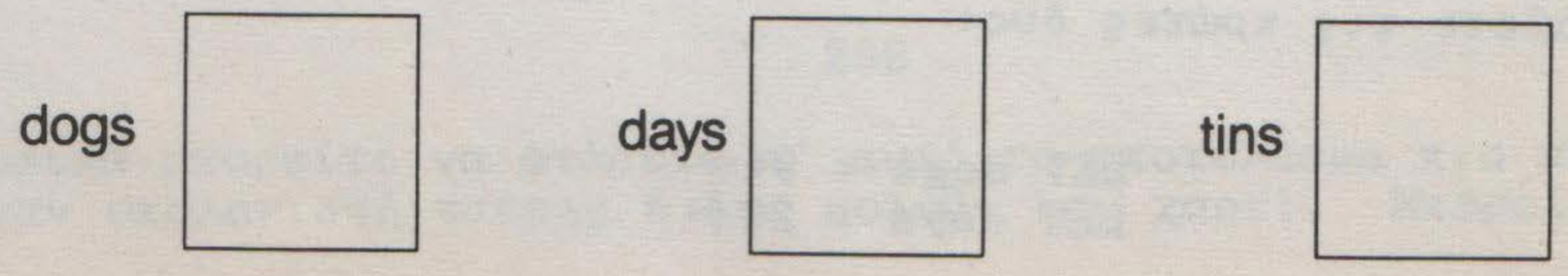
Παρόλο που δεν τις βλέπετε, χρειάζεται να δώσετε ονόματα στις καθορισμένες φωλιές. Υποθέτουμε ότι θέλετε να κάνετε τον ακόλουθο απλό υπολογισμό.

'Ενας παραγωγός σκύλων, έχει 9 σκύλους να ταίσει για 28 μέρες και χρειάζεται για τον καθένα μία κονσέρβα κρέας τη μέρα. Κάντε τον υπολογιστή να τυπώσει (να απεικονίσει στην οθόνη) τον απαιτούμενο αριθμό κονσέρβων.

'Ενας τρόπος επίλυσης του προβλήματος θα απαιτούσε τρεις φωλιές περιστεριών για

- αριθμός σκύλων (dogs)
- αριθμός ημερών (days)
- συνολικός αριθμός κονσερβών (tins)

Η SuperBASIC σας επιτρέπει να διαλέξετε ονόματα για τις φωλιές και μπορείτε να διαλέξετε όπως παρακάτω:



Μπορείτε να κάνετε τον υπολογιστή να φτιάξει μία φωλιά, να την ονομάσει και να αποθηκεύσει έναν αριθμό σ' αυτήν με μία μόνο εντολή ή δήλωση όπως

```
LET dogs = 9<-
```

Αυτό θα φτιάξει μία εσωτερική φωλιά, θα την ονομάσει dogs και θα τοποθετήσει σ' αυτήν τον αριθμό 9.


Η λέξη LET έχει ένα ειδικό νόημα στη SuperBASIC. Λέγεται δεσμευμένη λέξη. Η SuperBASIC έχει πολλές άλλες δεσμευμένες λέξεις όπως θα δείτε αργότερα. Πρέπει να είστε προσεκτικοί με το κενό διάστημα μετά το LET και τις άλλες δεσμευμένες λέξεις. Επειδή η SuperBASIC σας επιτρέπει να διαλέγετε ονόματα για φωλιές με μεγάλη ελευθερία, το LETdogs θα ήταν ένα επιτρεπτό όνομα για φωλιά.

Η δεσμευμένη λέξη LET είναι προαιρετική στη SuperBASIC και έτσι μία δήλωση όπως:

```
LETdogs = 3<-
```

είναι επιτρεπτή. Αυτή θα αναφερόταν σε μια φωλιά με όνομα LETdogs. 'Οπως και στη γλώσσα μας, τα ονόματα, οι αριθμοί και οι δεσμευμένες λέξεις πρέπει να χωρίζονται μεταξύ τους με διαστήματα αν δε χωρίζονται από ειδικούς χαρακτήρες.

Ακόμη κι αν δεν ήταν απαραίτητο, μια γραμμή προγράμματος χωρίς σωστά διαστήματα δείχνει κακό στυλ. Τα μηχανήματα με μικρό μέγεθος μνήμης μπορεί να αναγκάζουν τους προγραμματιστές να το κάνουν αυτό, αλλά με το QL δεν υπάρχει τέτοιο πρόβλημα.

Μπορείτε να ελέγξετε αν μια φωλιά υπάρχει εσωτερικά, πληκτρολογώντας:

```
PRINT dogs<-
```

Η οθόνη θα απεικονίσει ότι υπάρχει στη φωλιά:

```
9
```

Και πάλι προσέξτε να βάλετε ένα διάστημα μετά το PRINT.

Για να λύσουμε το πρόβλημα μπορούμε να γράψουμε ένα πρόγραμμα που είναι μια σειρά εντολών ή προτάσεων. Τώρα μπορείτε να καταλάβετε τις πρώτες δύο:

```
LET dogs = 9<-
LET days = 28<-
```

Αυτές έχουν αποτέλεσμα να φτιαχτούν δύο φωλιές, να ονομαστούν και να τους δοθούν αριθμοί ή ποσότητες.

Η επόμενη εντολή πρέπει να εκτελέσει έναν πολλαπλασιαμό, για τον οποίο το σύμβολο του υπολογιστή είναι * και να τοποθετήσει το αποτέλεσμα σε μία νέα φωλιά με όνομα tins.

LET tins = dogs * days<-

[1] Ο υπολογιστής παίρνει τις τιμές 9 και 28 από τις δύο φωλιές με ονόματα dogs και days.

[2] Ο αριθμός 9 πολλαπλασιάζεται με το 28.

[3] Μία νέα φωλιά δημιουργείται και ονομάζεται tins.

[4] Το αποτέλεσμα του πολλαπλασιασμού γίνετι η ποσότητα της φωλιάς με όνομα tins.

'Ολα αυτά μπορεί να φαίνονται περίτεχνα αλλά πρέπει να κατανοήσετε τις ιδέες, που είναι πολύ σημαντικές. Το αποτέλεσμα μπορείτε να το φανταστείτε πολύ απλά όπως φαίνεται: Η μόνη αποστολή που απομένει είναι να κάνετε τον υπολογιστή να τυπώσει το αποτέλεσμα, αυτό μπορεί να γίνει πληκτρολογώντας:

PRINT tins<-

που θα έχει αποτέλεσμα το:

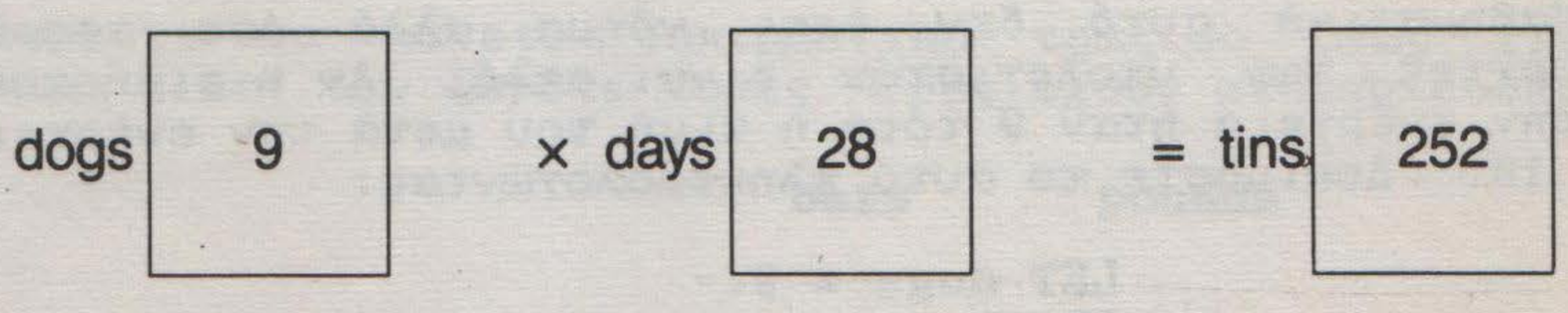
252

να εμφανιστεί στην οθόνη.

Συνοψίζοντας το πρόγραμμα:

```
LET dogs = 9<-
LET days = 28<-
LET tins = dogs * days<-
PRINT tins<-
```

'Εχει εσωτερικό αποτέλεσμα που πιο παραστατικά μπορούμε να το φανταστούμε σαν τρεις φωλιές με ονόματα που περιέχουν τους αριθμούς.




και το αποτέλεσμα στην οθόνη:

252

Φυσικά μπορείτε να φτάσετε σ' αυτό το αποτέλεσμα πιο εύκολα μ' έναν υπολογιστή τσέπης ή ένα μολύβι και χαρτί. Μπορείτε να το

κάνετε γρήγορα με το QL πληκτρολογώντας:

```
PRINT 9 * 28<-
```

το οποίο θα δώσει την απάντηση στην οθόνη. 'Ομως οι ιδέες που συζητήσαμε είναι οι βασικές αφαιτηρίες του προγραμματισμού στη SuperBASIC. Είναι τόσο βασικές ώστε συναντώνται σε πολλές γλώσσες υπολογιστών και τους έχουν δοθεί ειδικά ονόματα.

[1] Ονόματα όπως dogs, days και tins λέγονται: ταυτοποιητές ή αναγνωριστές. (identifiers).

[2] Μία απλή πρόταση όπως

```
LET dogs = 9<-
```

λέγεται εντολή.

[3] Η ταυτοποίηση του ονόματος και της αντίστοιχης φωλιάς λέγεται μεταβλητή. Η εκτέλεση της παραπάνω εντολής αποθηκεύει την τιμή 9 στη φωλιά που αναγνωρίζεται από τον αναγνωριστή dogs.

Μία εντολή όπως:

```
LET dogs = 9<-
```

είναι μία εντολή για μια δυναμική εσωτερική διεργασία αλλά το τυπωμένο κείμενο είναι στατικό και χρησιμοποιεί το σύμβολο = δανεισμένο από τα μαθηματικά. Είναι καλύτερα να σκέφτεστε ή να λέτε (αλλά όχι να πληκτρολογείτε).

```
LET dogs become 9
```

και να σκέφτεστε μία διεργασία με κατεύθυνση από δεξιά προς αριστερά (μην το πληκτρολογείτε αυτό):

```
dogs<--- 9
```

Η χρήση το = σε μία δήλωση LET δεν είναι η ίδια όπως η χρήση του = στα μαθηματικά. Για παράδειγμα αν εμφανιστεί άλλος ένας σκύλος θα θέλατε να γράψετε:

```
LET dogs = dogs + 1<-
```

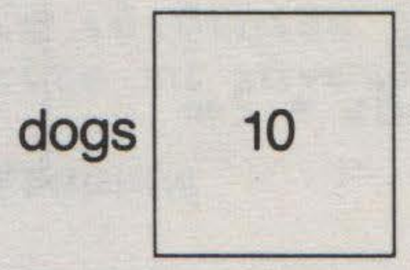
Στα Μαθηματικά αυτό δεν έχει νόημα αλλά όσο αφορά τις λειτουργίες των υπολογιστών είναι απλό. Αν η τιμή του dogs πριν την ενέργεια ήταν 9 τότε η τιμή του μετά την ενέργεια θα είναι 10. Δοκιμάστε το αυτό πληκτρολογώντας:

```
LET dogs = 9<-
PRINT dogs<-
LET dogs = dogs + 1<-
PRINT dogs<-
```

Η έξοδος στην οθόνη πρέπει να είναι:

```
9
10
```

δείχνοντας ότι η τελική τιμή στη φωλιά dogs είναι όπως φαίνεται:




'Ενας καλός τρόπος να καταλάβετε τι συμβαίνει στις φωλιές ή στις μεταβλητές, είναι αυτό που λέγεται "δοκιμαστικό τρέξιμο". Απλώς εξετάζετε κάθε εντολή με τη σειρά και καταγράφετε τις τιμές που προκύπτουν για να δείτε πως οι φωλιές δημιουργούνται, παίρνουν τιμές και πως διατηρούν τις τιμές τους καθώς το πρόγραμμα εκτελείται.

| | **dogs** | **days** | **tins** |
|---|---|---|---|
| **LET dogs = 9<-** | **9** | | |
| **LET days = 28<-** | **9** | **28** | |
| **LET tins = dogs * days<-** | **9** | **28** | **252** |
| **PRINT tins<-** | **9** | **28** | **252** |

Η έξοδος στην οθόνη πρέπει να είναι:

```
252
```

θα έχετε παρατηρήσει ως εδώ ότι ένα όνομα μεταβλητής χρησιμοποιείται για πρώτη φορά στο αριστερό μέλος μιας εντολής LET. Αφού η φωλιά δημιουργηθεί και πάρει μία τιμή, τότε το αντίστοιχο όνομα της μεταβλητής μπορεί να χρησιμοποιηθεί στο δεξιό μέλος μιας εντολής LET.

Τώρα ας υποθέσουμε ότι θέλετε να ενθαρύνετε ένα μικρό παιδί να κάνει οικονομίες. θα δίνατε δύο κομμάτια σοκολάτα για κάθε λίρα που αποταμιεύει το παιδί. Υποθέτουμε ότι θέλετε να το υπολογίσετε αυτό με τον ακόλουθο τρόπο:

```
LET bars = pounds * 2<-
PRINT bars<-
```

Δεν μπορείτε να κάνετε δοκιμαστικό τρέξιμο όπως έχει το πρόγραμμα γιατί δεν ξέρετε πόσες λίρες έχουν αποταμιευθεί.

| | **bars** | **pounds** |
|---|---|---|
| **LET bars = pounds * 2<-** | **?** | **?** |

'Εχουμε κάνει επίτηδες ένα λάθος χρησιμοποιώντας το pounds (=λίρες) στο δεξιό μέλος μιας εντολής LET χωρίς να το έχουμε πριν ορίσει και να του δώσουμε κάποια τιμή. Το QL σας, θα ψάξει εσωτερικά για τη μεταβλητή pounds. Δεν θα τη βρει και θα καταλήξει στο ότι υπάρχει ένα λάθος στο πρόγραμμα και θα δώσει ένα μήνυμα λάθους. Λέμε ότι η μεταβλητή pounds δεν έχει

οριστεί (δεν της έχει δοθεί μία αρχική τιμή). Το πρόγραμμα δουλεύει σωστά αν κάνετε αυτό πρώτα:

```
LET pounds = 7←
LET bars = pounds * 2←
```

| bars | pounds |
|---|---|
| | 7 |
| 14 | 7 |

Το πρόγραμμα δουλεύει σωστά και δίνει το αποτέλεσμα:

```
14
```

**'Ενα αποθηκευμένο πρόγραμμα**

Η πληκτρολόγηση εντολών χωρίς αριθμούς γραμμής μπορεί να έχει το ποθούμενο αποτέλεσμα αλλά υπάρχουν δύο λόγοι για τους οποίους η μέθοδος αυτή, όπως χρησιμοποιήθηκε ως τώρα, δεν είναι ικανοποιητική εκτός σαν μία πρώτη εισαγωγή.

1. Το πρόγραμμα εκτελείται τόσο γρήγορα όσο πληκτρολογείτε. Αυτό δεν είναι καθόλου εντυπωσιακό για ένα μηχάνημα που μπορεί να κάνει εκατομμύρια λειτουργίες το δευτερόλεπτο.

2. Οι εντολές δεν αποθηκεύονται μετά την εκτέλεση ώστε δεν μπορείτε να τρέξετε το πρόγραμμα ξανά ή να διορθώσετε ένα λάθος χωρίς να το ξαναπληκτρολογήσετε όλο από την αρχή.

Ο Charles Babbage, ένας πρωτοπόρος των υπολογιστών του 19ου αιώνα ήξερε ότι ένας πετυχημένος υπολογιστής έπρεπε να αποθηκεύει τις εντολές όπως και τα δεδομένα σε εσωτερικές φωλιές. Αυτές οι εντολές θα μπορούσαν τότε να εκτελεστούν με σειρά, γρήγορα χωρίς άλλη ανθρώπινη επέμβαση.

Οι εντολές του προγράμματος θα αποθηκευτούν αλλά δεν θα εκτελεστούν αν χρησιμοποιήσετε αριθμούς γραμμής. Δοκιμάστε αυτό:

```
10 LET price = 15<-
20 LET pens = 7<-
30 LET cost = price * pens<-
40 PRINT cost<-
```

Τίποτα δε συμβαίνει εξωτερικά ακόμη, αλλά ολόκληρο το πρόγραμμα είναι αποθηκευμένο εσωτερικά. Μπορείτε να το κάνετε να δουλέψει πληκτρολογώντας:

```
RUN<-
```

και η έξοδος:

```
105
```

θα εμφανιστεί.

Το πλεονέκτημα αυτής της διευθέτησης είναι ότι μπορείτε να

διορθώσετε ή να προσθέσετε κάτι στο πρόγραμμα με το ελάχιστο ποσό επί πλέον πληκτρολόγησης.

**Διορθώνοντας ένα πρόγραμμα**

Αργότερα θα δείτε όλες τις ιδιότητες διόρθωσης στη SuperBASIC αλλά ακόμη και τώρα μπορείτε να κάνετε τρία πράγματα εύκολα.

- αντικατάσταση μιας γραμμής
- εισαγωγή μιας νέας γραμμής
- διαγραφή μιας γραμμής

**Αντικατάσταση μιας γραμμής**

Υποθέτουμε ότι θέλετε να αλλάξετε το προηγούμενο πρόγραμμα επειδή η τιμή έχει αλλάξει σε 20 πέννες για κάθε στυλό (=pen). Απλώς ξαναπληκτρολογήστε τη γραμμή 10.

```
10 LET price = 20<-
```

Αυτή η γραμμή θα αντικαταστήσει την προηγούμενη γραμμή 10. Υποθέτοντας ότι οι άλλες γραμμές είναι ακόμη αποθηκευμένες, δοκιμάστε το πρόγραμμα πληκτρολογώντας:

```
RUN<-
```

και η νέα απάντηση, 140, θα εμφανιστεί.

**Εισαγωγή μιας νέας γραμμής**

Υποθέτουμε ότι θέλετε να εισάγετε μία γραμμή αμέσως πριν από την τελευταία, για να τυπώσει τις λέξεις "Total Cost". Αυτή η κατάσταση συμβαίνει συχνά γι' αυτό συνήθως διαλέγουμε αριθμούς γραμμής 10, 20, 30, ... για να υπάρχει χώρος για εισαγωγή νέων γραμμών, μεταξύ αυτών.

Για να τοποθετήσετε την επί πλέον γραμμή πληκτρολογήστε:

```
35 PRINT "Total Cost"<-
```

και αυτή θα τοποθετηθεί αμέσως πριν από τη γραμμή 40. Το σύστημα επιτρέπει αριθμούς γραμμής από 1 ως 32768 δίνοντας μας έτσι μεγάλη άνεση στην επιλογή τους. Είναι δύσκολο να είστε προκαταβολικά σίγουροι για το τι αλλαγές μπορεί να χρειαστούν.

Τώρα πληκτρολογήστε:

```
RUN<-
```

και η νέα έξοδος προς την οθόνη πρέπει να είναι:

```
Total cost
140
```

**Σβήσιμο γραμμής**

Μπορείτε να σβήσετε τη γραμμή 35 πληκτρολογώντας:

35<-

Είναι σαν να αντικαταστάθηκε η παλιά γραμμή από μία άδεια γραμμή.

**ΕΞΟΔΟΣ-ΕΚΤΥΠΩΣΗ**

Σημειώστε πόσο χρήσιμη είναι η εντολή PRINT. Μπορείτε να εκτυπώσετε κείμενο χρησιμοποιώντας εισαγωγικά ή αποστρόφους:

PRINT "Chocolate bars"<-

Μπορείτε να τυπώσετε τις τιμές των μεταβλητών (τα περιεχόμενα των φωλιών), πληκτρολογώντας εντολές όπως:

PRINT bars<-

χωρίς να χρησιμοποιείτε εισαγωγικά.

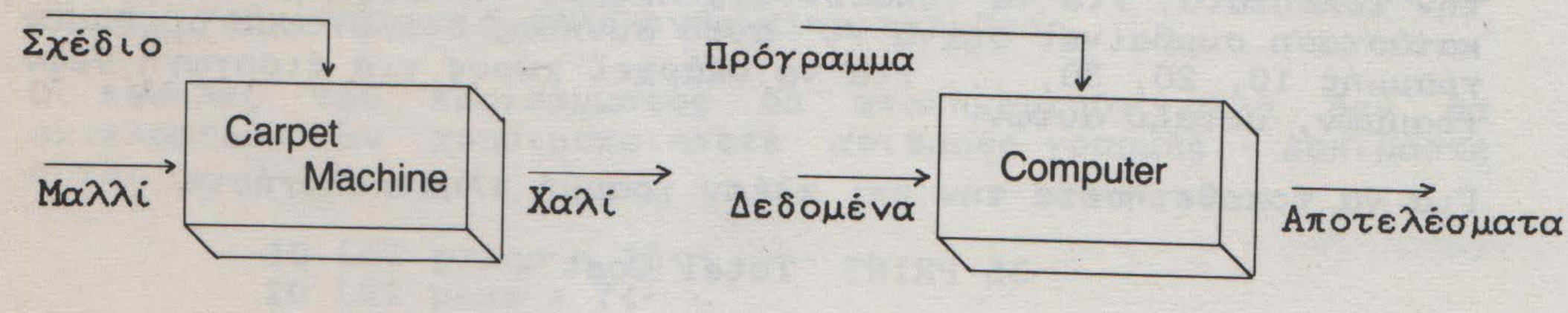
θα δείτε αργότερα πόσο έξυπνη είναι η εντολή PRINT στη SuperBASIC. θα σας επιτρέψει να εμφανίσετε κείμενο ή άλλο αποτέλεσμα στην οθόνη ακριβώς εκεί που το θέλετε. Αλλά προς το παρόν αυτές οι δύο δυνατότητες είναι αρκετά χρήσιμες:

- εκτύπωση κειμένου

- εκτύπωση τιμών μεταβλητών (τα περιεχόμενα των φωλιών)

**ΕΙΣΟΔΟΣ-INPUT, READ και DATA**

Μία μηχανή που φτιάχνει χαλιά χρειάζεται μαλλί για είσοδο. Μετά φτιάχνει χαλιά σύμφωνα με το τρέχον σχέδιο.




Αν το μαλλί αλλάζει μπορεί να πάρετε ένα διαφορετικό χαλί.

Το ίδιο είδος σχέσεων υπάρχει και σε έναν υπολογιστή.

'Ομως αν τα δεδομένα εισάγονται σε φωλιές με τον τρόπο του LET τότε υπάρχουν δύο μειονεκτήματα όταν προχωρήσετε πέρα από τα πολύ ασήμαντα προγράμματα:

- το γράψιμο των εντολών LET είναι κουραστικό

- η αλλαγή τέτοιων εισόδων είναι επίσης κουραστική.

Μπορείτε να το ρυθμίσετε έτσι ώστε τα δεδομένα να δίνονται στο πρόγραμμα καθώς αυτό τρέχει. Η εντολή INPUT θα έχει αποτέλεσμα να σταματήσει το πρόγραμμα και να σας περιμένει να γράψετε κάτι στο πληκτρολόγιο. Τώρα πληκτρολογήστε:

NEW

ώστε το προηγούμενο αποθηκευμένο πρόγραμμα (αν ήταν ακόμη εκεί) να σβηστεί και ο υπολογιστής να είναι έτοιμος για το νέο. Τώρα πληκτρολογήστε:

```
100 LET price = 15<-
110 PRINT "How many pens?"<-
120 INPUT pens<-
130 LET cost = price * pens<-
140 PRINT cost<-
RUN<-
```

Το πρόγραμμα σταματάει στη γραμμή 120 και πρέπει να πληκτρολογήσετε τον αριθμό των pens που θέλετε, ας πούμε:

```
4<-
```

Μην ξεχνάτε το πλήκτρο ENTER. Το αποτέλεσμα θα είναι:

```
60
```

Η εντολή INPUT χρειάζεται ένα όνομα μεταβλητής ώστε το σύστημα να ξέρει που να βάλει τα δεδομένα που δίνονται από εσάς όταν πληκτρολογείτε. Το αποτέλεσμα της γραμμής 120 μαζί με την πληκτρολόγηση σας είναι το ίδιο όπως μία εντολή LET. Είναι πιο βολικό για αρκετές περιπτώσεις όταν η επικοινωνία μεταξύ υπολογιστή και χρήστη είναι επιθυμητή. 'Ομως οι εντολές LET και INPUT είναι χρήσιμες μόνο για μικρά ποσά δεδομένων. Χρειαζόμαστε κάτι άλλο για να χειριστούμε μεγαλύτερα ποσά δεδομένων χωρίς παύσεις στην εκτέλεση του προγράμματος.

Η SuperBASIC όπως οι περισσότερες BASIC έχει μία άλλη μέθοδο εισαγωγής γνωστή σαν διάβασμα (READing) από εντολές δεδομένων (DATA). Μπορούμε να ξαναπληκτρολογήσουμε το παραπάνω πρόγραμμα σε μία νέα μορφή ώστε να δίνει τα ίδια αποτελέσματα χωρίς καμιά παύση. Δοκιμάστε αυτό:

```
NEW<-
100 READ price, pens<-
110 LET cost = price * pens<-
120 PRINT cost<-
130 DATA 15,4<-
RUN<-
```

Το αποτέλεσμα πρέπει να είναι:

```
60
```

όπως πριν.

Κάθε φορά που το πρόγραμμα τρέχει πρέπει να ξέρει η SuperBASIC από που να αρχίσει να διαβάζει δεδομένα. Αυτό μπορεί να γίνει είτε πληκτρολογώντας RESTORE ακολουθούμενο από τον αριθμό γραμμής της εντολής DATA ή πληκτρολογώντας CLEAR. Και οι δύο αυτές εντολές μπορούν να μπουν στην αρχή των προγραμμάτων.

'Οταν η γραμμή 100 εκτελεστεί το σύστημα ψάχνει το πρόγραμμα για μία εντολή DATA. Μετά χρησιμοποιεί τις τιμές στην εντολή

DATA για τις μεταβλητές της εντολής READ με την ίδια ακριβώς σειρά. Συνήθως βάζουμε τις εντολές DATA στο τέλος του προγράμματος. Αυτές χρησιμοποιούνται από το πρόγραμμα αλλά δεν εκτελούνται με την έννοια που οι άλλες εντολές εκτελούνται με σειρά. Οι εντολές DATA μπορούν να είναι οπουδήποτε σ' ένα πρόγραμμα αλλά είναι καλύτερα στο τέλος για να μην εμποδίζουν. Να τις θεωρείτε απαραίτητες για το πρόγραμμα που εκτελείται, αλλά όχι μέρος του. Οι κανόνες για τη READ και τη DATA είναι οι εξής:

1. 'Ολες εντολές DATA υπολογίζονται σαν μία μακριά σειρά στοιχείων. Μέχρι εδώ αυτά τα στοιχεία ήταν αριθμοί, αλλά θα μπορούσαν να ήταν λέξεις.

2. Κάθε φορά που μία εντολή READ εκτελείται οι απαραίτητες τιμές παίρνονται από τις εντολές DATA και δίνονται στις μεταβλητές που εμφανίζονται στην εντολή READ.

3. Το σύστημα σημειώνει ποιες τιμές έχουν διαβαστεί (READ) με τη βοήθεια ενός εσωτερικού δείκτη. Αν ένα πρόγραμμα επιχειρήσει να διαβάσει περισσότερα στοιχεία απ' όσα υπάρχουν σε όλες τις εντολές DATA τότε θα δοθεί σήμα ότι υπάρχει λάθος.

## ΑΝΑΓΝΩΡΙΣΤΕΣ (ONOMATA)

'Εχετε χρησιμοποιήσει ονόματα για τις φωλιές όπως dogs, bars. Μπορείτε να διαλέγετε λέξεις σαν αυτές σύμφωνα με ορισμένους κανόνες:

- 'Ενα όνομα δεν πρέπει να περιέχει διαστήματα (κενά).

- 'Ενα όνομα πρέπει να αρχίζει με γράμμα.

- 'Ενα όνομα πρέπει να αποτελείται από γράμματα, ψηφία, $, %, _ (υπογράμμιση).

  Τα σύμβολα $, % έχουν ειδικό σκοπό, που θα εξηγηθεί αργότερα, αλλά μπορείτε να χρησιμοποιείτε την υπογράμμιση για να κάνετε ονόματα όπως:

```
dog_food
month_wage_total
```

  πιο ευανάγνωστα.

  Η SuperBASIC δε διακρίνει ανάμεσα σε μικρά και κεφαλαία γράμματα, έτσι ονόματα όπως TINS και tins είναι τα ίδια.

  Οι περισσότεροι χαρακτήρες που μπορεί να έχει ένα όνομα είναι 255.

Τα ονόματα που κατασκευάζονται σύμφωνα μ' αυτούς τους κανόνες λέγονται αναγνωριστές. Οι αναγνωριστές χρησιμοποιούνται και για άλλο σκοπό στη SuperBASIC και πρέπει να τους καταλάβετε. Οι κανόνες επιτρέπουν μεγάλη ελευθερία στην επιλογή των ονομάτων, έτσι μπορείτε να κάνετε τα προγράμματά σας πιο ευκολονόητα. Ονόματα όπως total, count, pens βοηθούν πιο πολύ από ονόματα όπως Z, P, Q.

## ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 2

Μπορείτε να έχετε ένα αποτέλεσμα 21 βαθμών το πολύ από το ακόλουθο τέστ. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Πως πρέπει να φαντάζεστε την εσωτερική αποθήκευση ενός αριθμού;

2. Πέστε δύο τρόπους αποθήκευσης μιας τιμής σε μία εσωτερική "φωλιά" που θα δημιουργηθεί (2 βαθμοί).

3. Πως μπορείτε να βρείτε την τιμή μιας εσωτερικής "φωλιάς".

4. Ποια είναι η συνηθισμένη ορολογία για μία "φωλιά";

5. Πότε μια "φωλιά" παίρνει την πρώτη της τιμή;

6. Μία μεταβλητή ονομάζεται έτσι, επειδή η τιμή της μπορεί να μεταβάλλεται καθώς ένα πρόγραμμα εκτελείται. Ποιος είναι ο συνηθισμένος τρόπος να προκληθεί μία τέτοια αλλαγή;

7. Το σύμβολο = σε μία εντολή LET δε σημαίνει "ίσον" όπως στα μαθηματικά. Τι σημαίνει ακριβώς;

8. Τι συμβαίνει όταν εισάγετε μία εντολή χωρίς αριθμό γραμμής;

9. Τι συμβαίνει όταν εισάγετε μία εντολή με αριθμό γραμμής;

10. Ποιος είναι ο σκοπός των εισαγωγικών σε μία εντολή PRINT;

11. Τι συμβαίνει όταν δε χρησιμοποιείτε εισαγωγικά σε μία εντολή PRINT;

12. Τι κάνει μία εντολή INPUT που μία εντολή LET δεν κάνει;

13. Ποιο είδος εντολών προγράμματος δεν εκτελούνται ποτέ;

14. Ποιος είναι ο σκοπός των εντολών DATA;

15. Ποια άλλη λέξη χρησιμοποιούμε για το όνομα μιας "φωλιάς" (μεταβλητής);

16. Γράψτε τρεις επιτρεπτούς αναγνωριστές που χρησιμοποιούν γράμματα, γράμματα και ψηφία, γράμματα και υπογράμμιση (τρεις βαθμοί).

17. Γιατί το πλήκτρο space (διάστημα) είναι τόσο σημαντικό στη SuperBASIC;

18. Γιατί είναι σημαντικό να διαλέγονται οι αναγνωριστές ελεύθερα στον προγραμματισμό;

## ΑΠΑΝΤΗΣΕΙΣ ΤΟΥ ΑΥΤΟΕΛΕΓΧΟΥ ΣΤΟ ΚΕΦΑΛΑΙΟ 2

1. Η εσωτερική αποθήκευση αριθμού είναι σαν μία "φωλιά" που μπορεί να ονομαστεί και να πάρει μία τιμή.

2. Μία εντολή LET που χρησιμοποιεί ένα ορισμένο όνομα για πρώτη φορά, έχει αποτέλεσμα να δημιουργηθεί μία "φωλιά" και να ονομαστεί, για παράδειγμα:

```
LET count = 1
```

Μία εντολή READ που χρησιμοποιεί ένα όνομα για πρώτη φορά θα έχει το ίδιο αποτέλεσμα, για παράδειγμα:

```
READ count
```

3. Μπορείτε να βρείτε την τιμή μιας "φωλιάς" με μία εντολή PRINT.

4. Η τεχνική ορολογία για μία "φωλιά" είναι "μεταβλητή" γιατί οι τιμές της μπορούν να μεταβάλλονται καθώς το πρόγραμμα εκτελείται.

5. Μία μεταβλητή παίρνει την πρώτη της τιμή όταν χρησιμοποιηθεί πρώτη φορά σε εντολή LET, INPUT ή READ.

6. Η αλλαγή της τιμής μιας μεταβλητής συνήθως προκαλείται από την εκτέλεση μιας εντολής LET.

7. Το σύμβολο = σε μία εντολή LET αντιπροσωπεύει μία λειτουργία:

   Υπολόγισε την τιμή της παράστασης του δεξιού μέλους και τοποθέτηση την στη "φωλιά" με όνομα αυτό που είναι στο αριστερό μέλος" ή αλλιώς: "Εξίσωσε το αριστερό μέλος με το δεξιό μέλος".

8. Μία εντολή χωρίς αριθμό γραμμής εκτελείται αμέσως.

9. Μία εντολή με αριθμό γραμμής δεν εκτελείται αμέσως, αποθηκεύεται.

10. Τα εισαγωγικά στην εντολή PRINT περικλείουν το κείμενο που πρόκειται να εκτυπωθεί.

11. 'Οταν δε χρησιμοποιούνται εισαγωγικά τότε εκτυπώνετε την τιμή μιας μεταβλητής.

12. Μία εντολή INPUT κάνει το πρόγραμμα να περιμένει ώστε να πληκτρολογήσετε τα δεδομένα.

13. Οι εντολές DATA δεν εκτελούνται ποτέ.

14. Χρησιμοποιούνται για να δίνουν τιμές στις μεταβλητές των εντολών READ.

15. Η ορολογία για το όνομα μιας "φωλιάς" είναι "αναγνωριστής";

16. Παράδειγμα απαντήσεων:

    i. day

    ii. day_23

iii. day_of_week

17. Το πλήκτρο space είναι τόσο σημαντικό για να βάζετε διαστήματα μετά ή πριν από τις δεσμευμένες λέξεις ώστε να μη λαμβάνονται σαν αναγνωριστές (ονόματα) διαλεγμένα από το χρήστη.

18. Η ελεύθερη επιλογή αναγνωριστών είναι σημαντική βοηθά στο να κάνει τα προγράμματα πιο ευκολονόητα. Τέτοια προγράμματα δεν είναι τόσο επιρρεπη σε λάθη και προσαρμόζεται πιο εύκολα.

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

- 18 ως 21 είναι πολύ καλά. Συνεχίστε να διαβάζετε.
- 16 ως 17 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 2.
- 14 ως 15 είναι μέτρια αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 2 και ξανακάνετε το τέστ.
- Κάτω από 14. Πρέπει να μελετήσετε πάλι προσεκτικά το κεφάλαιο 2 και να επαναλάβετε το τέστ.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 2**

1. Κάντε ένα δοκιμαστικό τρέξιμο για να δείτε τις τιμές όλων των μεταβλητών καθώς κάθε γραμμή του ακόλουθου προγράμματος εκτελείται.

2. Γρέψτε και δοκιμάστε ένα πρόγραμμα, όμοιο μ' αυτό του προβλήματος 1, που να υπολογίζει την έκταση ενός χαλιού που έχει 3 μέτρα πλάτος και 4 μέτρα μήκος. Χρησιμοποιήστε τα ονόματα μεταβλητών: width, length, area.

3. Ξαναγράψτε το πρόγραμμα του προβλήματος 1 έτσι ώστε να χρησιμοποιεί 2 εντολές INPUT αντί για εντολές LET.

4. Ξαναγράψτε τη λύση στο πρόβλημα 2 χρησιμοποιώντας μία διαφορετική μέθοδο για εισαγωγή δεδομένων. Χρησιμοποιήστε READ και DATA αν είχατε αρχικά χρησιμοποιήσει LET και το αντίθετο.

5. Ξαναγράψτε το πρόγραμμα του προβλήματος 1 έτσι ώστε τα εισαγόμενα δεδομένα (4 και 3) να εμφανίζονται σε μία εντολή DATA αντί με εντολή LET.

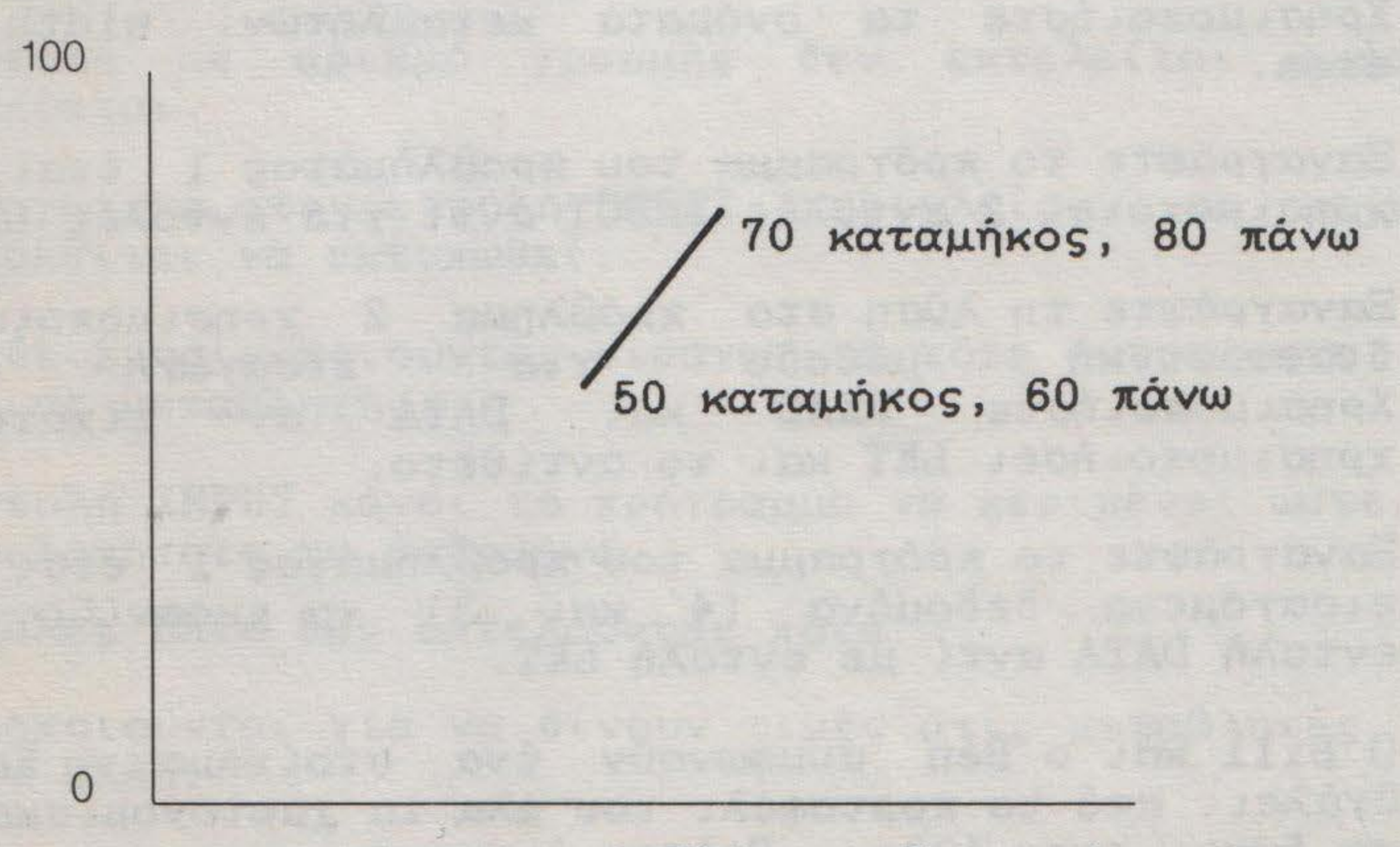
6. Ο Bill και ο Ben συμφωνούν ένα στοίχημα. Καθένας θα βγάλει από το πορτοφόλι του όλα τα χαρτονομίσματα και θα τα δώσει στον άλλο. Γράψτε ένα πρόγραμμα που να το κάνει αυτό, αποκλειστικά με LET και PRINT. Χρησιμοποιήστε ένα τρίτο πρόσωπο, την Sue που θα κρατάει τα χρήματα του Bill όσο αυτός θα παίρνει τα χρήματα του Ben.

7. Ξαναγράψτε το πρόγραμμα του προβλήματος 6 έτσι ώστε οι δύο αριθμοί που θα εναλλαχθούν να βρίσκονται σε μία δήλωση DATA.

# ΚΕΦΑΛΑΙΟ 3
# ΖΩΓΡΑΦΙΖΟΝΤΑΣ ΣΤΗΝ ΟΘΟΝΗ

Για να είναι το σύστημα συμβιβαστό με διάφορους τύπους μονάδων απεικόνισης (τηλεοράσεις ή μόνιτορς) δύο μορφές οθόνης είναι διαθέσιμες. 'Ομως θα ήταν πολύ κακό αν ένα πρόγραμμα γραμμένο για να σχεδιάζει κύκλους ή τετράγωνα στη μία μορφή οθόνης, σχεδίαζε ελλείψεις και παραλληλόγραμμα στην άλλη μορφή (όπως κάνουν μερικά συστήματα). Γι' αυτό χρησιμοποιούμε ένα σύστημα γραφικών με κλίμακα που αποφεύγει τέτοια προβλήματα. Απλώς διαλέγετε μία κατακόρυφη κλίμακα και δουλεύετε σ' αυτήν. Ο άλλος τύπος γραφικών (με ορισμό κουκίδας) είναι επίσης διαθέσιμος και περιγράφεται αναλυτικά σε επόμενο κεφάλαιο.

Υποθέτουμε, για παράδειγμα, ότι διαλέγουμε μία κατακόρυφη κλίμακα 100 και θέλουμε να σχεδιάσουμε μία γραμμή από τη θέση (50,60) στη θέση (70,80).




Κλίμακα Γραφικών

**Μία χρωματιστή γραμμή**

Πρέπει να ορίσουμε τέσσερρα πράγματα:

- Κλίμακα (κατακόρυφη) SPACE

- Χαρτί (χρώμα φόντου) PAPER

- Μελάνι (χρώμα σχεδίασης) INK

- Γραμμή (αρχικό και τελικό σημείο) LINE

Το ακόλουθο πρόγραμμα θα σχεδιάσει μία γραμμή όπως στο παραπάνω σχήμα σε κόκκινο χρώμα (κωδικούς χρώματος 2) πάνω σε λευκό (κώδικες χρώματος 7) φόντο.

```
NEW <-
100 PAPER 7 : CLS <-
110 INK 2 <-
120 LINE 50,60 TO 70,80 <-
RUN <-
```

Στη γραμμή 100 το χρώμα φόντου έχει επιλεγεί πρώτα αλλά ενεργοποιείται μόνο με μία άλλη εντολή, όπως CLS, που σημαίνει καθάρισε την οθόνη στο τρέχον χρώμα χαρτιού.

**Μορφές οθόνης και χρώματα**

Ως εδώ δεν είχε σημασία ποια μορφή οθόνης χρησιμοποιείτε, αλλά η κλίμακα των χρωμάτων επηρεάζεται από την επιλογή μορφής οθόνης.

- η MODE 8 επιτρέπει οκτώ βασικά χρώματα

- η MODE 4 επιτρέπει τέσσερα βασικά χρώματα

Τα χρώματα έχουν κώδικες όπως περιγράφεται παρακάτω:

| Κώδικας | Αποτέλεσμα | |
|---|---|---|
| | 8 χρώματα | 4 χρώματα |
| 0 | μαύρο | μαύρο |
| 1 | μπλε | μαύρο |
| 2 | κόκκινο | κόκκινο |
| 3 | μωβ | κόκκινο |
| 4 | πράσινο | πράσινο |
| 5 | γαλάζιο | πράσινο |
| 6 | κίτρινο | άσπρο |
| 7 | άσπρο | άσπρο |

Για παράδειγμα το INK 3 θα δώσει μωβ στο MODE 8 και κόκκινο στο MODE 4.

θα εξηγήσουμε σε επόμενο κεφάλαιο πως τα βασικά χρώματα μπορούν να αναμιχθούν με διάφορους τρόπους φτιάχνοντας μια καταπληκτική κλίμακα αποχρώσεων και σκιών αναμίξεων χρωμάτων.

**Τυχαίες επιδράσεις**

Μπορείτε να πάρετε μερικά ενδιαφέροντα αποτελέσματα με τυχαίους αριθμούς που μπορούν να παραχθούν με τη συνάρτηση RND. Για παράδειγμα:

```
PRINT RND(1 TO 6) <-
```

θα εκτυπώσει έναν ακέραιο από 1 ως 6, σαν να ρίξαμε ένα ζάρι. Το ακόλουθο πρόγραμμα θα το δείξει:

```
 NEW <-
100 LET die = RND(1 TO 6) <-
110 PRINT die <-
 RUN <-
```

Αν τρέξετε το πρόγραμμα αρκετές φορές θα πάρετε διαφορετικούς αριθμούς.

Μπορείτε να πάρετε ακέραιους τυχαίους αριθμούς σε όποια κλίμακα θέλετε. Για παράδειγμα:

```
RND(0 TO 100)
```

θα παράγει έναν αριθμό που μπορεί να χρησιμοποιηθεί σε κλίμακα γραφικών. Μπορείτε να ξαναγράψετε το πρόγραμμα που σχεδιάζει μία γραμμή, έτσι ώστε να παράγει ένα τυχαίο χρώμα. 'Οταν η κλίμακα των τυχαίων αριθμών αρχίζει από το 0 τότε μπορείτε να παραλείψετε τον πρώτο αριθμό και να γράψετε:

```
RND(100)

NEW <-
100 PAPER 7 : CLS <-
110 INK RND(5) <-
120 LINE 50,60 TO RND(100), RND(100) <-
RUN <-
```

Αυτό παράγει μία γραμμή που αρχίζει κάπου στο κέντρο της οθόνης και τελειώνει σε κάποιο τυχαίο σημείο. Η κλίμακα των πιθανών χρωμάτων εξαρτάται από τη μορφή οθόνης που διαλέξατε. θα ανακαλύψετε ότι μία κλίμακα αριθμών "από κάποιον έως κάποιον" συναντάται συχνά στη SuperBASIC.

## Περιθώρια

Το μέρος της οθόνης όπου σχεδιάσατε γραμμές ή δημιουργήσατε κάποιο άλλο αποτέλεσμα ονομάζεται "παράθυρο". Αργότερα θα δείτε πως μπορείτε να αλλάξετε το μέγεθος ενός παραθύρου και τη θέση του ή να δημιουργήσετε άλλα παράθυρα. Προς το παρόν θα αρκεστούμε στο να σχεδιάσουμε ένα περιθώριο γύρω από το τρέχον παράθυρο. Η μικρότερη περιοχή φωτεινότερου χρώματος που μπορείτε να σχεδιάσετε στην οθόνη λέγεται pixel. Στο MODE 8, που λέγεται "μορφή χαμηλής διακριτικότητας", υπάρχουν 256 δυνατές θέσεις pixel κατά μήκος της οθόνης και 256 θέσεις pixel κατά τη κατακόρυφο. Στη MODE 4, που λέγεται μορφή υψηλής διακριτικότητας υπάρχουν 512 pixels κατά μήκος της οθόνης και 256 κατά την κατακόρυφο. 'Ετσι το μέγεθος μιας κουκίδας εξαρτάται από τη μορφή οθόνης.

Μπορείτε να φτιάξετε ένα περιθώριο γύρω από την εσωτερική πλευρά του παραθύρου πληκτρολογώντας για παράδειγμα:

```
BORDER 4,2 <-
```

Αυτό θα δημιουργήσει ένα περιθώριο πλάτους 4 pixels και χρώματος κόκκινου (κώδικας 2). Το πραγματικό μέγεθος του παραθύρου μειώνεται με το περιθώριο. Αυτό σημαίνει ότι κάθε επόμενη εκτύπωση ή γραφική παράσταση θα ταιριάσουν αυτόματα μέσα στο νέο μέγεθος του παραθύρου. Η μόνη εξαίρεση σ' αυτό είναι ένα επί πλέον περιθώριο που θα γραφτεί πάνω στη θέση του υπάρχοντος.

**'Ενας απλός βρόχος**

Οι υπολογιστές μπορούν να κάνουν πολλά πράγματα πολύ γρήγορα αλλά δε θα ήταν δυνατό να εκμεταλλευτούμε αυτή τη μεγάλη ισχύ αν κάθε εργασία έπρεπε να γράφεται σαν μία εντολή. 'Ενας επιστάτης σε οικοδομή έχει ένα παρόμοιο πρόβλημα. Αν θέλει να τοποθετήσει ένας εργάτης εκατό πλακάκια τότε του λέει αυτό το πράγμα. Δεν του δίνει εκατό ξεχωριστές εντολές. 'Ενας παραδοσιακός τρόπος για να κάνουμε βρόχους ή επαναλήψεις στη BASIC είναι με τη χρήση της εντολής GO TO (ή GOTO, είναι το ίδιο) ως ακολούθως:

```
 NEW <-
100 PAPER 6 <-
110 BORDER 1,2 <-
120 INK RND(5) <-
130 LINE 50,60 TO RND(100), RND(100) <-
140 GOTO 40 <-
 RUN <-
```

'Ισως προτιμάτε να μην πληκτρολογήσετε αυτό το πρόγραμμα επειδή η SuperBASIC έχει έναν καλύτερο τρόπο για να κάνετε επαναλήψεις. Σημειώστε ορισμένα πράγματα για κάθε γραμμή:

```
100
110 Σταθερό μέρος. Δεν επαναλαμβάνεται
120
130
140 Εναλασσόμενο μέρος. Επαναλαμβάνεται
150 Ελέγχει το πρόγραμμα
```

Μπορείτε να ξαναγράψετε τα παραπάνω προγράμματα παραλείποντας το GOTO και αντί γι' αυτό, βάζοντας το REPEAT και END REPEAT γύρω από το μέρος που επαναλαμβάνεται.

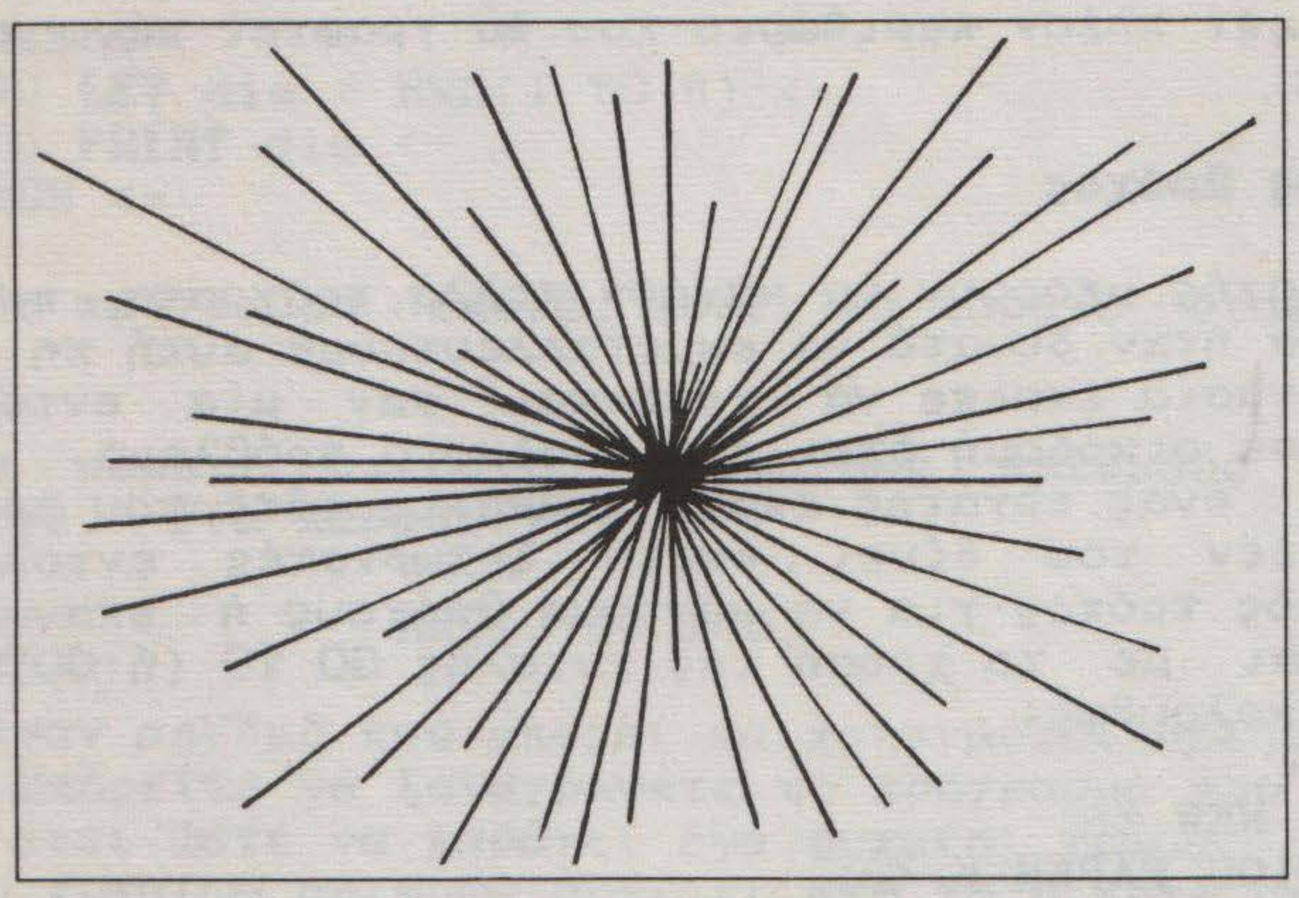
```
 NEW <-
100 PAPER 6 <-
110 BORDER 1,2 <-
120 REPEAT star <-
130 INK RND5 <-
140 LINE 50,60 TO RND(100), RND(100) <-
150 END REPEAT star <-
 RUN <-
```

Δώσαμε το όνομα star στη δομή της επανάληψης. Η δομή αποτελείται από δύο γραμμές:

```
REPeat star
END REPeat star
```

και ότι υπάρχει μεταξύ τους λέγεται περιεχόμενο της δομής.

Το πρόγραμμα πρέπει να φτιάξει χρωματιστές γραμμές τυχαία, για να κάνει ένα αστέρι όπως φαίνεται στο παρακάτω σχήμα:



Το πρόγραμμα STAR

Μπορείτε να το σταματήσετε πατώντας τα πλήκτρα BREAK:

κρατήστε πατημένο το CTRL και μετά πιέστε το SPACE

Η SuperBASIC περιέχει ένα συνεπή και έξυπνο τρόπο να σταματάτε επαναλαμβανόμενες διαδικασίες. Φανταστείτε να τρέχετε γύρω-γύρω μέσα σ' ένα πρόγραμμα εκτελώντας εντολές. Πως μπορείτε να ξεφύγετε; Η απάντηση είναι χρησιμοποιώντας μία δήλωση EXIT. Αλλά πρέπει να υπάρχει κάποιος λόγος για να ξεφύγετε. Μπορεί να θέλετε να επεκτείνετε την επιλογή των χρωμάτων της γραμμής για να συμπεριλάβετε και το άσπρο πληκτρολογώντας σαν τροποποίηση στο πρόγραμμα (μην πληκτρολογήσετε NEW ).

120 INK RND 0,6 <-

έτσι ώστε αν το RND παράγει το 6, τότε το μελάνι είναι άσπρο και δε θα το δείτε. Αυτός μπορεί να είναι ο λόγος για τον τερματισμό της επανάληψης. Μπορούμε να διευθετήσουμε το πρόγραμμα όπως φαίνεται:

```
 NEW <-
100 PAPER 6 <-
110 BORDER 1,2 <-
120 REPeat star <-
130 LET colour = RND(6) <-
140 IF colour = 6 EXIT star <-
150 INK colour <-
160 LINE 50,60 TO RND(100), RND(100) <-
170 END REPeat star <-
```

Το σημαντικό, που πρέπει να σημειώσετε, είναι ότι το πρόγραμμα συνεχίζει μέχρι το χρώμα να γίνει 6 (colour=6). Τότε ο έλεγχος ξεφεύγει από το βρόχο στο σημείο αμέσως μετά τη γραμμή 170. Αφού δεν υπάρχουν άλλες γραμμές μετά το 170, το πρόγραμμα σταματάει.

Μία άλλη σημαντική ιδέα έχει παρουσιαστεί. Είναι η ιδέα της απόφασης:

```
IF colour = 6 THEN EXIT star
```

Αυτό είναι μία άλλη πολύ χρήσιμη δομή γιατί είναι μία επιλογή αν θα γίνει κάτι ή όχι. Τη λέμε απλή δυαδική επιλογή. Η γενική της μορφή είναι:

```
IF συνθήκη THEN εντολή(ές)
```

θα δείτε αργότερα πως οι δύο ιδέες της επανάληψης (των βρόχων) και της λήψης αποφάσεων (των επιλογών) είναι οι βασικές δομές για τον έλεγχο των προγραμμάτων. Μπορείτε να το σταματήσετε πατώντας τα πλήκτρα BREAK κρατώντας πατημένο το CTRL και πατώντας ύστερα το SPACE.

## ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 3

Στο ακόλουθο τέστ, το άριστα είναι 13 βαθμοί. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Τί είναι μία pixel;
2. Πόσα pixels χωρούν κατά μήκος της οθόνης στη μορφή χαμηλής διακριτικότητας;
3. Πόσα pixels χωρούν στην κατακόρυφο στη μορφή χαμηλής διακριτικότητας;
4. Ποιοί είναι οι δύο αριθμοί που καθορίζουν τη "διεύθυνση" ή τη θέση ενός γραφικού σημείου στην οθόνη όταν χρησιμοποιείται κλίμακα;
5. Πόσα χρώματα είναι διαθέσιμα στη μορφή χαμηλής διακριτικότητας;
6. Ονομάστε τις δεσμευμένες λέξεις που κάνουν τα ακόλουθα;
   i. διαλέγει την κλίμακα για σχεδίαση
   ii. σχεδιάζει μία γραμμή
   iii. διαλέγει ένα χρώμα για σχεδίαση
   iv. διαλέγει ένα χρώμα φόντου
   v. σχεδιάζει ένα περιθώριο (5 βαθμοί)
7. Ποιές είναι οι δηλώσεις που ανοίγουν και κλείνουν ένα βρόχο REPeat;
8. Πότε ένας βρόχος REPeat που εκτελείται τελειώνει;

9. Γιατί οι βρόχοι στη SuperBASIC έχουν ονόματα;

**ΑΠΑΝΤΗΣΕΙΣ ΣΤΟΝ ΑΥΤΟΕΛΕΓΧΟ ΤΟΥ ΚΕΦΑΛΑΙΟΥ 3**

1. Το pixel είναι η μικρότερη φωτεινή περιοχή που μπορεί να απεικονιστεί στην οθόνη.

2. Υπάρχουν 256 θέσεις pixels κατά μήκος στη μορφή χαμηλής διακριτικότητας.

3. Υπάρχουν 256 θέσεις pixels κατά την κατακόρυφο στη μορφή χαμηλής διακριτικότητας.

4. Μία διεύθυνση καθορίζεται από:

   - την τιμή κατά μήκος από 0 ως 100

   - την τιμή στην κατακόρυφο, από 0 ως μία τιμή που υπολογίζεται από το σύστημα.

5. Υπάρχουν οκτώ χρώματα διαθέσιμα στη μορφή χαμηλής διακριτικότητας, συμπεριλαμβανομένων του μαύρου και του άσπρου.

6. i. η LINE σχεδιάζει μία γραμμή π.χ. LINE a,b TO x,y

   ii. η INK διαλέγει ένα χρώμα για σχεδίαση π.χ. INK 5

   iii. η PAPER διαλέγει ένα χρώμα φόντου: π.χ. PAPER 7

   iv. η BORDER σχεδιάζει ένα περιθώριο π.χ. BORDER 1,5

7. REPeat όνομα ... END REPeat όνομα

8. 'Ενας βρόχος REPEAT τελειώνει όταν εκτελεστεί μία δήλωση "EXIT όνομα"

9. Οι βρόχοι στη SuperBASIC έχουν ονόματα για να είναι δυνατή η έξοδος απ' αυτούς με έναν άμεσο τρόπο.

**ΕΛΕΓΞΕΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

- 11 ως 13 είναι πολύ καλά. Συνεχίστε να διαβάζετε.

- 8 ως 10 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου τρία.

- 6 και 7 είναι μέτρια αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 3 και ξανακάντε το τεστ.

- Κάτω από 6. Πρέπει να μελετήσετε πάλι προσεκτικά το κεφάλαιο 3 και να επαναλάβετε το τέστ.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 3**

1. Γράψτε ένα πρόγραμμα που να σχεδιάζει ευθείες γραμμές σε

όλη την οθόνη. Οι γραμμές πρέπει να είναι τυχαίου μήκους και διεύθυνσης. Κάθε μία πρέπει να ξεκινάει από εκεί που τελείωσε η προηγούμενη και κάθε μία πρέπει να έχει ένα χρώμα επιλεγμένο τυχαία.

2. Γράψτε ένα πρόγραμμα που να σχεδιάζει γραμμές τυχαία με τον περιορισμό ότι κάθε γραμμή έχει μία τυχαία αρχή στην αριστερή άκρη της οθόνης.

3. Γράψτε ένα πρόγραμμα που να σχεδιάζει γραμμές τυχαία με τον περιορισμό οι γραμμές να ξεκινούν από το ίδιο σημείο στην κάτω άκρη της οθόνης.

4. Γράψτε ένα πρόγραμμα που να φτιάχνει γραμμές τυχαίου μήκους, αρχικού σημείου και χρώματος. 'Ολες οι γραμμές πρέπει νάναι οριζόντιες.

5. 'Οπως το πρόβλημα 4 αλλά κάντε τις γραμμές κατακόρυφες.

6. Γράψτε ένα πρόγραμμα που να φτιάχνει ένα τετράγωνο "σπιράλ" με τέτοιο τρόπο που κάθε γραμμή να έχει τυχαίο χρώμα.

ΥΠΟΔΕΙΞΗ: Βρέστε πρώτα τις συντεταγμένες μερικών γωνιών και μετά βάλτε τις σε ομάδες των τεσσάρων. θα πρέπει να ανακαλύψετε ένα σχέδιο.

ΚΕΦΑΛΑΙΟ 4

# ΧΑΡΑΚΤΗΡΕΣ ΚΑΙ ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ

Μερικές φορές θέλουμε να εκτιμήσουμε την ικανότητα ανάγνωσης που χρειάζεται για ορισμένα βιβλία ή μέσα διδασκαλίας. Χρησιμοποιούμε διάφορα τέστ και μερικά από αυτά υπολογίζουν το μέσο μήκος λέξεων και προτάσεων. θα εισάγουμε ιδέες σχετικές με το χειρισμό λέξεων ή σειρών από χαρακτήρες, εξετάζοντας απλές προσεγγίσεις στην εύρεση του μέσου μήκους λέξης.

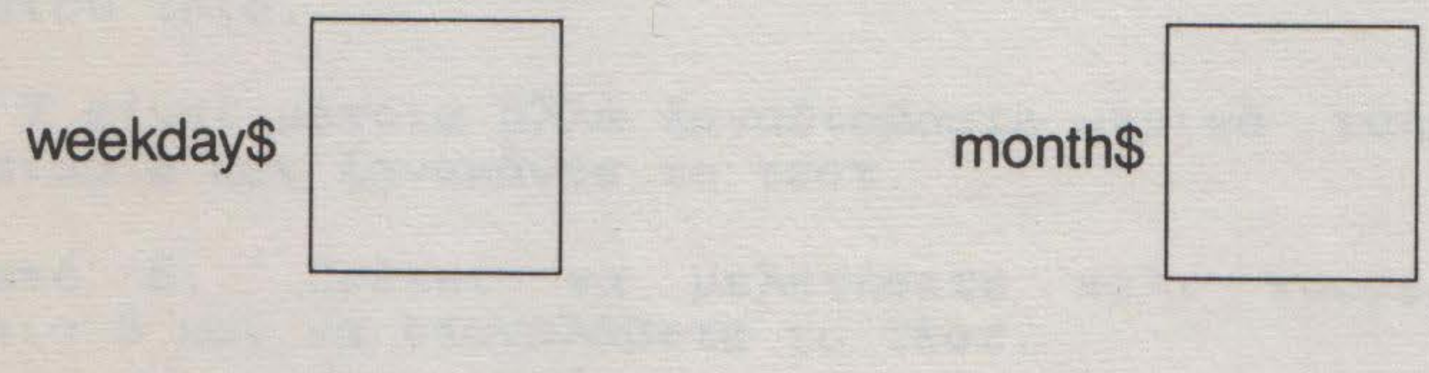
Μιλάμε για ακολουθίες από γράμματα, ψηφία ή άλλα σύμβολα που μπορεί να είναι ή να μην είναι λέξεις. Γι' αυτό ευφευρέθηκε ο όρος "σειρά χαρακτήρων". Συνήθως αυτό λέγεται αλφαριθμητική μεταβλητή. Ο χειρισμός των αλφαριθμητικών μεταβλητών είναι σε πολλά σημεία όμοιος με το χειρισμό των αριθμών αλλά φυσικά δεν μπορούμε να κάνουμε τις ίδιες λειτουργίες πάνω τους. Δεν πολαπλασιάζουμε και δεν αφαιρούμε αλφαριθμητιές μεταβλητές. Τις ενώνουμε, τις χωρίζουμε, τις ψάχνουμε και γενικά τις χειριζόμαστε όπως χρειάζεται.

## ΟΜΑΤΑ ΚΑΙ ΦΩΛΙΕΣ ΓΙΑ ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ

Μπορείτε να δημιουργήσετε "φωλιές" για αλφαριθμηκές μεταβλητές. Μπορείτε να βάλετε σειρές χαρακτήρων σε "φωλιές" και να χρησιμοποιήσετε τις πληροφορίες όπως ακριβώς με τους αριθμούς. Αν θέλετε να αποθηκεύσετε λέξεις (όχι όλες μαζί) όπως:

FIRST SECOND THIRD
and
JANUARY FEBRUARY MARCH

μπορεί να διαλέξετε να ονομάσετε δύο φωλιές περιστεριών:




Σημειώστε το σύμβολο του δολαρίου. Οι φωλιές για αλφαριθμητικές μεταβλητές είναι εσωτεκά διαφορετικές από τις φωλιές των αριθμών και η SuperBASIC χρειάζεται να ξέρει ποια είναι ποια. 'Ολα τα ονόματα φωλιών για αλφαριθμητικές μεταβλητές πρέπει να τελειώνουν με $. Κατά τα άλλα οι κανόνες επιλογής των ονομάτων είναι οι ίδιοι όπως στα ονόματα των φωλιών για αριθμητικές μεταβλητές.

Μπορείτε να προσφέρετε:

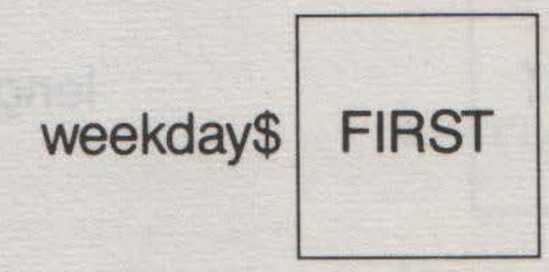
weekday$  σαν weekdaydollar

month$  σαν monthdollar

Η εντολή LET λειτουργεί με τον ίδιο τρόπο όπως και στους αριθμούς. Αν πληκτρολογήσετε:

LET weekday$ ='FIRST' <-

μια εσωτερική φωλιά με όνομα weekday$ θα δημιουργηθεί και θα περιέχει το FIRST έτσι:




Τα εισαγωγικά δεν αποθηκεύονται. Χρησιμοποιούνται στο LET για να δείξουν ακριβώς τι πρόκειται να αποθηκευτεί στη φωλιά. Μπορείτε να χρησιμοποιείτε αποστρόφους αντί για εισαγωγικά. Μπορείτε να ελέγξετε πληκτρολογώντας:

PRINT weekday$ <-

και η οθόνη πρέπει να απεικονίσει ότι υπάρχει μέσα στη φωλιά:

FIRST

**ΜΗΚΗ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ**

Είναι εύκολο στη SuperBASIC να βρείτε το μήκος, ή τον αριθμό των χαρακτήρων, οποιασδήποτε αλφαριθμητικής μεταβλητής. Απλώς γράφετε, για παράδειγμα:

PRINT LEN(weekday$)<-

Αν η φωλιά weekday$ περιέχει το FIRST τότε ο αριθμός 5 θα εμφανιστεί. Μπορείτε να δείτε το αποτέλεσμα σ' ένα απλό πρόγραμμα.

```
 NEW<-
100 LET weekday$ = "FIRST"<-
110 PRINT LEN(weekday$)<-
RUN<-
```

Η οθόνη πρέπει να εμφανίσει: 5.

Το LEN είναι μία λέξη κλειδί της SuperBASIC.

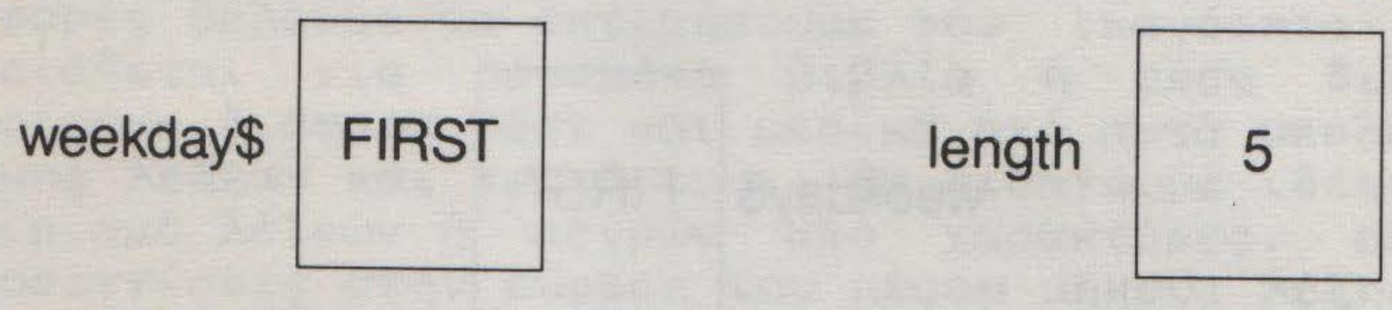
Μια εναλλακτική μέθοδος να φτάσουμε στο ίδιο αποτέλεσμα χρησιμοποιεί μία φωλιά αλφαριθμητικής και μια φωλιά αριθμητικής μεταβλητής.

```
 NEW<-
100 LET day$ = "FIRST"<-
110 LET length = LEN(day$)<-
120 PRINT length<-
RUN<-
```

Η οθόνη πρέπει να εμφανίσει:

```
 5
```

όπως πρώτα και δύο εσωτερικές φωλιές περιέχουν τις τιμές που φαίνονται:




Ας επιστρέψουμε στο πρόβλημα του μέσου μήκους λέξεων.

Γράψτε ένα πρόγραμμα που θα βρίσκει το μέσο μήκος των τριών λέξεων.

FIRST, OF, FEBRUARY

'Οταν τα προβλήματα ξεπερνούν αυτό που θεωρείτε πολύ απλοικό, είναι καλή ιδέα να κατασκευάζετε ένα σχέδιο προγράμματος πριν γράψετε το ίδιο πρόγραμμα.

**ΣΧΕΔΙΑΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ - ΠΡΟΓΡΑΜΜΑ**

[1] Αποθήκευση των τριών λέξεων σε φωλιές.

[2] Υπολογισμός των μηκών και αποθήκευσή τους.

[3] Υπολογισμός του μέσου όρου.

[4] Εκτύπωση του αποτελέσματος.

```
 NEW<-
100 LET day$ = "FIRST"<-
110 LET word$ = "OF"<-
120 LET month$ = "FEBRUARY"<-
130 LET length1 ? LEN(day$)<-
140 LET length2 ? LEN(word$)<-
150 LET length3 ? LEN(month$)
160 LET sum = length1 + length2 + length3<-
170 LET average = sum = 3<-
180 PRINT average<-
RUN<-
```

Το σύμβολο / σημαίνει διαίρεση. Η έξοδος ή το αποτέλεσμα της εκτέλεσης του προγράμματος είναι απλώς:

και έχουμε χρησιμοποιήσει οκτώ εσωτερικές φωλιές.

```
day$      FIRST                    length1    5

word$     OF                       length2    2

month$    FEBRUARY                 length3    8

                                   sum        15

                                   average    5
```

Αν νομίζετε ότι αυτό είναι μεγάλη φασαρία για ένα απλό πρόβλημα τότε μπορείτε βέβαια να το συντομεύσετε. Η πιο μικρή μορφή θα ήταν μία μόνο γραμμή αλλά δε θα ήταν τόσο εύκολο να διαβαστεί. 'Ενας λογικός συμβιβασμός χρησιμοποιεί το σύμβολο & που δηλώνει τη λειτουργία:

ένωση δύο αλφαριθμητικών μεταβλητών

Τώρα πληκτρολογήστε:

```
 NEW<-
100 LET weekday$ = "FIRST"<-
110 LET word$ = "OF"<-
120 LET month$ = "FEBRUARY"<-
130 LET phrase$ = weekday$ & words & month$<-
140 LET length = LEN(phrase$)<-
150 PRINT length / 3<-
 RUN<-
```

Το αποτέλεσμα είναι 5 όπως πριν αλλά υπάρχουν μερικά διαφορετικά εσωτερικά αποτελέσματα:

```
day$      FIRST                    length     15

word$     OF

month$    FEBRUARY

phrase$   FIRSTOFFEBRUARY
```

Υπάρχει ακόμη μία λογική απλοποίηση με τη χρήση του READ και DATA αντί για τις πρώτες τρεις εντολές LET Πληκτρολογήστε:

```
NEW<-
100 READ weekday$, word$, month$<-
110 LET phrase$   weekday$ & word$ & month$<-
```

```
120 LET length = LEN(phrase)<-
130 PRINT length / 3<-
140 DATA "FIRST","OF","FEBRUARY"<-
RUN<-
```

Τα εσωτερικά αποτελέσματα αυτής της μορφής είναι ακριβώς τα ίδια όπως της προηγούμενης. Το READ έχει αποτέλεσμα τη δημιουργία εσωτερικών φωλιών με τιμές, όπως κάνει και το LET.

**ΑΝΑΓΝΩΡΙΣΤΕΣ ΚΑΙ ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ**

Ονόματα για φωλιές όπως:

```
weekday$
word$
month$
phrase$
```

λέγονται αναγνωριστές για αλφαριθμητικές μεταβλητές. Το σύμβολο του δολλαρίου δηλώνει ότι οι φωλιές προορίζονται για σειρές χαρακτήρων. Το δολάριο πρέπει πάντα να είναι στο τέλος. Μερικοί προφέρουν αυτές τις λέξεις, "daydollar", "worddollar" κ.ο.κ.

Οι φωλιές περιστεριών αυτού του είδους λέγονται αλφαριθμητικές μεταβλητές γιατί περιέχουν μόνο σειρές χαρακτήρων που μπορεί να μεταβάλλονται καθώς ένα πρόγραμμα εκτελείται.

Τα περιεχόμενα τέτοιων φωλιών λέγονται τιμές. 'Ετσι λέξεις όπως "FIRST" και "OF" μπορεί να είναι τιμές αλφαριθμητικών μεταβλητών με ονόματα day$ και word$.

**ΤΥΧΑΙΟΙ ΧΑΡΑΚΤΗΡΕΣ**

Μπορείτε να χρησιμοποιήσετε κώδικες χαρακτήρων (βλέπε 'Εννοιες) για να δημιουργήσετε τυχαία γράμματα. Τα κεφαλαία γράμματα A-Z έχουν κώδικες από 65-90. Η σύναρτηση CHR$ μετατρέπει αυτούς τους κώδικες σε γράμματα. Το ακόλουθο πρόγραμμα θα τυπώσει το γράμμα B.

```
NEW<-
10 LET code = 66<-
20 PRINT CHR$(code)<-
RUN<-
```

Το ακόλουθο πρόγραμμα θα δημιουργήσει τριάδες γραμμάτων A, B και C μέχρι να σχηματιστεί τυχαία η λέξη CAB

```
NEW<-
10 REPeąt taxi
20   LET first$ = CHR$(RND(65 TO 67))
30   LET second$ = CHR$(RND(65 TO 67))
40   LET third$ = CHR$(RND(65 TO 67))
50   LET word$ = first$ & second$ & third?
60   PRINT !word$!
70   IF word$ = "CAB" THEN EXIT taxi
80 END REPeat taxi
```

Οι τυχαίοι χαρακτήρες όπως οι τυχαίοι αριθμοί ή τα τυχαία σημεία είναι χρήσιμα στην εκμάθηση του προγραμματισμού.

Μπορείτε εύκολα να πάρετε ενδιαφέροντα αποτελέσματα για παραδείγματα προγραμμάτων και ασκήσεις.

**ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 4**

Στο ακόλουθο τέστ το άριστα είναι 10 βαθμοί. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Τι είναι μία σειρά χαρακτήρων;

2. Ποιά είναι η συνήθης ονομασία του όρου "σειρά χαρακτήρων";

3. Τι διακρίνει το όνομα μιας αλφαριθμητικής μεταβλητής;

4. Πως πρόφεραν μερικοί τις λέξεις όπως "word$";

5. Ποιά δεσμευμένη λέξη χρησιμοποιείται για να βρούμε τον αριθμό των χαρακτήρων μιας αλφαριθμητικής μεταβλητής;

6. Ποιό σύμβολο χρησιμοποιείται για την ένωση δύο αλφαριθμητικών μεταβλητών;

7. Τα διαστήματα μπορούν να περιλαμβάνονται σε μία αλφαριθμητική μεταβλητή; Πως ξεχωρίζουν τα όριά της;

8. 'Οταν μια εντολή όπως:

```
LET meat$ = "steak"
```

εκτελεστεί, τα εισαγωγικά αποθηκεύονται;

9. Ποιά συνάρτηση θα μετατρέψει έναν κατάλληλο αριθμητικό κώδικα σε γράμμα;

10. Πως μπορείτε να δημιουργήσετε τυχαία κεφαλαία γράμματα;

**ΑΠΑΝΤΗΣΕΙΣ ΣΤΟΝ ΑΥΤΟΕΛΕΓΧΟ**

1. Μία σειρά χαρακτήρων είναι μία ακολουθία χαρακτήρων όπως γράμματα, αριθμοί, ή άλλα σύμβολα.

2. Η ονομασία της σειράς χαρακτήρων είναι αλφαριθμητική μεταβλητή.

3. Το όνομα μιας αλφαριθμητικής μεταβλητής πάντα τελειώνει με
   $.

4. Ονόματα όπως "word$" συχνά προφέρονται "worddollar".

5. Η δεσμευμένη λέξη LEN βρίσκει το μήκος ή τον αριθμό των χαρακτήρων σε μία αλφαριθμητική μεταβλητή. Για παράδειγμα αν η μεταβλητή meat$ έχει την τιμή "steak" τότε η εντολή:

```
PRIN LEN(meat$)
```

θα δώσει αποτέλεσμα 5.

6. Το σύμβολο της ένωσης δύο αλφαριθμητικών μεταβλητών είναι &

7. Τα όρια μιας αλφαριθμητικής μεταβλητής ορίζονται από εισαγωγικά ή αποστρόφους.

8. Τα εισαγωγικά δεν είναι μέρος της αλφαριθμητικής μεταβλητής και δεν αποθηκεύονται.

9. Η συνάρτηση είναι η CHR$. Πρέπει να τη χρησιμοποιείτε με παρενθέσεις όπως στο CHR$(66) ή όπως στο CHR$(RND(65 TO 67)).

10. Μπορείτε να δημιουργήσετε τυχαίους αριθμούς με εντολές όπως:

```
code = RND(65 TO 90)
PRINT CHR$(code)
```

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

- 9 ή 10 είναι πολύ καλά. Συνεχίστε να διαβάζετε.

- 7 ή 8 είναι καλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 4.

- 5 ή 6 είναι μέτρια αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου 4 και ξανακάντε το τέστ.

- κάτω από 5. Πρέπει να μελετήσετε προσεκτικά το κεφάλαιο τέσσερα και να επαναλάβετε το τέστ.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 4**

1. Αποθηκεύστε τις λέξεις "Good" και "day" σε δύο ξεχωριστές μεταβλητές. Χρησιμοποιήστε μία δήλωση LET για να ενώσετε τις δύο τιμές των δύο μεταβλητών σε μία τρίτη μεταβλητή. Τυπώστε το αποτέλεσμα.

2. Αποθηκεύστε τις ακόλουθες λέξεις σε τέσσερις ξεχωριστές φωλιές περιστεριών:

light let be there

Ενώστε τις λέξεις για να φτιάξετε μία πρόταση προσθέτοντας διαστήματα και μία τελεία. Αποθηκεύστε ολόκληρη την πρόταση σε μία μεταβλητή με όνομα sent$ και τυπώστε την πρόταση και το συνολικό αριθμό χαρακτήρων που περιέχει.

3. Γράψτε ένα πρόγραμμα που να χρησιμοποιεί τις δεσμευμένες λέξεις

CHR$(RND(65 TO 90))

για να δημιουργήσετε εκατό τυχαίες λέξεις με τρία γράμματα. Κοιτάξτε μήπως κατά λάθος δημιουργήσατε καμμία πραγματική Αγγλική λέξη. Δοκιμάστε τα αποτελέσματα των:

a. ; στο τέλος μιας εντολής PRINT
b. ! σε οποιαδήποτε πλευρά των στοιχείων που θα εκτυπωθούν.

ΚΕΦΑΛΑΙΟ 5

# ΓΝΩΣΤΗ ΚΑΛΗ ΠΡΑΚΤΙΚΗ

'Ηδη έχετε αρχίσει να δουλεύετε αποτελεσματικά με τα μικρά προγράμματα. 'Ισως να έχετε διαπιστώσει ότι οι ακόλουθες μέθοδοι βοηθούν πολύ.

[1] Χρήση μικρών γραμμάτων στους αναγνωριστές: ονόματα μεταβλητών (φωλιές) ή δομές επανάληψης κ.τ.λ.

[2] Ταυτοποίηση εντολών που δείχνει το περιεχόμενο μιας δομής επανάληψης.

[3] Καλή επιλογή αναγνωριστών που δείχνει για ποιο λόγο χρησιμοποιείται μία μεταβλητή ή μια δομή επανάληψης.

[4] Διόρθωση του προγράμματος με:

- αντικατάσταση μιας γραμμής
- εισαγωγή μιας νέας γραμμής
- διαγραφή μιας γραμμής

**ΠΡΟΓΡΑΜΜΑΤΑ ΩΣ ΠΑΡΑΔΕΙΓΜΑΤΑ**

'Εχετε φτάσει στο στάδιο όπου είναι χρήσιμο να μπορείτε να μελετάτε προγράμματα για να μαθαίνετε απ' αυτά και να προσπαθείτε να καταλαβαίνετε τι κάνουν. Η τεχνική της εκτέλεσης αυτών στην πράξη, πρέπει να έχει πια κατανοηθεί καλά και έτσι στα επόμενα κεφάλαια θα σταματήσουμε τη συνεχή επανάληψή του.

NEW  πριν από κάθε πρόγραμμα

<=  στο τέλος κάθε γραμμής

RUN  για έναρξη κάθε προγράμματος

θα καταλαβαίνετε ότι πρέπει να χρησιμοποιείτε όλα αυτά όταν θέλετε να εισάγετε και να τρέξετε ένα πρόγραμμα. Αλλά η παράλειψη τους στο κείμενο θα σας επιτρέψει να δείτε άλλες λεπτομέρειες πιο καθαρά καθώς προσπαθείτε να φανταστείτε τι θα κάνει το πρόγραμμα όταν εκτελείται. Παραλείποντας τις παραπάνω λεπτομέρειες μπορούμε να χρησιμοποιούμε και να καταλαβαίνουμε τα προγράμματα πιο εύκολα χωρίς την τεχνική σύγχιση. Για παράδειγμα, το ακόλουθο πρόγραμμα παράγει τυχαία κεφαλαία γράμματα μέχρι να εμφανιστεί ένα Z. Οι λέξεις NEW, RUN και το σύμβολο ENTER δε φαίνονται αλλά εσείς πρέπει να τις χρησιμοποιείτε.

```
100 REPeat letters
110   lettercode = RND(65 TO 90)
120   cap$ = CHR$(lettercode)
130   PRINT cap$
140   IF cap$ = "Z" THEN EXIT letters
150 END REPeat letters
```

Σ' αυτό και σε ακόλουθα κεφάλαια, τα προγράμματα θα φαίνονται χωρίς τα σύμβολα ENTER. Οι άμεσες εντολές (commands) επίσης θα φαίνονται χωρίς τα σύμβολα ENTER. Αλλά εσείς πρέπει να χρησιμοποιείτε αυτά τα πλήκτρα όπως συνήθως. Επίσης πρέπει να θυμάστε να χρησιμοποιείτε το NEW και RUN όποτε χρειάζεται.

**ΑΥΤΟΜΑΤΗ ΑΡΙΘΜΗΣΗ ΓΡΑΜΜΩΝ**

Είναι κουραστικό και ανιαρό να εισάγετε τους αριθμούς γραμμής συνέχεια. Αντί γι' αυτό πληκτρολογήστε:

AUTO.

και πριν αρχίσετε να προγραμματίζετε, το QL θα απαντήσει μ' έναν αριθμό γραμμής:

100

Συνεχίστε να πληκτρολογείτε γραμμές μέχρι να τελειώσετε το πρόγραμμά σας η οθόνη θα δείχνει:

```
100 PRINT "First"
110 PRINT "Second"
120 PRINT END
```

Για να τερματίσετε την αυτόματη παραγωγή αριθμών γραμμής χρησιμοποιήστε το BREAK. Κρατήστε πατημένο το CTRL και πιέστε το SPACE. Αυτό θα παράγει το μήνυμα:

130 not coplete

και η γραμμή 130 δε θα συμπεριληφθεί στο πρόγραμμά σας.

Αν κάνετε ένα λάθος που δεν προκαλεί την έξοδο από την αυτόματη αρίθμηση, μπορείτε να συνεχίσετε και να διορθώσετε (EDIT) τη γραμμή αργότερα. Αν κάνετε συντακτικό λάθος θα συμβεί BREAK που ακολουθείται από μήνυμα λάθους. Αν μετά θέλετε να συνεχίσετε από τη γραμμή 110, ας πούμε, πληκτρολογήστε:

AUTO 110

και συνεχίστε όπως πριν.
Αν θέλετε να αρχίσετε από κάποιο συγκεκριμένο αριθμό γραμμής, ας πούμε 600, και να χρησιμοποιήσετε σα βήμα κάτι άλλο εκτός από 10 μπορείτε να πληκτρολογήσετε για βήμα 5:

AUTO 600,5

Οι γραμμές τότε θα αριθμούνται 600, 605, 610 κ.τ.λ.

## ΔΙΟΡΘΩΣΗ ΜΙΑΣ ΓΡΑΜΜΗΣ

Για να διορθώσετε μία γραμμή απλώς πληκτρολογείστε EDIT ακολουθούμενο από τον αριθμό γραμμής, για παράδειγμα:

```
EDIT 110
```

Η γραμμή θα απεικονιστεί με το δείκτη στο τέλος έτσι:

```
110 PRINT "First"
```

Μπορείτε να μετακινήσετε το δείκτη χρησιμοποιώντας:

```
<= μία θέση αριστερά
=> μία θέση δεξιά
```

Για να σβήσετε ένα χαρακτήρα προς τα δεξιά χρησιμοποιήστε:

```
CTRL μαζί με <=
```

Για να σβήσετε το χαρακτήρα στη θέση του δείκτη πατήστε:

```
CTRL μαζί με =>
```

και ο χαρακτήρας στα δεξιά του δείκτη θα κινηθεί και θα καλύψει το κενό.

## ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΑ MICRODRIVES

Πριν χρησιμοποιήσετε μια καινούργια μικροκασέτα πρέπει να φορμαριστεί. Ακολουθήστε τις οδηγίες της Εισαγωγής του Οδηγού Χρήσης. Η επιλογή του ονόματος της μικροκασέτας ακολουθεί τους ίδιους κανόνες όπως οι αναγνωριστές της SuperBASIC; αλλά περιορίζεται σε 10 μόνο χαρακτήρες. Είναι καλή ιδέα να γράψετε το όνομα της μικροκασέτας πάνω της χρησιμοποιώντας μία από τις αυτοκόλλητες ετικέτες που δίνονται με τη μικροκασέτα.

Πρέπει να κρατάτε πάντα τουλάχιστον ένα αντίγραφο υποστήριξης κάθε προγράμματος ή δεδομένων, ακολουθήστε τις οδηγίες του τμήματος πληροφοριών στο τέλος του Οδηγού Χρήσης.

ΠΡΟΣΟΧΗ

Αν φορμάρετε μία μικροκασέτα που περιέχει προγράμματα και/ή δεδομένα τότε ΟΛΑ τα προγράμματα και/ή τα δεδομένα θα χαθούν.

## ΣΩΖΟΝΤΑΣ ΠΡΟΓΡΑΜΜΑΤΑ

Το ακόλουθο πρόγραμμα τοποθετεί περιθώρια πλάτους 8 pixels, σε κόκκινο (κώδικας 2) σε τρία περιθώρια, ορισμένα # 0, # 1, # 2.

```
10 REMark Border
20 FOR k = 0 TO 2 : BORDER k,8,2
```

Μπορείτε να σώσετε σε μία μικροκασέτα βάζοντας την στη θέση 1 πληκτρολογώντας

SAVE mdv_bord

Το πρόγραμμα θα σωθεί στη μικροκασέτα με το όνομα bord.

**ΕΛΕΓΧΟΝΤΑΣ ΜΙΑ ΜΙΚΡΟΚΑΣΕΤΑ**

Αν θέλετε να μάθετε τι προγράμματα ή αρχεία δεδομένων υπάρχουν σε μία συγκεκριμένη μικροκασέτα, τοποθετήστε την στη θέση μικροκασέτας και πληκτρολογήστε:

DIR mdv1_

Ο κατάλογος θα εμφανιστεί στην οθόνη. Αν η μικροκασέτα είναι στη θέση 2 τότε πληκτρολογήστε:

DIR mdv2_

**ΑΝΤΙΓΡΑΦΩΝΤΑΣ ΠΡΟΓΡΑΜΜΑΤΑ**

Από τη στιγμή που ένα πρόγραμμα έχει αποθηκευτεί σαν αρχείο σε μία μικροκασέτα, μπορεί να αντιγραφεί. Αυτός είναι ένας τρόπος να κάνετε αντίγραφο υποστήριξης μιας μικροκασέτας. Μπορείτε να αντιγράψετε όλο το προηγούμενο πρόγραμμα σε μια άλλη μικροκασέτα τοποθετημένη στη θέση 2 πληκτρολογώντας:

COPY mdv1_bord TO mdv2_bord

**ΔΙΑΓΡΑΦΟΝΤΑΣ ΕΝΑ ΑΡΧΕΙΟ**

'Ενα αρχείο είναι οτιδήποτε, όπως πρόγραμμα ή δεδομένα, αποθηκευμένο στη μικροκασέτα. Για να διαγράψετε ένα πρόγραμμα που λέγεται prog πληκτρολογήστε:

DELETE MDV1_prog

**ΦΟΡΤΩΝΟΝΤΑΣ ΠΡΟΓΡΑΜΜΑΤΑ**

'Ενα πρόγραμμα μπορεί να φορτωθεί από μία μικροκασέτα πληκτρολογώντας:

LOAD mdv2_bord

Αν το πρόγραμμα φορτωθεί σωστά, θα αποδείξει ότι και τα δύο αντίγραφα είναι σωστά. Μπορείτε να δοκιμάσετε το πρόγραμμα χρησιμοποιώντας:

LIST  για να δείτε τη λίστα εντολών

RUN  για να το εκτελέσετε

Αντί να χρησιμοποιείτε το LOAD ακολουθούμενο από το RUN, μπορείτε να συνδυάσετε τις δύο λειτουργίες σε μία εντολή:

LRUN  MDV2_BORD

Το πρόγραμμα θα φορτωθεί και θα εκτελεστεί αμέσως.

## ΣΥΓΧΩΝΕΥΟΝΤΑΣ ΠΡΟΓΡΑΜΜΑΤΑ

Υποθέτουμε ότι έχετε δύο προγράμματα αποθηκευμένα σε μικροκασέτα στη θέση 1 με ονόματα prog1 και prog2.

```
1ο πρόγραμμα 10 PRINT "First",
2ο πρόγραμμα 20 PRINT "Second"
```

Αν πληκτρολογήσετε

**LOAD mdv1_prog1**

ακολουθούμενο από

**MERGE mdv1_prog2**

Τότε τα δύο προγράμματα θα συγχωνευτούν σε ένα. Για να το επαληθεύσετε πληκτρολογήστε LIST και πρέπει να δείτε:

```
10 PRINT "First"
20 PRINT "Second"
```

Αν πρόκειται να συγχωνεύσετε ένα πρόγραμμα βεβαιωθείτε ότι όλοι οι αριθμοί εντολών είναι διαφορετικοί από αυτούς του προγράμματος που υπάρχει ήδη στη κυρία μνήμη. Αλλιώς θα σβήσει μερικές από τις γραμμές του πρώτου προγράμματος. Αυτή η ευκολία αποκτά μεγάλη αξία καθώς γίνεστε ικανότεροι στο χειρισμό των διαδικασιών. Τότε είναι πολύ φυσικό να φτιάχνετε ένα πρόγραμμα προσθέτοντας διαδικασίες ή λειτουργίες σ' αυτό.

## ΓΕΝΙΚΑ

Να είστε προσεκτικοί και μεθοδικοί με τις μικροκασέτες. Πάντα να κρατάτε ένα αντίγραφο υποστήριξης και αν υποπτευθείτε κάποιο πρόβλημα σε μία μικροκασέτα ή στη μονάδα μικροκασέτας τότε κρατήστε άλλο ένα αντίγραφο υποστήριξης. Οι επαγγελματίες στους υπολογιστές πολύ σπάνια χάνουν δεδομένα. Γνωρίζουν ότι ακόμη και τα καλύτερα μηχανήματα ή συσκευές θα κάνουν περιστασιακά λάθη και είναι προετοιμασμένοι γι' αυτό.

Αν θέλετε να ονομάσετε ένα πρόγραμμα μ' ένα συγκεκριμένο όνομα, ας πούμε square, τότε ίσως είναι καλή ιδέα να χρησιμοποιείτε ονόματα όπως sq1, sq2, ... για τις προηγούμενες κόπιες. 'Οταν το πρόγραμμα είναι στην τελική του μορφή κάντε τουλάχιστο δύο αντίγραφα με όνομα square και τα άλλα μπορούν να σβηστούν με ξαναφορμάρισμα ή με κάποια πιο επιλεκτική μέθοδο.

## ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 5

Στο ακόλουθο τέστ το άριστα είναι 14 βαθμοί. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Γιατί τα μικρά γράμματα είναι προτιμότερα για τις λέξεις του προγράμματος που διαλέγετε εσείς;

2. Ποιος είναι ο σκοπός της ταυτοποίησης;

3. Τι πρέπει να οδηγεί φυσιολογικά την επιλογή αναγνωριστών για μεταβλητές και βρόχους;

4. Ονομάστε τρεις τρόπους διόρθωσης ενός προγράμματος που βρίσκεται στη κύρια μνήμη του υπολογιστή.

5. Τι πρέπει να θυμάστε να πληκτρολογείτε στο τέλος κάθε εντολής ή γραμμής προγράμματος όταν την εισάγετε;

6. Τι πρέπει να πληκτρολογήσετε φυσιολογικά, πριν αρχίσετε εισαγωγή ενός προγράμματος από το πληκτρολόγιο;

7. Τι πρέπει να υπάρχει στην αρχή κάθε γραμμής που πρόκειται να αποθηκευτεί σαν μέρος ενός προγράμματος;

8. Τι πρέπει να θυμάστε να πληκτρολογείτε για να εκτελεστεί ένα πρόγραμμα;

9. Ποιά δεσμευμένη λέξη σας επιτρέπει να βάλετε σ' ένα πρόγραμμα πληροφορίες που δεν έχουν κανένα αποτέλεσμα κατά την εκτέλεση;

10. Ποιες δύο δεσμευμένες λέξεις σας βοηθούν να αποθηκεύσετε προγράμματα και να τα επανακτάτε από μικροκασέτες;

## ΑΠΑΝΤΗΣΕΙΣ ΣΤΟΝ ΑΥΤΟΕΛΕΓΧΟ ΤΟΥ ΚΕΦΑΛΑΙΟΥ 5

1. Τα μικρά γράμματα στα ονόματα των μεταβλητών ή των βρόχων φαίνονται καλύτερα εξ αιτίας της αντίθεσης με τις δεσμευμένες λέξεις που κατά ένα μέρος τουλάχιστον, απεικονίζονται σε κεφαλαία.

2. Η ταυτοποίηση δείχνει καθαρά ποιο είναι το περιεχόμενο των βρόχων (και άλλων δομών).

3. Οι αναγνωριστές (ονόματα) πρέπει φυσιολογικά να διαλέγονται έτσι ώστε να σημαίνουν κάτι, για παράδειγμα count ή word$ αντί για C ή W$.

4. Μπορείτε να διορθώσετε ένα πρόγραμμα:

   - αντικαθιστώντας μία γραμμή
   - τοποθετώντας μία νέα γραμμή
   - διαγράφοντας μία γραμμή

5. Το πλήκτρο ENTER πρέπει να χρησιμοποιείται για την εισαγωγή μιας εντολής ή γραμμής προγράμματος.

6. Η λέξη NEW θα σβήσει το προηγούμενο πρόγραμμα της SuperBASIC στο QL και έτσι το νέο πρόγραμμα που θα εισάγετε δε θα συγχωνευθεί με κάποιο παλιό.

7. Αν θέλετε να αποθηκεύσετε μία γραμμή σαν μέρος ενός προγράμματος τότε πρέπει να χρησιμοποιήσετε αριθμό γραμμής.

8. Η λέξη RUN ακολουθούμενη από <= θα προκαλέσει την εκτέλεση ενός προγράμματος.

9. Η λέξη REMark σας επιτρέπει να βάλετε σ' ένα πρόγραμμα πληροφορίες που θα αγνοηθούν τη στιγμή της εκτέλεσης.

10. Οι δεσμευμένες λέξεις SAVE και LOAD σας επιτρέπουν να αποθηκεύετε προγράμματα και να τα επανακτάτε από μικροκασέτες.

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

- 12 ως 14 είναι πολύ καλά. Συνεχίστε να διαβάζετε.
- 1ή 11 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου πέντε.
- 8 ή 9 είναι μέτρια αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου πέντε και ξανακάντε το τέστ.
- Κάτω από 8. Πρέπει να μελετήσετε προσεκτικά το κεφάλαιο πέντε και να επαναλάβετε το τέστ.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 5**

1. Ξαναγράψτε το ακόλουθο πρόγραμμα χρησιμοποιώντας μικρά γράμματα για μία καλύτερη παρουσίαση. Προσθέστε τις λέξεις NEW και RUN. Χρησιμοποιήστε αριθμούς γραμμής και το σύμβολο ENTER όπως όταν εισάγετε και τρέχετε ένα πρόγραμμα. Χρησιμοποιήστε REMark για να δώσετε στο πρόγραμμα ένα όνομα.

```
LET TWO$ = "TWO"
LET FOUR$ = "FOUR"
LET SIX$ = TWO$ & FOUR$
PRINT LEN(SIX$)
```

Εξηγείστε πως το two (2) και το four (4) παράγουν το 7.

2. Χρησιμοποιήστε ταυτοποίηση, μικρά γράμματα, NEW, RUN, αριθμούς γραμμής και το σύμβολο ENTER για να δείξετε πως κανονικά θα εσάγατε και θα τρέχατε το παρακάτω πρόγραμμα:

```
REPeat loop
letter_code = RND(65 TO 90)
LET letters$ = CHR$(letter_code)
PRINT letter$
IF letter$ = Z' THEN EXIT loop
END REPeat loop
```

3. Ξαναγράψτε το ακόλουθο πρόγραμμα με καλύτερο στυλ, χρησιμοποιώντας ονόματα μεταβλητών με νόημα και καλή παρουσίαση. Γράψτε το πρόγραμμα όπως θα το εισάγατε.

```
LET S = 0
REPeat TOTAL
LET N = RND(1 TO 6)
PRINT N !
LET S = S + N
IF n = 6 THEN EXIT TOTAL
END REPeat TOTAL
PRINT S
```

4. Αποφασίστε τι κάνει το πρόγραμμα και μετά εισάγετε το και εκτελέστε το για να ελέγξετε αν το βρήκατε σωστά.

# ΚΕΦΑΛΑΙΟ 6
# ΠΙΝΑΚΕΣ ΚΑΙ ΒΡΟΧΟΙ FOR

## ΤΙ ΕΙΝΑΙ ΕΝΑΣ ΠΙΝΑΚΑΣ

Γνωρίζετε ότι αριθμοί ή σειρές χαρακτήρων μπορούν να γίνουν τιμές μεταβλητών. Μπορείτε να παραστήσετε αυτό με εσωτερικές φωλιές ή σπίτια όπου πηγαίνουν οι αριθμοί ή οι λέξεις. Υποθέτουμε για παράδειγμα ότι τέσσερις υπάλληλοι μιας εταιρείας πρόκειται να σταλούν σ' ένα μικρό χωριό επειδή ίσως έχει βρεθεί εκεί πετρέλαιο. Το χωριό είναι ένα από τα λίγα μέρη όπου μόνο τα σπίτια έχουν ονόματα και υπάρχουν τέσσερα σπίτια διαθέσιμα για νοικίασμα. 'Ολα τα ονόματα των σπιτιών τελειώνουν μ' ένα σύμβολο δολαρίου.

Westlea$ Lakeside$ Roselawn$ Oaktree$

Οι τέσσερις υπάλληλοι λέγονται:

VAL HAL MEL DEL

Μπορούν να τοποθετηθούν στα σπίτια με μία από τις δύο μεθόδους:

**ΠΡΟΓΡΑΜΜΑ 1**

```
100 LET westlea$ = "VAL"
110 LET lakeside$ = "HAL"
120 LET roselawn$ = "MEL"
130 LET oaktree$ = "DEL"
140 PRINT ! westlea$ ! lakeside$ ! roselawn$ ! oaktree$ !
```

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 READ westlea$, lakeside$, roselawn$, oaktree$
110 PRINT ! westlea$ ! lakeside$ ! roselawn$ ! oaktree$ !
120 DATA "VAL", "HAL", "MEL", "DEL"
```

```
westlea$ lakeside$ roselawn$ oaktree$
   ¦        ¦         ¦         ¦
   V        V         V         V
  VAL      HAL       MEL       DEL
```

Καθώς το ποσό των δεδομένων αυξάνει, τα πλεονεκτήματα των READ και DATA απέναντι στο LET γίνονται μεγαλύτερα. Αλλά όταν τα δεδομένα γίνουν πραγματικά πολλά τότε το πρόβλημα της εύρεσης ονομάτων για σπίτια γίνεται τόσο δύσκολο όσο η ανεύρεση άδειων σπιτιών σ' ένα μικρό χωριό.

Η λύση σ' αυτό και σε πολλά άλλα προβλήματα χειρισμού δεδομένων βρίσκεται σ' ένα νέο τύπο φωλιάς ή μεταβλητής όπου πολλοί μπορούν να μοιραστούν ένα μόνο όνομα. 'Ομως πρέπει να

ξεχωρίζουν, γι αυτό κάθε μεταβλητή έχει επίσης έναν αριθμό, όπως τα αριθμημένα σπίτια στον ίδιο δρόμο. Υποθέτουμε ότι χρειάζεστε τέσσερα άδεια σπίτια στη High Street με αριθμούς από 1 ως 4. Στη SuperBASIC λέμε ότι αυτό είναι ένας πίνακας τεσσάρων σπιτιών. Το όνομα του πίνακα είναι high_st$ και τα τέσσερα σπίτια πρόκειται να αριθμηθούν από 1 ως 4.

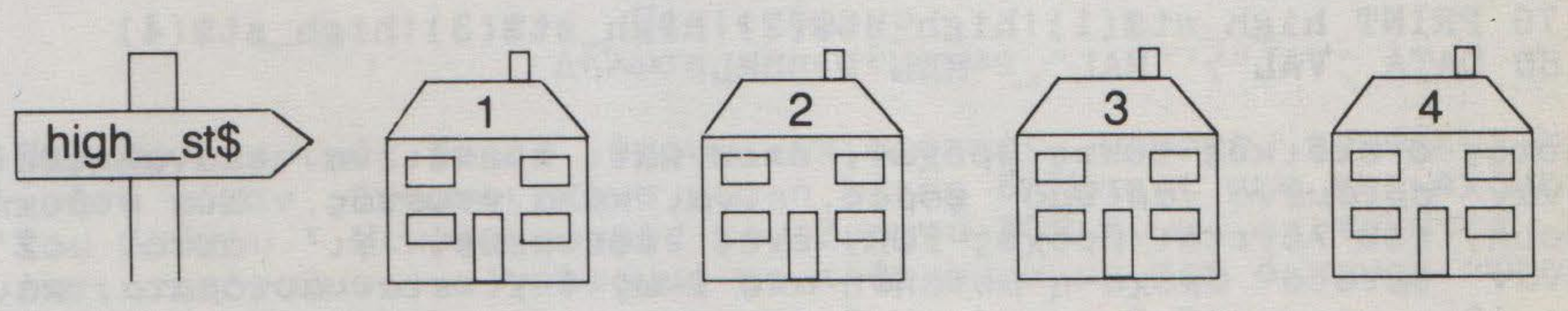
'Ομως δεν μπορείτε να χρησιμοποιήσετε αυτές τις μεταβλητές του πίνακα όπως τις κανονικές (απλές) μεταβλητές. Πρέπει να καθορίσετε τις διαστάσεις (το μέγεθος) του πίνακα πρώτα. Ο υπολογιστής κατανέμει χώρο εσωτερικά και πρέπει να ξέρει πόσες αλφαριθμητικές μεταβλητές υπάρχουν σ' έναν πίνακα και επίσης το μέγιστο μήκος κάθε αλφαριθμητικής μεταβλητής. Χρησιμοποιήστε μία εντολή DIM έτσι:

```
DIM high_st$(4,3)
```

μέγιστο μήκος αλφαριθμητικής μεταβλητής

αριθμός αλφα - ριθμητικών με- ταβλητών

Μετά την εκτέλεση της εντολής DIM, οι μεταβλητές είναι διαθέσιμες για χρήση. Είναι σαν να κτίστηκαν τα σπίτια αλλά είναι ακόμη άδεια. Τα τέσσερα "σπίτια" μοιράζονται ένα κοινό όνομα, το high_st$, αλλά το καθένα έχει το δικό του αριθμό και μπορεί να κρατήσει ως τρεις χαρακτήρες.




Υπάρχουν πέντε προγράμματα παρακάτω και όλα κάνουν το ίδιο πράγμα: κάνουν τα τέσσερα "σπίτια" να "κατοικηθούν" και εκτυπώνουν για να δείξουν ότι πράγματι τα σπίτια "κατοικήθηκαν". Η τελική μέθοδος χρησιμοποιεί μόνο τέσσερις γραμμές αλλά οι άλλες τέσσερις οδηγούν σ' αυτή με μία μετακίνηση από παλιές ιδέες σε καινούργιες ή καινούργιες χρήσεις παλιών ιδεών. Η μετακίνηση γίνεται επίσης προς μεγαλύτερη οικονομία.

Αν καταλαβαίνετε τις πρώτες δύο ή τρεις μεθόδους τέλεια, τότε ίσως προτιμάτε να πάτε κατ' ευθείαν στις μεθόδους 4 και 5. Αλλά αν έχετε αμφιβολίες, οι μέθοδοι 1, 2 και 3 θα ξεκαθαρίσουν τα πράγματα.

**ΠΡΟΓΡΑΜΜΑ 1**

```
100 DIM high_st$(4,3)
110 LET high_st$(1) = "VAL"
```

```
120 LET high_st$(2) = "HAL"
130 LET high_st$(3) = "MEL"
140 LET high_st$(4) = "DEL"
150 PRINT ! high_st$(1) ! high_st$(2)!
160 PRINT ! high_st$(3) ! high_st$(4)!
```

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 DIM high_st$(4,3)
110 READ high_st$(1),high_st$(2),high_st$(3),high_st$(4)
120 PRINT ! high_st$(1) ! high_st$(2)!
130 PRINT ! high_st$(3) ! high_st$(4)!
140 DATA "VAL", "HAL", "MEL", "DEL"
```

Αυτό δείχνει πως να εξοικονομήσετε ονόματα μεταβλητών αλλά η συνεχής επανάληψη του high_st$ είναι κουραστική και προκαλεί την ακατάστατη όψη των προγραμμάτων. Μπορούμε πάλι να χρησιμοποιήσουμε μία γνωστή τεχνική το βρόχο REPeat - για να βελτιώσουμε τα πράγματα. Δημιουργούμε ένα μετρητή, το number, που αυξάνει κατά ένα και ο βρόχος REPeat συνεχίζεται.

**ΠΡΟΓΡΑΜΜΑ 3**

```
95 RESTORE 180
100 DIM high_st$(4,3)
110 LET number = 0
120 REPeat houses
130   LET number = number + 1
140   IF number = 5 THEN EXIT houses
150   READ high_st$(number)
160 END REPeat houses
170 PRINT high_st$(1)!high_st$(2)!high_st$(3)!high_st$(4)
180 DATA "VAL", "HAL", "MEL", "DEL"
```

Αυτός ο ειδικός τύπος βρόχων, όπου κάτι πρέπει να επαναληφθεί έναν ορισμένο αριθμό φορές, είναι καλά γνωστός. Μία ειδική δομή, που λέγεται βρόχος FOR, έχει εφευρεθεί γι' αυτό. Σ' έναν τέτοιο βρόχο η μέτρηση από 1 ως 4 γίνεται αυτόματα, και το ίδιο και η έξοδος από το βρόχο όταν και τα τέσσερα στοιχεία έχουν χειριστεί.

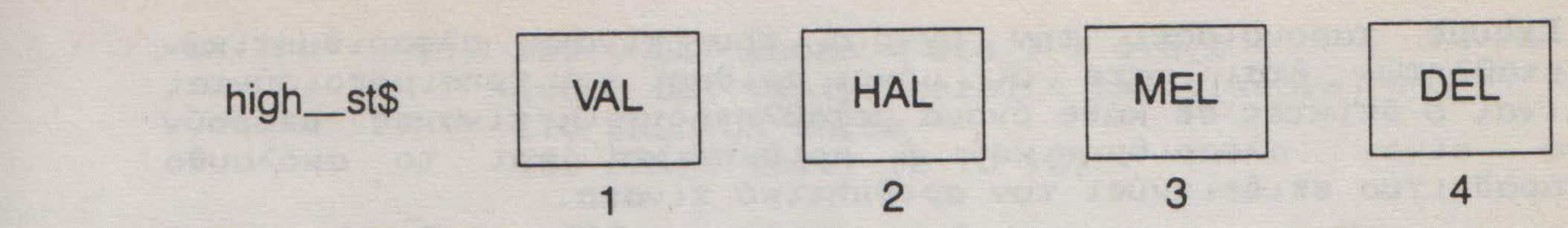
**ΠΡΟΓΡΑΜΜΑ 4**

```
100 DIM high_st$(4,3)
110 FOR number = 1 TO 4
120   READ high_st$(number)
130   PRINT ! high_st$(number)!
140 END FOR number
150 DATA "VAL","HAL","MEL","DEL"
```

Το αποτέλεσμα και από τα τέσσερα προγράμματα είναι το ίδιο:

VAL HAL MEL DEL

και δείχνει ότι τα δεδομένα έχουν αποθηκευτεί εσωτερικά σωστά στις τέσσερεις μεταβλητές του πίνακα:




Η μέθοδος 4 είναι σαφώς η καλύτερη ως εδώ, επειδή μπορεί να χειριστεί με τον ίδιο τρόπο 4 ή 40 ή 400 στοιχεία απλώς με την αλλαγή του αριθμού 4 και την πρόσθεση περισσοτέρων στοιχείων με εντολές DATA. Μπορείτε να χρησιμοποιήσετε οσεσδήποτε δηλώσεις DATA χρειάζεστε.

Στην απλούστερη μορφή, ο βρόχος FOR είναι σαν την απλούστερη μορφή του βρόχου REPeat. Οι δύο βρόχοι μπορούν να συγκριθούν.

```
100 REPeat greeting          100 FOR greeting 100 TO 130
110   PRINT "Hello"          110   PRINT "Hello"
120 END REPeat greeting      120 END FOR greeting
```

Και οι δύο βρόχοι θα δουλέψουν. Ο βρόχος REPeat θα τυπώνει συνεχώς "Hello" (σταματήστε το με το BREAK ) και ο βρόχος FOR θα τυπώσει "Hello" μόνο τριάντα φορές.

Σημειώστε ότι το όνομα του βρόχου FOR είναι επίσης μεταβλητή, το greeting, που η τιμή της μεταβάλλεται από 100 ως 130 κατά το τρέξιμο του προγράμματος. Αυτή η μεταβλητή συχνά λέγεται μεταβλητή βρόχου ή μεταβλητή ελέγχου του βρόχου.

Σημειώστε ότι η δομή και των δύο βρόχων παίρνει τη μορφή:

Δήλωση ανοίγματος
Περιεχόμενο
Δήλωση κλεισίματος

Πάντως, μερικές δομές έχουν επιτρεπτές μικρές μορφές, για χρήση όταν υπάρχει μόνο μία ή λίγες δηλώσεις στο περιεχόμενο του βρόχου. Μικρές μορφές του βρόχου FOR είναι επιτρεπτές κι έτσι μπορούμε να γράψουμε το πρόγραμμα με τον οικονομικώτερο τρόπο.

ΠΡΟΓΡΑΜΜΑ 5

```
100 DIM high_st$(4,3)
110 FOR number = 1 TO 4 : READ high_st$(number)
110 FOR number = 1 TO 4 : PRINT ! high_st$(number)
 40 DATA "VAL", "HAL", "MEL", "DEL"
```

Οι άνω-κάτω τελείες χρησιμεύουν σα σύμβολα λήξης της εντολής αντί για το ENTER και το ENTER στις γραμμές 110, 120 χρησιμεύουν σα δηλώσεις END FOR.

Υπάρχει ένα ακόμη συντομότερος τρόπος να γράψουμε το πρόγραμμα 5. Για να τυπώσουμε τα περιεχόμενα του πίνακα high_st$ μπορούμε να αντικαταστήσουμε τη γραμμή 120 με:

```
110 PRINT high_st$
```

Αυτό χρησιμοποιεί έναν τμήμα πίνακα, που θα συζητηθεί αργότερα, στο κεφάλαιο 13.

'Εχουμε παρουσιάσει την έννοια του πίνακα αλφαριθμητικών μεταβλητών έτσι ώστε οι μόνοι αριθμοί που χρησιμοποιούνται είναι ο δείκτες σε κάθε όνομα μεταβλητής. Οι πίνακες μπορούν να είναι αλφαριθμητικοί ή αριθμητικοί και το ακόλουθο παράδειγμα επιδεικνύει τον αριθμητικό πίνακα.

Προσομοιάστε το ρίξιμο δύο ζαριών (όπως στο παιχνίδι Monopoly) τετρακόσιες φορές. Κρατήστε μία σημείωση για τον αριθμό εμφανίσεων κάθε πιθανού αποτελέσματος από 2 ως 12.

**ΠΡΟΓΡΑΜΜΑ 1**

```
100 REMark DICE1
110 LET two = 0:three = 0:four = 0:five = 0:six = 0
120 LET seven = 0:eight = 0:nine = 0:ten = 0:eleven = 0:twelve = 0
130 FOR throw = 1 TO 400
140    LET die1 = RND(1 TO 6)
150    LET die2 = RND(1 TO 6)
160    LET score = die1 + die2
170    IF score = 2 THEN LET two = two + 1
180    IF score = 3 THEN LET three = three + 1
190    IF score = 4 THEN LET four = four + 1
200    IF score = 5 THEN LET five = five + 1
210    IF score = 6 THEN LET six = six + 1
220    IF score = 7 THEN LET seven = seven + 1
230    IF score = 8 THEN LET eight = eight + 1
240    IF score = 9 THEN LET nine = nine + 1
250    IF score = 10 THEN LET ten = ten + 1
260    IF score = 11 THEN LET eleven = eleven + 1
270    IF score = 12 THEN LET twelve = twelve + 1
280 END FOR throw
290 PRINT ! two ! three ! four ! five ! six
300 PRINT ! seven ! eight ! nine ! ten ! eleven ! twelve
```

Στο παραπάνω πρόγραμμα ορίζουμε έντεκα απλές μεταβλητές για να αποθηκεύσουμε το λογαριασμό των αποτελεσμάτων. Αν απεικονίσετε γραφικά τα αποτελέσματα που τυπώθηκαν στο τέλος θα βρείτε ότι το ιστόγραμμα είναι σχεδόν τριγωνικό. Τα ψηλότερα αποτελέσματα είναι αυτά που αντιστοιχούν στο έξη, επτά και οκτώ και τα χαμηλότερα σ' αυτά του δύο και δώδεκα. 'Οπως γνωρίζει κάθε παίκτης της Monopoly αυτό αντανακλά τη συχνότητα του μέσου όρου των αποτελεσμάτων (έξη, επτά, οκτώ) και τη σπανιότητα του δύο και του δώδεκα.

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 REMark Dice2
110 DIM tally(12)
120 FOR throw = 1 TO 400
130    LET die_1 = RND(1 TO 6)
140    LET die_2 = RND(1 TO 6)
```

```
150   LET score = die_1 + die_2
160   LET tally(score) = tally(score) + 1
170 END FOR throw
180 PRINT tally (2 TO 12) !
```

Στον πρώτο βρόχο FOR τον throw, ο δείκτης της μεταβλητής του πίνακα είναι το score. Αυτό σημαίνει ότι ο σωστός δείκτης του πίνακα διαλέγεται αυτόματα για αύξηση του λογαριασμού μετά από κάθε ρίξιμο. Μπορείτε να φαντάζεστε τον πίνακα tally σα μία ομάδα από φωλιές αριθμημένες από 2 ως 12. Κάθε φορά που ένα συγκεκριμένο αποτέλεσμα έρχεται, τότε ο λογαριασμός αυτού του αποτελέσματος αυξάνεται πετώντας μία πέτρα στην ανάλογη φωλιά.

Στο δεύτερο βρόχο FOR (μικρή μορφή) ο δείκτης είναι ο number. Καθώς η τιμή του number αλλάζει από 2 ως 12 όλες οι τιμές των αθροισμάτων τυπώνονται.

Σημειώστε ότι στη εντολή DIM για έναν αριθμητικό πίνακα χρειάζεται να δηλώσετε μόνο τον αριθμό των μεταβλητών που απαιτούνται. Δεν υπάρχει θέμα μέγιστου μήκους όπως υπάρχει σε έναν αλφαριθμητικό πίνακα. Αν έχετε χρησιμοποιήσει άλλες εκδόσεις της BASIC τότε θα αναρρωτιέστε τι απέγινε η εντολή NEXT. 'Ολες οι δομές της SuperBASIC τελειώνουν με μία εντολή END. Αυτό είναι συνεπές και λογικό αλλά η εντολή NEXT παίζει κάποιο ρόλο όπως θα δείτε σε επόμενα κεφάλαια.

## ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 6

Στο ακόλουθο τέστ το άριστα είναι 16 βαθμοί. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Αναφέρετε δύο δυσκολίες που προκύπτουν όταν τα δεδομένα που χρειάζονται για ένα πρόγραμμα είναι πολλά και προσπαθείτε να τα χειριστήτε χωρίς πίνακες.

2. Σε έναν πίνακα, αν δέκα μεταβλητές έχουν το ίδιο όνομα τότε πως ξέρετε ποια είναι ποια;

3. Ποιά είναι η λέξη για τον αριθμό που καθορίζει τον αριθμό των μεταβλητών σ' έναν πίνακα; Πως ορίζουμε έναν πίνακα;

4. Ποιά είναι η λέξη για τον αριθμό που καθορίζει μια συγκεκριμένη μεταβλητή από τις άλλες που μοιράζονται το ίδιο όνομα;

5. Μπορείτε να σκεφτείτε δύο ιδέες στην καθημερινή ζωή που να ανταποκρίνονται στην ιδέα του πίνακα στον προγραμματισμό;

6. Σ' ένα βρόχο REPeat η διαδικασία τελειώνει όταν καποιος όρος προκαλέσει την εκτέλεση μιας εντολής EXIT. Τι προκαλεί τη διαδικασία σ' ένα βρόχο FOR να τελειώσει;

7. 'Ενας βρόχος REPeat χρειάζεται όνομα ώστε να γίνεται EXIT στο τέλος του σωστά. 'Ενας βρόχος FOR χρειάζεται επίσης όνομα αλλά ποια άλλη λειτουργία κανει το όνομα ενός βρόχου FOR;

8. Ποιες είναι οι δύο φράσεις που χρησιμοποιούνται για την περιγραφή της μεταβλητής που είναι επίσης το όνομα ενός βρόχου FOR;

9. Οι τιμές της μεταβλητής βρόχου αλλάζουν αυτόματα καθώς ένας βρόχος FOR εκτελείται. Αναφέρετε μία δυνατή σημαντική χρήση αυτών των τιμών.

10. Ποια από τα ακόλουθα είναι κοινά στις μεγάλες μορφές των βρόχων REPeat και FOR; Για καθένα από τα τέσσερα θέματα αναφέρετε σε ποιο τύπο βρόχου συναντάται.

    a. Εντολή ή δεσμευμένη λέξη ανοίγματος.

    b. Εντολή ή δεσμευμένη λέξη κλεισίματος.

    c. 'Ονομα βρόχου.

    d. Μεταβλητή βρόχου ή μεταβλητή ελέγχου.

**ΑΠΑΝΤΗΣΕΙΣ ΣΤΟΝ ΑΥΤΟΕΛΕΓΧΟ**

1. Είναι δύσκολο να βρείτε πολλά διαφορετικά ονόματα μεταβλητών για την αποθήκευση των δεδομένων. Αν μπορέσετε να βρείτε αρκετά ονόματα, καθένα πρέπει να γραφεί σε μία εντολή LET ή READ αν δε χρησιμοποιείται πίνακες.

2. 'Ενας αριθμός που λέγεται δείκτης, είναι μέρος κάθε ονόματος μεταβλητής πίνακα. 'Ολες οι μεταβλητές σ' έναν πίνακα μοιράζονται το ίδιο όνομα αλλά καθεμιά έχει διαφορετικό δείκτη.

3. Πρέπει να ορίσετε έναν πίνακα δίνοντας το μέγεθος του (τη διάστασή του) σε μία εντολή DIM που συνήθως μπαίνει στην αρχή του προγράμματος.

4. Ο αριθμός που ξεχωρίζει ένα στοιχείο πίνακα λέγεται δείκτης.

5. Τα σπίτια ενός δρόμου έχουν το ίδιο όνομα δρόμου αλλά καθένα έχει τον αριθμό του.

   Τα κρεβάτια ενός θαλάμου νοσοκομείου μοιράζονται τον ίδιο αριμό θαλάμου αλλά κάθε κρεβάτι μπορεί να είναι αριθμημένο.

   Οι τρύπες σ' ένα γήπεδο γκολφ π.χ. η πέμπτη τρύπα του Royal Birkdale.

6. 'Ενας βρόχος FOR τελειώνει όταν η διαδικασία που αναφέρεται στην τελευταία τιμή της μετλητής βρόχου έχει συντελεστεί.

7. Το όνομα ενός βρόχου FOR είναι επίσης το όνομα της μεταβλητής που ελέγχει το βρόχο.

8. Οι δύο φράσεις γι αυτή τη μεταβλητή είναι "μεταβλητή βρόχου" ή "μεταβλητή ελέγχου".

9. Οι τιμές της μεταβλητής ελέγχου μπορούν να χρησιμοποιηθούν σα δείκτες για ονόματα μεταβλητών πίνακα. 'Ετσι καθώς ο βρόχος προχωρά, κάθε στοιχείο του πίνακα θα "δέχεται επίσκεψη" μία φορά.

10. Και οι δύο βρόχοι FOR και REPeat:

    a. έχουν μία εντολή ανοίγματος:

        REPeat, FOR

    b. 'Εχουν μία εντολή κλεισίματος:

        END REPeat όνομα, END FOR όνομα

    c. έχουν όνομα βρόχου

    d. Μόνο ο βρόχος FOR έχει μεταβλητή βρόχου ή μεταβλητή ελέγχου.

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

Αυτό το τέστ είναι πιο ερευνητικό από τα προηγούμενα.

- 15 ή 16 είναι άριστα. Συνεχίστε να διαβάζετε.
- 13 ή 14 είναι πολύ καλά αλλά ξανασκεφτείτε λίγο μερικές ιδέες. Κοιτάξτε τα προγράμματα να δείτε πως δουλεύουν.
- 11 ή 12 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου έξη.
- 8 ως 10 είναι μέτρια αλλά αιαβάστε μερικά τμήματα του κεφαλαίου έξη και ξανακάντε το τέστ.
- κάτω από 8. Πρέπει να μελετήσετε προσεκτικά το κεφάλαιο έξη και να επαναλάβετε το τέστ.

**ΠΡΟΒΛΗΜΑΤΑ ΣΤΟ ΚΕΦΑΛΑΙΟ 6**

1. Χρησιμοποιήστε ένα βρόχο FOR για να τοποθετήσετε έναν από τους τέσσερις αριθμούς 1, 2, 3, 4 τυχαία σε πέντε μεταβλητές πίνακα:

    card(1), card(2), card(3), card(4), card(5)

    Δεν έχει σημασία αν μερικοί από τους τέσσερις αριθμούς επαναλαμβάνονται. Χρησιμοποιήστε ένα δεύτερο βρόχο FOR για να εξάγετε τις τιμές των πέντε μεταβλητών card.

2. Φανταστείτε ότι οι τέσσερις αριθμοί 1, 2, 3, 4 αντιπροσωπεύουν "Κούπες", "Μπαστούνια", "Καρρά", "Σπαθιά". Τι γραμμές θα χρειαζόσασταν για να έχετε έξοδο στη μορφή αυτών των λέξεων αντί για αριθμούς;

3. Χρησιμοποιήστε ένα βρόχο FOR για να τοποθετήσετε πέντε τυχαίους αριθμούς στην κλίμακα 1 ως 13 σ' έναν πίνακα πέντε μεταβλητών:

    card(1), card(2), card(3) και card(5)

Χρησιμοποιήστε ένα δεύτερο βρόχο FOR για να εξάγετε τις τιμές των πέντε μεταβλητών card.

4. Φανταστείτε ότι οι τυχαίοι αριθμοί που παράχθηκαν στο πρόβλημα 1 αντιπροσωπεύουν τραπουλόχαρτα. Γράψτε τις επιπλέον εντολές που θα επιφέρουν την ακόλουθη έξοδο.

| Αριθμός | Έξοδος |
|---|---|
| 1 | η λέξη "Ace" |
| 2 ως 10 | ο ίδιος ο αριθμός |
| 11 | η λέξη "Jack" |
| 12 | η λέξη "Queen" |
| 13 | η λέξη "King" |

# ΚΕΦΑΛΑΙΟ 7
# ΑΠΛΕΣ ΔΙΑΔΙΚΑΣΙΕΣ

**ΚΑΤΑΚΕΡΜΑΤΙΣΜΟΣ (Modularity)**

Αν επρόκειτο να γράψετε προγράμματα που να επιλύουν περίπλοκα προβλήματα, θα ήταν δύσκολο να ακολουθείτε την εξέλιξη των πραγμάτων. 'Ετσι ένας μεθοδικός προγραμματιστής διαιρεί μία μεγάλη ή πολύπλοκη εργασία σε μικρότερα τμήματα ή αποστολές και μετά ξαναδιαιρεί αυτές σε άλλες μικρότερες και ούτω καθ' εξής μέχρις ότου να μπορεί να τις χειριστεί εύκολα.

Αυτό είναι παρόμοιο με τη διευθέτηση των πολύπλοκων ανθρωπίνων υποθέσεων. Η επιτυχία μιας κυβέρνησης εξαρτάται από τον καταμερισμό των ευθυνών. Ο πρωθυπουργός διαιρεί τη δουλειά ανάμεσα στους υπουργούς οι οποίοι την καταμερίζουν ακόμη περισσότερο μέσω των Δημοσίων Υπηρεσιών μέχρις ότου οι αποστολές να μπορούν να πραγματοποιηθούν από άτομα χωρίς άλλη διαίρεση. Υπάρχουν πολύπλοκα χαρακτηριστικά όπως κοινές υπηρεσίες και συνεργασία μεταξύ των ίδιων και διαφορετικών επιπέδων αλλά η ιεραρχική δομή είναι η κυρίαρχη.

'Ενας καλός προγραμματιστής θα δουλεύει μ' αυτόν τον τρόπο και μία σύγχρονη γλώσσα όπως η SuperBASIC που επιτρέπει σωστά ονομασμένες και καλά ορισμένες διαδικασίες, θα του είναι πιο χρήσιμη από άλλες διαλέκτους που δεν έχουν τέτοια χαρακτηριστικά.

Η ιδέα είναι ότι ένα χωριστά ονομασμένο μπλοκ κώδικα θα έπρεπε να γραφεί για μία συγκεκριμένη αποστολή. Δεν έχει σημασία σε ποιο σημείο του προγράμματος βρίσκεται το μπλοκ του κώδικα. Αν βρίσκεται κάπου εκεί τότε η χρήση του ονόματος του:

- θα ενεργοποιήσει τον κώδικα
- θα επιστρέψει τον έλεγχο στο σημείο του προγράμματος αμέσως μετά τη χρήση.

Αν μία διαδικασία, η square, σχεδιάζει ένα τετράγωνο, το σχεδιάγραμμα είναι όπως φαίνεται παρακάτω:

```
ορισμός διαδικασίας                    κλήση διαδικασίας

DEFine PROCedure square
REMark κώδικας  για  τη          ←—  square
τετραγώνου
END DEFine
                  ↓
        σχεδιάζει ένα τετράγωνο
```

Στην πράξη οι χωριστές αποστολές σε μία δουλειά μπορούν να αναγνωριστούν και να ονομαστούν πριν ο κώδικας καθορισμού γραφεί. Το όνομα είναι το μόνο που χρειάζεται για την κλήση της διαδικασίας ώστε η βασική διάταξη του προγράμματος μπορεί να γραφεί πριν οριστούν όλες οι αποστολές.

Εναλλακτικά αν προτιμάτε, μπορείτε να γράψετε και να δοκιμάσετε πρώτα τις αποστολές. Αν λειτουργήσουν τότε μπορείτε να ξεχάσετε τις λεπτομέρειες και να θυμάστε μόνο το όνομα και το τι κάνει κάθε διαδικασία.

Το ακόλουθο παράδειγμα θα μπορούσε εύκολα να γραφεί χωρίς διαδικασίες αλλά δείχνει πως μπορούν να χρησιμοποιηθούν σε μία λογικά απλή έκφραση. Σχεδόν οποιαδήποτε αποστολή μπορεί να χωριστεί με όμοιο τρόπο, πράγμα που σημαίνει ότι ποτέ δε θα έχετε να ανυσηχείτε για περισσότερες από πέντε ως τριάντα ας πούμε, γραμμές κάθε φορά. Αν μπορείτε να γράψετε προγράμματα τριάντα γραμμών καλά και να χειρίζεστε τις διαδικασίες τότε έχετε την ικανότητα να γράφετε προγράμματα τριακοσίων γραμμών.

Μπορείτε να παράγετε έτοιμες φανταχτερές προτάσεις για πολιτικούς ή άλλους που θέλουν να κάνουν εντύπωση με την τεχνολογική τους ευφράδεια χωρίς στην πράξη να ξέρουν τίποτε. Αποθηκεύστε τις ακόλουθες λέξεις σε τρεις πίνακες και μετά φτιάξτε δέκα τυχαίες φανταχτερές φράσεις.

| adjec1$ | adjec2$ | nouns$ |
|---|---|---|
| Full | fifth-generation | systems |
| Systematic | knowledge-based | machines |
| Intelligent | compatible | computers |
| Controlled | cybernetic | feedback |
| Automated | user-friendly | transputers |
| Synchronised | parallel | micro-chips |
| Functional | learning | capability |
| Optional | adaptable | programming |
| Positive | modular | packages |
| Balanced | structured | databases |
| Integrated | logic-oriented | spreadsheets |
| Coordinated | file-oriented | word-processors |
| Sophisticated | standardised | objectives |

**ΑΝΑΛΥΣΗ**

Θα γράψουμε ένα πρόγραμμα που να παράγει δέκα φανταχτερές φράσεις. Τα στάδια του προγράμματος είναι:

[1] Αποθήκευση των λέξεων σε τρεις αλφαριθμητικούς πίνακες.

[2] Επιλογή τριών τυχαίων αριθμών που θα είναι οι δείκτες των μεταβλητών του πίνακα.

[3] Τύπωμα της φράσης.

[4] Επανάληψη των 2 και 3 δέκα φορές.

**ΣΧΕΔΙΑΣΜΟΣ**

**ΜΕΤΑΒΛΗΤΕΣ**

Αναγνωρίζουμε τρεις πίνακες από τους οποίους οι πρώτοι δύο θα περιέχουν επίθετα ή λέξεις που χρησιμοποιούνται σαν επίθετα - περιγραφικές λέξεις. Ο τρίτος πίνακας θα περιέχει τα ουσιαστικά. Υπάρχουν 13 λέξεις σε κάθε τμήμα και η μακρύτερη λέξη έχει 16 χαρακτήρες συμπεριλαμβανομένης της παύλας.

| **Πίνακας** | **Σκοπός** |
|---|---|
| **adjec1$(13,12)** | **πρώτα επίθετα** |
| **adjec2$(13,16)** | **δεύτερα επίθετα** |
| **main$(13,15)** | **ουσιαστικά** |

**ΔΙΑΔΙΚΑΣΙΕΣ**

Χρησιμοποιούμε τρεις διαδικασίες για να ταιριάσουν με τις δουλειές που αναγνωρίσαμε.

- η store_data αποθηκεύει τις τρεις ομάδες των δεκατριών λέξεων

- η get_random παράγει τρεις τυχαίους αριθμούς από 1 ως 13

- η make_phrase τυπώνει μία φράση

**ΚΥΡΙΟ ΠΡΟΓΡΑΜΜΑ**

Αυτό είναι πολύ απλό γιατί η βασική δουλειά έχει γίνει στις διαδικασίες.

[1] Δήλωση (DIM) των πινάκων

[2] Store_data

[3] FOR δέκα φράσεις
get_random
make_phrase
END

ΠΡΟΓΡΑΜΜΑ

```
100 REMark ************
110 REMark * Buzzword *
120 REMark ************
130 DIM adjec1$(13,12), adjec2$(13,16),noun$(13,15)
140 store_data
```

```
150 FOR phrase = 1 TO 10
160   get_random
170   make_phrase
180 END FOR phrase
190 REMark ****************************
200 REMark * Procedure Definitions   *
210 REMark ****************************
220 DEFine PROCedure store_data
230   REMark *** procedure to store the buzzword data ***
240   RESTORE 420
250   FOR item = 1 TO 13
260     READ adjec1$(item), adjec2$(item),noun$(item)
270   END FOR item
280 END DEFine
290 DEFine PROCedure get_random
300   REMark *** procedure to select the phrase ***
310   LET ad1 = RND(1 TO 13)
320   LET ad2 = RND(1 TO 13)
330   LET n = RND(1 TO 13)
340 END DEFine
350 DEFine PROCedure make_phrase
360   REMark *** procedure to print out the phrase ***
370   PRINT ! adjec1$(ad1) ! adjec2$(ad2) ! noun$(n)
380 END DEFine
390 REMark ******************
400 REMark * Program Data *
410 REMark ******************
420 DATA "Full", "fifth-generation", "systems"
430 DATA "Systematic", "knowledge-based", "machines"
440 DATA "Intelligent", "compatible", "computers"
450 DATA "Controlled", "cybernetic", "feedback"
460 DATA "Automated", "user-friendly", "transputers"
470 DATA "Synchronised", "parallel", "micro-chips"
480 DATA "Functional", "learning", "capability"
490 DATA "Optional", "adaptable", "programming"
500 DATA "Positive", "modular", "packages"
510 DATA "Balanced", "structured", "databases"
520 DATA "Integrated", "logic-oriented", "spreadsheets"
530 DATA "Coordinated", "file-oriented", "word-processors"
540 DATA "Sophisticated", "standardised", "objectives"
```

**ΔΕΙΓΜΑ ΕΞΟΔΟΥ**

```
Automated fifth-generation capability
Functional learning packages
Full parallel objectives
Positive user-friendly spreadsheets
```

Intelligent file-oriented capability
Synchronised cybernetic transputers
Functional logic-oriented micro-chips
Positive parallel feedback
Balanced learning databases
Controlled cybernetic objectives

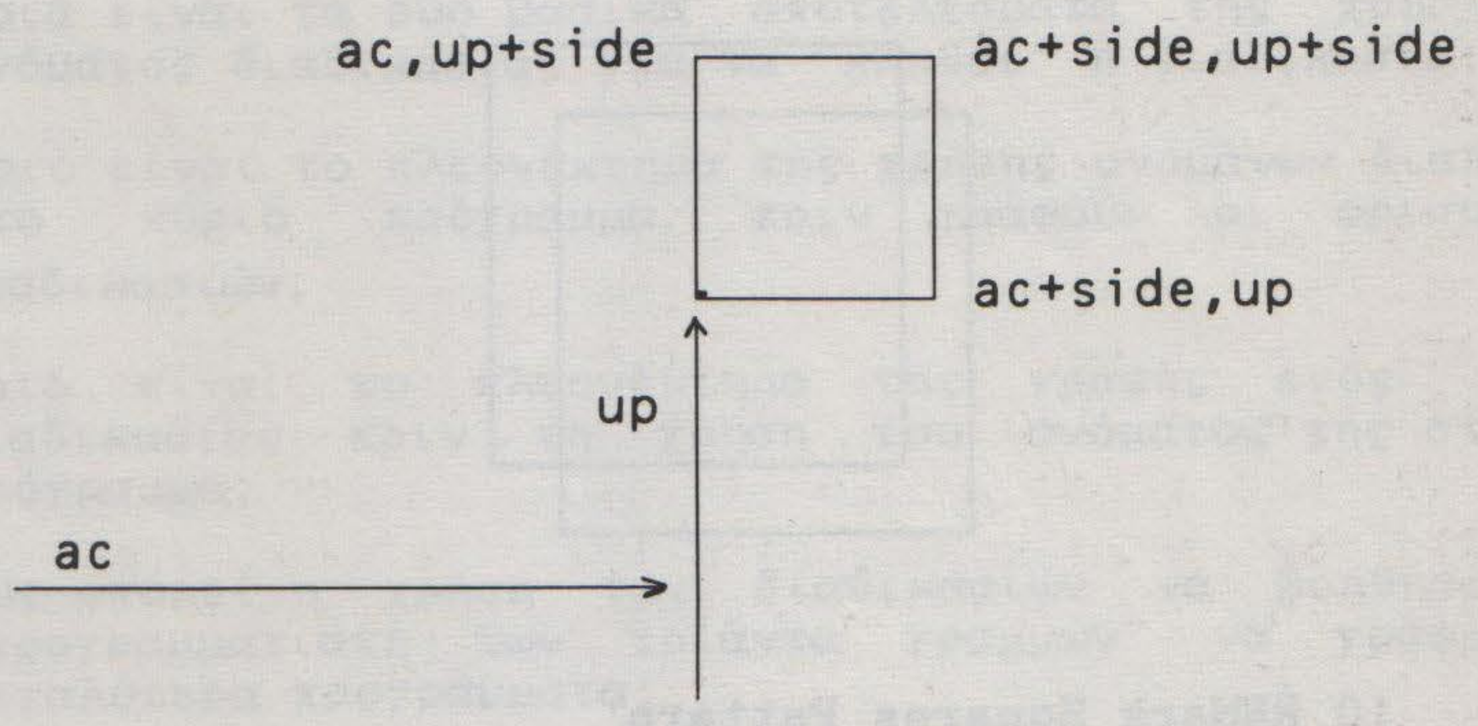
## ΠΕΡΝΩΝΤΑΣ ΠΛΗΡΟΦΟΡΙΕΣ ΣΤΙΣ ΔΙΑΔΙΚΑΣΙΕΣ

Υποθέτουμε ότι θέλουμε να σχεδιάσουμε τετράγωνα διαφόρων μεγεθών και χρωμάτων σε διάφορες θέσεις της οθόνης γραφικών με κλίμακα.

Αν ορίσουμε μία διαδικασία, τη square, για να το κάνει αυτό, θα χρειαστεί τέσσερις πληροφορίες:

- μήκος πλευράς
- χρώμα (κώδικας χρώματος)
- θέση (κατά μήκος και ύψος)

Η θέση του τετραγώνου καθορίζεται αν δώσουμε δύο τιμές, την κατά μήκος και την κατά ύψος, οι οποίες ορίζουν την αριστερή κάτω γωνία όπως φαίνεται:




Το χρώμα του τετραγώνου ορίζεται εύκολα αλλά το τετράγωνο χρησιμοποιεί τις τιμές του side και την ac και up όπως φαίνεται παρακάτω:

```
200 DEFine PROCedure square(side,ac,up)
210   LINE ac,up TO ac+side,up
220   LINE TO ac+side,up+side
230   LINE TO ac,up+side TO ac,up
240 END DEFine
```

Για να κάνουμε τη διαδικασία να δουλέψει πρέπει να δοθούν οι τιμές των side, ac και up. Αυτές οι τιμές δίνονται κατά την

κλήση της διαδικασίας. Για παράδειγμα, μπορείτε να προσθέσετε το ακόλουθο βασικό πρόγραμμα για να πάρετε ένα πράσινο τετράγωνο πλευράς 20.

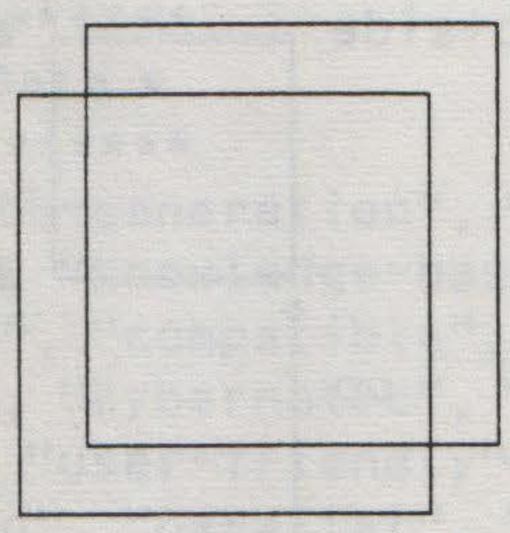
```
10 PAPER 7: CLS
20 INK 4
30 square 20,50,50
```

Οι αριθμοί 20, 50, 50 λέγονται παράμετροι και μεταφέρονται στις μεταβλητές που ονομάζονται στον ορισμό διαδικασίας έτσι ώστε:

```
square 20,50,50
       ↘  ↘  ↘
DEFINE PROCedure square(side,ac,up)
```

Οι αριθμοί 20, 50, 50 λέγονται πραγματικές παράμετροι. Είναι αριθμοί σ' αυτή την περίπτωση αλλά θα μπορούσαν να είναι μεταβλητές ή παραστάσεις. Οι μεταβλητές side, ac, up λέγονται τυπικές παράμετροι. Πρέπει να είναι μεταβλητές γιατί "δέχονται" τιμές.

'Ενα πιο ενδιαφέρον βασικό πρόγραμμα χρησιμοποιεί την ίδια διαδικασία για να δημιουργήσει ένα τυχαίο σχήμα από χρωματιστά ζεύγη τετραγώνων. Κάθε ζευγάρι τετραγώνων γίνεται με μεταφορά του δεύτερου τετραγώνου κατά το ένα πέμπτο της πλευράς, κατά μήκος και κατά ύψος ώστε:



```
10 REMark Squares Pattern'
20 PAPER 7 : CLS
30 FOR pair = 1 TO 20
40    INK RND(5)
50    LET side = RND(10 TO 20)
60    LET ac = RND(50) : up = RND(70)
70    square side,ac,up
80    LET ac=ac+side/5 : up = up+side/5
90    square side,ac,up
100 END FOR pair
```

Τα πλεονεκτήματα των διαδικασιών είναι:

[1] Μπορείτε να χρησιμοποιείτε τον ίδιο κώδικα περισσότερες από μία φορές, στο ίδιο πρόγραμμα ή σε διαφορετικά προγράμματα.

[2] Μπορείτε να χωρίσετε μία αποστολή σε υπο-αποστολές και να γράψετε μία διαδικασία για κάθε υπο-αποστολή. Αυτό βοηθάει την ανάλυση και το σχεδιασμό.

[3] Οι διαδικασίες μπορούν να δοκιμαστούν χωριστά. Αυτό βοηθάει τον έλεγχο και την εύρεση των λαθών.

[4] Τα ονόματα των διαδικασιών με νόημα και τα σαφώς ορισμένα αρχικά και τελικά σημεία βοηθούν στο να γίνεται ένα πρόγραμμα πιο ευανάγνωστο.

'Οταν συνηθίσετε τη χρήση σωστά ονομασμένων διαδικασιών με καλές ευκολίες παραμέτρων τότε θα δείτε ότι η ικανότητα σας να λύνετε προβλήματα και να προγραμματίζετε θα έχει αυξηθεί σημαντικά.

## ΑΥΤΟΕΛΕΓΧΟΣ ΣΤΟ ΚΕΦΑΛΑΙΟ 7

Στο ακόλουθο τέστ το άριστα είναι 14 βαθμοί. Δέστε το αποτέλεσμα που φέρατε από τις απαντήσεις στην επόμενη σελίδα.

1. Πως φυσιολογικά χειριζόμαστε το πρόβλημα του μεγέθους και της πολυπλοκότητας στις ανθρώπινες υποθέσεις;

2. Πως μπορεί αυτή η αρχή να εφαρμοστεί στον προγραμματισμό;

3. Ποιά είναι τα δύο πιο προφανή χαρακτηριστικά ενός απλού ορισμού διαδικασίας;

4. Ποιά είναι τα δύο βασικά αποτελέσματα της χρήσης ενός ονόματος διαδικασίας για να "κληθεί" η διαδικασία;

5. Ποιό είναι το πλεονέκτημα της χρήσης ονομάτων διαδικασιών στο κύριο πρόγραμμα πριν γραφούν οι ορισμοί των διαδικασιών;

6. Ποιό είναι το πλεονέκτημα της γραφής ενός ορισμού διαδικασίας πριν τη χρήση του ονόματος της στο κύριο πρόγραμμα;

7. Πως μπορεί η χρήση των διαδικασιών να βοηθήσει έναν "προγραμματιστή των τριάντα γραμμών" να γράψει πολύ μεγαλύτερα προγράμματα;

8. Μερικά προγράμματα χρησιμοποιούν περισσότερη μνήμη κατά τον ορισμό διαδικασιών, αλλά σε ποια περίπτωση οι διαδικασίες κάνουν οικονομία μνήμης;

9. Ονομάστε δύο τρόπους με τους οποίους μπορούν πληροφορίες να περάσουν από ένα κύριο πρόγραμμα σε μία διαδικασία.

10. Τι είναι οι πραγματικές παράμετροι;

11. Τι είναι οι τυπικές παράμετροι;

## ΑΠΑΝΤΗΣΕΙΣ ΣΤΟΝ ΑΥΤΟΕΛΕΓΧΟ ΤΟΥ ΚΕΦΑΛΑΙΟΥ 7

1. Κανονικά χωρίζουμε με τις μεγάλες ή πολύπλοκες δουλειές,

σε μικρότερες αποστολές μέχρι που να γίνουν τόσο μικρές ώστε να πραγματοποιούνται.

2. Αυτή η αρχή μπορεί να εφαρμοστεί στον προγραμματισμό χωρίζοντας τη συνολική δουλειά και γράφοντας μία διαδικασία για κάθε αποστολή.

3. Μία απλή διαδικασία είναι:

   - ένας χωριστός τομέας κώδικα

   - με κατάλληλο όνομα.

4. Μία κλήση διαδικασίας έχει αποτέλεσμα:

   - να ενεργοποιηθεί η διαδικασία

   - να επιστρέψει ο έλεγχος στο σημείο αμέσως μετά το σημείο κλήσης.

5. Τα ονόματα διαδικασιών μπορούν να χρησιμοποιηθούν στο κυρίως πρόγραμμα πριν γραφούν οι διαδικασίες. Αυτό σας επιτρέπει να σκέφτεστε συνολικά τη δουλειά και να έχετε μία γενική ιδέα χωρίς ν' ανησυχείτε για τις λεπτομέρειες.

6. Αν γράφετε τον ορισμό μιας διαδικασίας πριν χρησιμοποιήσετε το όνομά της τότε μπορείτε να τη δοκιμάσετε και όταν δουλέψει σωστά να ξεχάσετε τις λεπτομέρειες. Χρειάζεται μόνο να θυμάστε το όνομά της και τι κάνει σε γενικές γραμμές.

7. 'Ενας προγραμματιστής ικανός να γράφει προγράμματα μέχρι τριάντα γραμμών μπορεί να χωρίσει μία πολύπλοκη αποστολή σε διαδικασίες με τέτοιο τρόπο ώστε καμμιά να μην είναι πάνω από τριάντα γραμμές και πολλές θα είναι πολύ λιγότερο. Μ' αυτόν τον τρόπο θα τον απασχολεί ένα μέρος της δουλειάς κάθε φορά.

8. Η χρήση μιας διαδικασίας κάνει οικονομία μνήμης αν χρειάζεται αν χρειάζεται να κληθεί περισσότερες από μία φορές, από διαφορετικά μέρη του προγράμματος. Η διαδικασία ορίζεται μία μόνο φορά αλλά μπορεί να κληθεί όσο συχνά χρειάζεται.

9. 'Ενα κύριο πρόγραμμα μπορεί να τοποθετήσει πληροφορίες σε "φωλιές" με δηλώσεις LET και READ. Αυτές οι "φωλιές" μπορούν να προσπελαστούν από τη διαδικασία. 'Ετσι η διαδικασία χρησιμοποιεί πληροφορίες που αρχικά δημιουργήθηκαν από το κυρίως πρόγραμμα.

   Μία δεύτερη μέθοδος είναι η χρήση παραμέτρων στην κλήση της διαδικασίας. Αυτές οι τιμές μεταφέρονται σε μεταβλητές στον ορισμό της διαδικασίας που μετά τις χρησιμοποιεί όπως χρειάζεται.

10. Μία πραγματική παράμετρος είναι η πραγματική τιμή που μεταφέρεται από την κλήση διαδικασίας στο βασικό πρόγραμμα, προς τη διαδικασία.

11. Μία τυπική παράμετρος είναι η μεταβλητή σ' έναν ορισμό διαδικασίας, η οποία παίρνει την τιμή που μεταφέρεται στη διαδικασία από το κύριο πρόγραμμα.

**ΕΛΕΓΞΤΕ ΤΟΥΣ ΒΑΘΜΟΥΣ ΣΑΣ**

Αυτό το τέστ είναι ερευνητικό. 'Ισως χρειαστείτε περισσότερη πείρα πάνω στη χρήση διαδικασιών πριν κατανοήσετε εντελώς τις ιδέες. Αλλά είναι πολύ ισχυρές και όταν κατανοηθούν θα σας βοηθήσουν πολύ. Αυτό αξίζει όποιον κόπο κι αν καταβάλλετε.

- 12 ως 14 είναι άριστα. Συνεχίστε το διάβασμα με αυτοπεποίθηση.

- 10 ή 11 είναι πολύ καλά. Απλώς ξανακοιτάξτε ορισμένα σημεία.

- 8 ή 9 είναι καλά αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου επτά.
- 6 ή 7 είναι μέτρια αλλά ξαναδιαβάστε μερικά τμήματα του κεφαλαίου επτά. Μελετήστε προσεκτικά τα προγράμματα, σημειώνοντας όλες τις αλλαγές στις τιμές των μεταβλητών. Μετά ξανακάντε το τέστ.

- Κάτω από 6 διαβάστε το κεφάλαιο επτά ξανά. Πάρτε το σιγά-σιγά μελετώντας όλα τα προγράμματα. Αυτές οι ιδέες μπορεί να μην είναι εύκολες αλλά αξίζουν τον κόπο. 'Οταν είστε έτοιμοι, δοκιμάστε το τέστ ξανά.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 7**

1. Γράψτε μία διαδικασία που να έχει έξοδο μία από τις ακόλουθες λέξεις: "Hearts", "Clubs", "Diamonds", ή "Spades". Καλέστε τη διαδικασία πέντε φορές για να πάρετε τυχαία πέντε λέξεις.

2. Γράψτε ένα πρόγραμμα για το πρόβλημα 1 χρησιμοποιώντας έναν αριθμό στην κλίμακα 1 ως 4 σαν παράμετρο που θα ορίζει την λέξη της εξόδου. Αν το έχετε ήδη κάνει αυτό, τότε δοκιμάστε να γράψετε το πρόγραμμα χωρίς παραμέτρους.

3. Γράψτε μία διαδικασία που θα εξάγει την τιμή ενός τραπουλόχαρτου στην κλίμακα 2 ως 10 ή μία από τις λέξεις "Ace", "Jack", "Queen", "King".

4. Γράψτε ένα πρόγραμμα που καλεί αυτή τη διαδικασία πέντε φορές ώστε να εξαχθούν πέντε τυχαίες τιμές.

5. Γράψτε το πρόγραμμα του προβλήματος 3 ξανά, χρησιμοποιώντας έναν αριθμό από 1 ως 13 σαν παράμετρο που θα μεταφερθεί στη διαδικασία. Αν αυτή τη μέθοδο χρησιμοποιήσατε την πρώτη φορά, τότε προσπαθήστε να γράψετε το πρόγραμμα χωρίς παραμέτρους.

6. Γράψτε το πιο κομψό πρόγραμμα που μπορείτε, χρησιμοποιώντας διαδικασίες, για να εξάγετε τέσσερις σειρές πέντε τραπουλόχαρτων η κάθε μια. Μην ανησυχείτε για τα διπλά χαρτιά. Μπορείτε να ερμηνεύσετε την

κομψότητα σαν ένα κατάλληλο μίγμα ευαναγνωστικότητας, συντομίας και αποδοτικότητας. Διαφορετικοί άνθρωποι και ή διαφορετικές περιστάσεις θα δώσουν διαφορετική σημασία σε κάθε έναν από αυτούς τους τρεις παράγοντες που πολλές φορές ανταγωνίζονται ο ένας τον άλλο.

# ΚΕΦΑΛΑΙΟ 8
# ΑΠΟ ΤΗ BASIC ΣΤΗ SUPERBASIC

## ΕΙΣΑΓΩΓΗ

Αν είστε εξοικιωμένος με μία από τις προηγούμενες εκδόσεις της BASIC τότε ίσως προτιμήσετε να παραλείψετε τα πρώτα επτά κεφάλαια και να χρησιμοποιήσετε αυτό το κεφάλαιο στη θέση τους σαν μία γέφυρα μεταξύ αυτών που ήδη γνωρίζετε και των επόμενων κεφαλαίων. Αν το κάνετε αυτό και ακόμη βρίσκετε δυσκολίες τότε ίσως βοηθήσει αν πάτε λίγο πίσω σε μερικά από τα προηγούμενα κεφάλαια.

Αν έχετε μελετήσει τα προηγούμενα κεφάλαια τότε αυτό θα πρέπει να σας είναι εύκολο. 'Ισως ανακαλύψετε ότι παράλληλα με την παρουσίαση μερικών νέων ιδεών, δίνει και μία ενδιαφέρουσα κλίση με τον τρόπο που αναπτύσσεται η BASIC. Εκτός από τις ευκολίες δόμησης προγραμμάτων, η SuperBASIC προωθεί τα όρια της καλής παρουσίασης οθόνης, σύνταξης λειτουργικών ευκολιών και γραφικών. Με δύο λόγια είναι ένας συνδυασμός φιλικότητας και υπολογιστικής ισχύος που δεν έχει υπάρξει ποτέ πριν.

'Ετσι, όταν κάνετε τη μετάβαση από τη BASIC στη SuperBASIC, δεν κινείστε απλώς προς μία πιο ισχυρή και πιο βοηθητική γλώσσα, αλλά επίσης μεταβαίνετε σ' ένα εξαιρετικά προηγμένο υπολογιστικό περιβάλλον.

Τώρα θα συζητήσουμε τα βασικά χαρακτηριστικά της SuperBASIC και τα χρακτηριστικά που την διαχωρίζουν από τις άλλες BASIC.

## ΑΛΦΑΒΗΤΙΚΕΣ ΣΥΓΚΡΙΣΕΙΣ

Οι συνήθεις απλές αριθμητικές συγκρίσεις είναι δυνατές. Μπορείτε να γράψετε:

```
LET pet1$ = "CAT"
LET pet2$ = "DOG"
IF pet1$ < pet2$ THEN PRINT "Meow"
```

Η έξοδος θα είναι Meow γιατί σ' αυτή την περίσταση, το σύμβολο < σημαίνει προηγούμενο (πιο κοντά στο A, στο αλφάβητο).

Η SuperBASIC κάνει τις συγκρίσεις λογικά. Για παράδειγμα θα πρέπει να περιμένετε:

το "CAT" να είναι προηγούμενο του "DOG"

και

το "ERD98L" να είναι προηγούμενο του "ERD746L"

Μία απλοική προσέγγιση, με τυφλή χρήση της εσωτερικής κωδικοποίησης των χαρακτήρων, θα έδινε "λάθος" αποτέλεσμα και στις δύο παραπάνω περιπτώσεις αλλά δοκιμάστε το ακόλουθο πρόγραμμα που βρίσκει την "προηγούμενη" από δύο σειρές χαρακτήρων:

```
100 INPUT item1$, item2$
110 IF item1$ < item2$ THEN PRINT item1$
120 IF item2$ = item2$ THEN PRINT "Equal"
130 IF item1$ > item2$ THEN PRINT item2$
```

| ΕΙΣΟΔΟΣ | | ΕΞΟΔΟΣ |
|---|---|---|
| cat | dog | cat |
| cat DOG | cat | |
| ERD98L | ERD746L | ERD98L |
| ABC | abc | ABC |

Το τμήμα 'Εννοιες θα σας δώσει όλες τις λεπτομέρειες για τον τρόπο με τον οποίο γίνεται η σύγκριση αλφαριθμητικών μεταβλητών στη SuperBASIC.

## ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΟΝΟΜΑΤΑ - ΑΝΑΓΝΩΡΙΣΤΕΣ

Οι περισσότερες BASIC έχουν αριθμητικές και αλφαριθμητικές μεταβλητές. 'Οπως σε άλλες BASIC, η χαρακτηριστική διαφορά ενός ονόματος αλφαριθμητικής μεταβλητής στη SuperBASIC είναι το σύμβολο του δολαρίου στο τέλος. 'Ετσι:

| | | | |
|---|---|---|---|
| αριθμητικές: | count | αλφαριθμητικές: | word$ |
| | sum | | high_st$ |
| | total | | day_of_week$ |

'Ισως να μην είχατε συναντήσει ονόματα μεταβλητών με νόημα αν και μερικές πρόσφατες BASIC τα επιτρέπουν. Οι κανόνες των αναγνωριστών στη SuperBASIC δίνονται στις 'Εννοιες. Το μέγιστο μήκος ενός αναγνωριστή είναι 255 χαρακτήρες. Η επιλογή των αναγνωριστών είναι προσωπική. Μερικές φορές οι πιο μεγάλοι βοηθούν περισσότερο στο να καταλάβει ο αναγνώστης τι θα πρέπει να κάνει ένα πρόγραμμα. Αλλά πρέπει να πληκτρολογηθούν και, όπως και στη συνηθισμένη γλώσσα, το όνομα spade είναι πιο λογικό από το horticultural earth-turning implement. Οι μικρότερες λέξεις προτιμούνται αν μετατρέπουν το νόημα, αλλά πολύ μικρές λέξεις ή απλά γράμματα πρέπει να χρησιμοποιούνται πολύ φειδωλά. Ονόματα μεταβλητών όπως X, Z, P3, Q2 εισάγουν ένα επίπεδο αφαίρεσης που θα δυσκολέψει πολλούς.

## ΑΚΕΡΑΙΕΣ ΜΕΤΑΒΛΗΤΕΣ

Η SuperBASIC επιτρέπει ακέραιες μεταβλητές που δέχονται μόνο ακέραιες τιμές. Τις διακρίνουμε με ένα σύμβολο επί τοις εκατό στο τέλος του ονόματος:

```
count%
number%
nearest_pound%
```

Υπάρχουν τώρα δύο είδη αριθμητικών μεταβλητών. Τον άλλο τύπο, που δέχεται ακέραιες ή δεκαδικές τιμές το λέμε κινητής υποδιαστολής. 'Ετσι μπορείτε να γράψετε:

```
LET price = 9
LET cost = 7.31
LET count% = 13
```

Αλλά αν γράψετε:

```
LET count% = 5.43
```

τότε η τιμή του count% θα γίνει 5. Από την άλλη μεριά:

```
LET count% = 5.73
```

θα έχει αποτέλεσμα η τιμή του count% να γίνει 6. 'Οπως βλέπετε η SuperBASIC κάνει ότι μπορεί στρογγυλεύοντας τους αριθμούς στον πλησιέστερο ακέραιο.

**ΕΞΑΝΑΓΚΑΣΜΟΣ**

Η αρχή του να προσπαθείς πάντα να είσαι έξυπνα βοηθητικός, παρά να δίνεις ένα μήνυμα λάθους ή να κάνεις κάτι προφανές ανεπιθύμητο, συνεχίζεται παραπέρα. Για παράδειγμα, αν μία αλφαριθμητική μεταβλητή, η mark$ έχει την τιμή:

```
"64"
```

τότε:

```
LET score = mark$
```

θα παράγει μία αριθμητική τιμή του 64 για το score. 'Αλλες εκδόσεις της BASIC μάλλον θα σταματούσαν και θα έλεγαν κάτι όπως:

```
"Type mis-match"
or "Nonsense in BASICC"
```

Αν η αλφαριθμητική μεταβλητή δεν μπορεί να μετατραπεί τότε αναφέρεται λάθος.

**ΛΟΓΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΑΠΛΕΣ ΔΙΑΔΙΚΑΣΙΕΣ**

Αυτός είναι ένας άλλος τύπος μεταβλητής στη SuperBASIC, ή μάλλον το σύστημα της SuperBASIC το κάνει να φαίνεται έτσι. Δέστε τη δήλωση σε SuperBASIC.

```
IF windy THEN fly_kite
```

Σε άλλες BASIC ίσως να γράφατε:

```
IF w=1 THEN GOSUB 300
```

Σ' αυτήν την περίπτωση το w=1 είναι ένας όρος ή λογική έκφραση που μπορεί να είναι αληθής ή ψευδής. Αν είναι αληθής τότε μία υπορουτίνα που αρχίζει στη γραμμή 300 θα εκτελεστεί. Η

υποροutίνα μπορεί να σχετίζεται με το πέταγμα χαρταετών αλλά δεν μπορείτε να το καταλάβετε από την παραπάνω γραμμή. 'Ενας προσεκτικός προγραμματιστής θα έγραφε:

```
IF w=1 THEN GOSUB 300 : REM fly_kite
```

για να το κάνει πιο ευανάγνωστο. Αλλά η δήλωση της SuperBASIC είναι ευανάγνωστη από μόνη της. Ο αναγνωριστής windy ερμηνεύεται ως αληθής ή ψευδής παρόλο που στην πραγματικότητα είναι μεταβλητή κινητής υποδιαστολής. Μια τιμή 1 ή οποιαδήποτε μη-μηδενική τιμή λαμβάνεται ως αληθής. Το μηδέν λαμβάνεται ως ψευδές. 'Ετσι η απλή λέξη windy έχει το ίδιο αποτέλεσμα όπως ένας όρος ή μία λογική έκφραση.

Η άλλη λέξη, το fly_kite είναι μία διαδικασία. Κάνει μία δουλειά παρόμοια αλλά μάλλον καλύτερη από το GOSUB 300.

Το ακόλουθο πρόγραμμα θα μεταδώσει την ιδέα της λογικής μεταβλητής και του απλούστερου τύπου ονομασμένης διαδικασίας:

```
100 INPUT windy
100 IF windy THEN fly_kite
120 IF NOT windy THEN tidy_shed
130 DEFine PROCedure fly_kite
140   PRINT "See it in the air."
150 END DEFine
160 DEFine PROCedure tidy_shed
170   PRINT "Sort out rubbish."
180 END DEFine
```

| ΕΙΣΟΔΟΣ | ΕΞΟΔΟΣ |
|---|---|
| 0 | Sort out rubbish. |
| 1 | See it in the air |
| 2 | See it in the air |
| -2 | See it in the air |

Μπορείτε να δείτε ότι μόνο το μηδέν λαμβάνεται ότι σημαίνει ψευδές. Φυσιολογικά δε θα γράφατε διαδικασίες με μόνο μία δήλωση, αλλά το πρόγραμμα επιδεικνύει την ιδέα και τη σύνταξη σε μία πολύ απλή ανάλογη έκφραση. θα ειπωθούν περισσότερα σχετικά με τις διαδικασίες αργότερα σ' αυτό το κεφάλαιο.

**ΕΝΤΟΛΕΣ LET**

Στη SuperBASIC το LET είναι προαιρετικό αλλά το χρησιμοποιούμε σ' αυτό το εγχειρίδιο για να υπάρχει μικρή πιθανότητα σύγχισης μεταξύ των δύο πιθανών χρήσεων του =. Το νόημα του = στο:

```
LET count = 3
```

και στο

```
IF count = 3 THEN EXIT
```

είναι διαφορετικό και το LET δίνει έμφαση σ' αυτό. 'Ομως αν υπάρχουν πάνω από δύο εντολές LET που κάνουν κάποια απλή

δουλειά όπως τοποθέτηση αρχικών τιμών, μπορεί να γίνει εξαίρεση.

Για παράδειγμα:

```
110 LET first = 0
120 LET second = 0
130 LET third = 0
```

μπορούν να ξαναγραφούν:

```
110 LET first = 0 : second = 0 : third = 0
```

χωρίς μείωση της καθαρότητας ή του στυλ. Αυτό είναι σύμφωνο με το γενικό σχέδιο των επιτρεπτών μικρών μορφών και άλλων δομών, όπου χρησιμοποιούνται με απλούς τρόπους.

Η άνω κάτω τελεία : είναι μία επιτρεπτή λήξη εντολής και μπορεί να χρησιμοποιείται και με άλλες εντολές εκτός από το LET.

## Η ΟΘΟΝΗ ΤΗΣ BASIC

### ΜΟΡΦΕΣ ΚΑΙ PIXELS

Σε επόμενο κεφάλαιο θα εξηγήσουμε με ποιο τρόπο μπορούμε να χειριστούμε άλλες ευκολίες γραφικών, όπως σχεδίαση κύκλων με ευκολία, αλλά εδώ υπογραμμίζουμε τα χαρακτηριστικά με προσανατολισμό στα pixels. Υπάρχουν δύο μορφές που μπορούν να ενεργοποιηθούν με οποιοδήποτε από τα ακόλουθα:

| | |
|---|---|
| **Χαμηλή διακριτικότητα** | |
| **Μορφή 8 Χρωμάτων** | **MODE 256** |
| **256 pixels κατά μήκος, 256 κάτω** | **MODE 8** |
| **Ψηλή διακριτικότητα** | |
| **Μορφή 4 χρωμάτων** | **MODE 512** |
| **512 pixels κατά μήκος, 256 κάτω** | **MODE 4** |

Και στις δύο μορφές τα pixels έχουν διευθύνσεις στην κλίμακα των αριθμών

```
0 - 511 κατά μήκος
και 0 - 255 κατά ύψος
```

Αφού η MODE 8 έχει μόνο τα μισά pixels κατά μήκος της οθόνης απ' ότι η MODE 4, τα pixels της MODE 8 είναι διπλά σε πλάτος από αυτά της MODE 4, κι έτσι τα pixels της MODE 8 μπορούν να οριστούν από δύο συντεταγμένες. Για παράδειγμα:

```
0 ή 1   2 ή 3   510 ή 511
```

Επίσης σημαίνει ότι μπορείτε να χρησιμοποιείτε την ίδια κλίμακα αριθμών για τις διευθύνσεις των pixels, ανεξάρτητα της μορφής. Πάντα να θυμάστε 0-511 κατά μήκος και 0-255 κατά ύψος.

Αν χρησιμοποιείτε τηλεόραση τότε μπορεί να μην είναι ορατά όλα τα pixels.

**ΧΡΩΜΑΤΑ**

Τα διαθέσιμα χρώματα είναι:

| MODE 256 | Code | MODE 512 |
|---|---|---|
| μαύρο | 0 | μαύρο |
| μπλε | 1 | |
| κόκκινο | 2 | κόκκινο |
| μωβ | 3 | |
| πράσινο | 4 | πράσινο |
| γαλάζιο | 5 | |
| κίτρινο | 6 | άσπρο |
| άσπρο | 7 | |

**Κώδικες Χρωμάτων**

'Ισως βρείτε την ακόλουθη μνημονική πρόταση χρήσιμη στο να θυμάστε τους κώδικες.

Bonny Babies Really Make Good Children, You Wonder

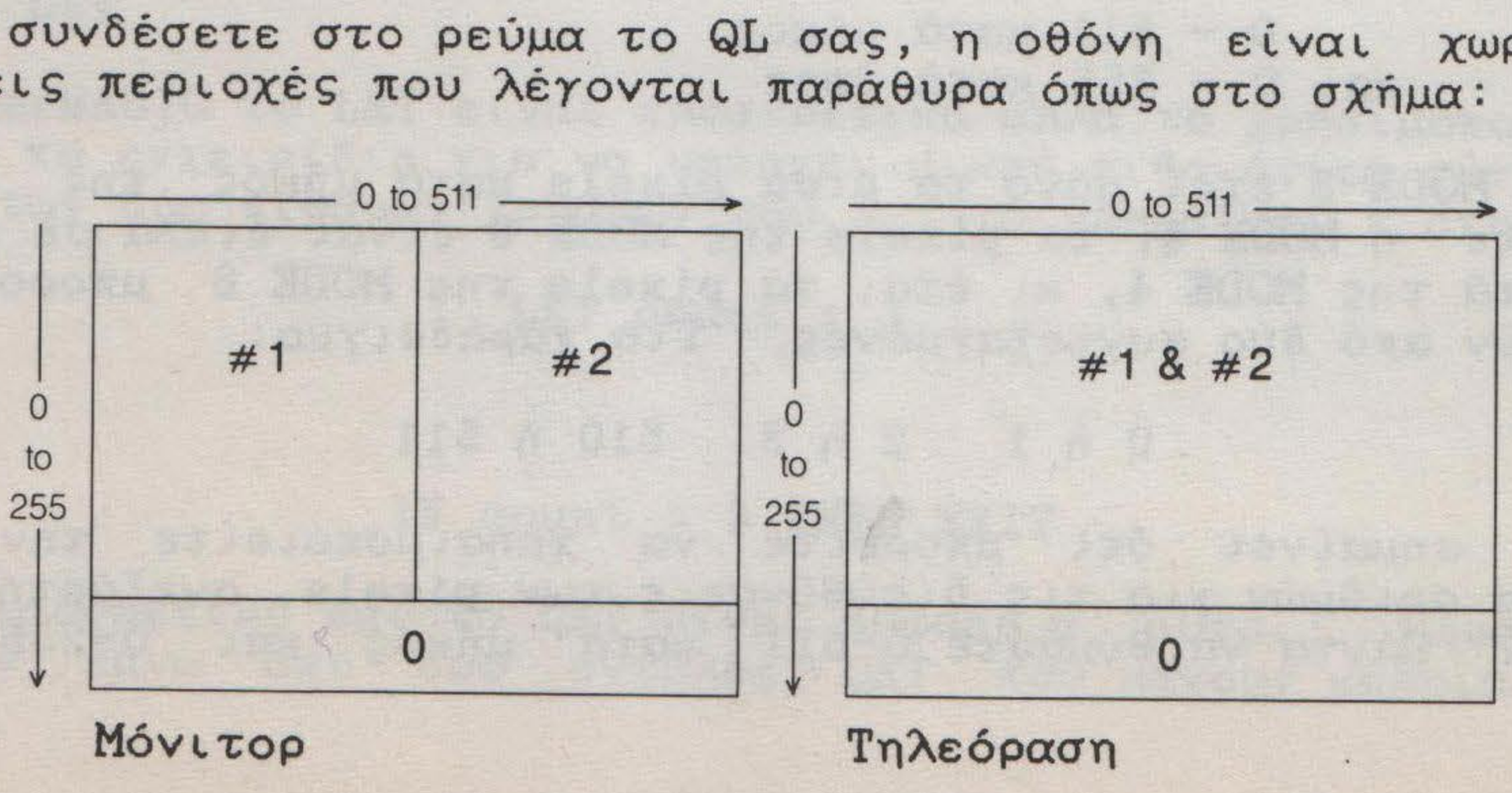
Στη μορφή υψηλής διακριτικότητας κάθε χρώμα μπορεί να επιλέγει από δύο κώδικες. θα δείτε αργότερα πως μία καταπληκτική κλίμακα χρωμάτων και αποτελεσμάτων από pixels χρωμάτων (πλοκή χρωμάτων) μπορεί να παραχθεί αν έχετε ένα μόνιτορ καλής ποιότητας.

Μερικές από τις δεσμευμένες λέξεις της παρουσίασης της οθόνης είναι οι ακόλουθες:

| | | |
|---|---|---|
| **INK** | colour | χρώμα μελάνης. |
| **BORDER** | width, colour | σχεδίαση περιθωρίου στην άκρη της οθόνης ή του παραθύρου. |
| **PAPER** | colour | χρώμα φόντου. |
| **BLOCK** | width, height, accross, down, colour | χρωματισμός ενός παραλληλογράμμου που έχει την πάνω αριστερή γωνιά στη θέση accross, down |

**ΟΡΓΑΝΩΣΗ ΟΘΟΝΗΣ ΤΟΥ QL**

'Οταν συνδέσετε στο ρεύμα το QL σας, η οθόνη είναι χωρισμένη σε τρεις περιοχές που λέγονται παράθυρα όπως στο σχήμα:



Τα παράθυρα αναγνωρίζονται με #0, #1 και #2, ώστε να συσχετίσετε διάφορα αποτελέσματα με συγκεκριμένα παράθυρα. Για παράδειγμα:

```
CLS
```

θα καθορίσει το παράθυρο #1 (διαλέγει το σύστημα) ώστε αν θέλετε να καθαρίσετε την αριστερή περιοχή, πρέπει να πληκτρολογήσετε:

```
CLS #2
```

Αν θέλετε ένα διαφορετικό PAPER (χρώμα φόντου) πληκτρολογήστε για πράσινο:

```
PAPER 4 : CLS
```

ή

```
PAPER #2,4 : CLS #2
```

ή

αν θέλετε να καθαρίσετε το παράθυρο #2 σε πράσινο χρώμα φόντου. Οι αριθμοί #0, #1, #2 λέγονται αριθμοί καναλιών. Σ' αυτή τη συγκεκριμένη περίπτωση σας επιτρέπουν να κατευθύνετε ορισμένα αποτελέσματα στο παράθυρο της επιλογής σας. θα ανακαλύψετε επίσης ότι οι αριθμοί καναλιών έχουν πολλές άλλες χρήσεις αλλά για την ώρα σημειώστε ότι όλες οι ακόλουθες εντολές μπορούν να έχουν αριθμό καναλιού. Η τρίτη στήλη δείχνει το κανάλι παράλειψης αυτό που διαλέγει το σύστημα αν δεν το καθορίσετε εσείς.

Σημειώστε ότι τα παράθυρα μπορεί να αλληλεπικαλύπτονται. Αν χρησιμοποιείτε TV τότε το σύστημα αυτόματα αλληλεπικαλύπτει τα παράθυρα #1 και #2 ώστε να υπάρχουν περισσότερες θέσεις χαρακτήρων διαθέσιμοι σε κάθε γραμμή για τις λίστερς των προγραμμάτων.

| Λέξη-κλειδί | Αποτέλεσμα | Default |
|---|---|---|
| **AT** | **θέση χαρακτήρος** | **#1** |
| **BLOCK** | **σχεδίαση τετραπλεύρου** | **#1** |
| **BORDER** | **σχεδίαση περιθωρίου** | **#1** |
| **CLS** | **καθάρισμα οθόνης** | **#1** |
| **CSIZE** | **μέγεθος χαρακτήρων** | **#1** |
| **CURSOR** | **θέση του δείκτη** | **#1** |
| **FLASH** | **προκαλεί-αναιρεί το αναβόσβημα καναλιών** | **#1** |
| **INK** | **χρώμα μελάνης** | **#1** |
| **OVER** | **αποτέλεσμα τυπώματος και γραφικών** | **#1** |
| **PAN** | **μετακινεί την οθόνη πλάγια** | **#1** |
| **PAPER** | **χρώμα φόντου** | **#1** |
| **RECOL** | **αλλαγή χρώματος** | **#1** |
| **SCROLL** | **μετακινεί την οθόνη κάθετα** | **#1** |
| **STRIP** | **φόντος για τύπωμα** | **#1** |

| | | |
|---|---|---|
| **UNDER** | υπογράμμιση | #1 |
| **WINDOW** | αλλάζει το υπάρχουν παράθυρο | #1 |
| **LIST** | κάνει λίστα του προγράμματος | #2 |
| **DIR** | κάνει λίστα του πίνακα περιεχομένων | #1 |
| **PRINT** | τύπωμα χαρακτήρων | #1 |
| **INPUT** | λήψη εισόδου από το πληκτρολόγιο | #1 |

Οι εντολές οι άμεσες διαταγές εμφανίζονται στο παράθυρο #0. Για περισσότερες λεπτομέρειες σχετικά με την σύνταξη και τη χρήση αυτών των δεσμευμένων λέξεων, δέστε άλλα τμήματα του εγχειριδίου.

## ΠΑΡΑΛΛΗΛΟΓΡΑΜΜΑ ΚΑΙ ΓΡΑΜΜΕΣ

Το ακόλουθο πρόγραμμα σχεδιάζει ένα πράσινο παραλληλόγραμμο στη μορφή 256, σε κόκκινο φόντο με κίτρινο περιθώριο πλάτους ενός pixel. Το παραλληλόγραμμο έχει την πάνω αριστερή γωνία του στις συντεταγμένες pixel 100,100 (δές τις 'Εννοιες του QL). Το πλάτος του είναι 80 μονάδες κατά μήκος (40 pixels) και το ύψος του είναι 20 μονάδες ύψος (20 pixels).

```
100 REMark Rectangle
110 MODE 256
120 BORDER 1,6
130 PAPER 1 : CLS
140 BLOCK 80, 20, 100, 100,4
```

Πρέπει να είστε λίγο προσεκτικοί στο MODE 256 επειδή οι τιμές κατά μήκος είναι από 0 ως 511 παρόλο που υπάρχουν μόνο 256 pixels. Δεν μπορούμε να πούμε ότι το παραλληλόγραμμο που φτιάχτηκε από το παραπάνω πρόγραμμα είναι 80 στίγματα πλατύ κι έτσι λέμε 80 pixels.

## ΕΙΣΟΔΟΣ ΚΑΙ ΕΞΟΔΟΣ

Η SuperBASIC έχει τις συνηθισμένες δηλώσεις LET, INPUT, READ και DATA για είσοδο. Η δήλωση PRINT χειρίζεται τις περισσότερες εξόδους κειμένου με το συνήθη τρόπο με τους διαχωριστές:

, κατατάσσει σε πίνακα την έξοδο

; απλώς χωρίζει - δεν έχει μορφοποιητικό αποτέλεσμα

\ δίνει καινούργια γραμμή

! έξυπνο διάστημα. Φυσιολογικά δίνει ένα διάστημα αλλά όχι στην αρχή της γραμμής. Αν ένα θέμα δεν χωράει στο τέλος της γραμμής τότε δίνει μία νέα γραμμή.

## ΒΡΟΧΟΙ

θα είστε εξοικιωμένοι με δύο τύπους επαναληπτικών βρόχων όπως στα ακόλουθα παραδείγματα:

α. Προσομοιώνει 6 ρίψεις ενός κοινού ζαριού με έξι πλευρές.

```
100 FOR throw = 1 TO 6
110 PRINT RND(1 TO 6)
120 NEXT throw
```

β. Προσομοιώνει ρίψεις ενός ζαριού μέχρι να εμφανιστεί ένα έξι.

```
100 die = RND(1 TO 6)
110 PRINT die
120 IF die <> 6 THEN GOTO 100
```

Και τα δύο αυτά προγράμματα θα δουλέψουν στη SuperBASIC αλλά συνιστούμε αντί γι' αυτά, τα ακόλουθα, που κάνουν ακριβώς τις ίδιες δουλειές. Παρόλο που το πρόγραμμα b) είναι λίγο πιο πολύπλοκο, υπάρχουν καλοί λόγοι για να το προτιμήσετε.

α.
```
100 FOR throw = 1 TO 6
110   PRINT RND(1 TO 6)
120 END FOR throw
```

β.
```
100 REPeat throws
110   die = RND(1 TO 6)
120   PRINT die
130   IF die = 6 THEN EXIT throws
140 END REPeat throws
```

Είναι λογικό να υπάρχουν δύο δομές για βρόχους, που τελειώνουν με έναν όρο (βρόχοι REPeat) ή που ελέγχονται από μετρητή.

Η θεμελιώδης δομή REPeat είναι:

```
REPeat αναγνωριστής

    εντολές

END REPeat αναγνωριστής
```

Η εντολή EXIT μπορεί να τοποθετηθεί οπουδήποτε μέσα στη δομή αλλά πρέπει να ακολουθείται από έναν αναγνωριστή που θα λέει στη SuperBASIC από ποιο βρόχο να εξέλθει. Για παράδειγμα:

```
EXIT throws
```

θα μεταφέρει τον έλεγχο στη εντολή μετά από την

```
END REPeat throws
```

Αυτό μπορεί να φαίνεται σαν να χρησιμοποιούμε μια βαρειά για να σπάσουμε το καρύδι του απλού προβλήματος που επιδείχθηκε. 'Ομως η δομή REPeat είναι πολύ ισχυρή και θα σας πάει μακρυά. Αν ξέρετε άλλες γλώσσες θα δείτε ότι κάνει τις δουλειές των δομών REPeat και WHILE και ακόμη τα βγάζει πέρα και με άλλες πιο άτεχνες καταστάσεις.

Ο βρόχος REPeat της SuperBASIC είναι ονομαζόμενος ώστε να γίνεται μία σωστή καθαρή έξοδος. Ο βρόχος FOR όπως όλες οι δομές της SuperBASIC τελειώνει με END και του δίνεται όνομα για λόγους που θα δειχτούν καλύτερα αργότερα.

θα δείτε επίσης αργότερα πως αυτές οι δομές βρόχων μπορούν να χρησιμοποιηθούν σε απλές ή πολύπλοκες καταστάσεις ταιριάζοντας απόλυτα μ' αυτό που θέλετε να κάνετε. θα αναφέρουμε μόνο τρία ακόμη χαρακτηριστικά των βρόχων σ' αυτό το στάδιο. θα σας είναι οικεία αν είστε έμπειρος χρήστης της BASIC.

Η αύξηση της μεταβλητής ελέγχου ενός βρόχου FOR είναι φυσιολογικά 1 αλλά μπορείτε να την αλλάξετε χρησιμοποιώντας τη δεσμευμένη λέξη STEP όπως δείχνει το παράδειγμα:

i.
```
100 FOR even = 2 TO 10 STEP 2
110   PRINT !  even !
120 END FOR even
```

η έξοδος είναι: 2 4 6 8 10

ii.
```
100 FOR backwards = 9 TO 1 STEP - 1
110   PRINT ! backwards !
120 END FOR backwards
```

η έξοδος είναι: 9 8 7 6 5 4 3 2 1

Το δεύτερο χαρακτηριστικό είναι ότι οι βρόχοι μπορούν να είναι φωλιασμένοι. Μπορεί να είστε εξοικιωμένοι με φωλιασμένους βρόχους FOR. Για παράδειγμα το ακόλουθο πρόγραμμα δίνει τέσσερις σειρές από δέκα χ.

```
100 REMark Crosses
110 FOR row = 1 TO 4
120   PRINT ' Row number'!row
130   FOR cross = 0 TO 10
140     PRINT!X!
150   END FOR cross
160   PRINT
170 PRINT ' END of row number'!row
180 END FOR row
```

η έξοδος είναι:

```
Row number 1
X X X X X X X X X X
End of row number 1
Row number 2
X X X X X X X X X X
End of row number 2
Row number 3
X X X X X X X X X X
End of row number 3
Row number 4
X X X X X X X X X X
End of row number 4
```

'Ενα μεγάλο πλεονέκτημα της SuperBASIC είναι ότι έχει δομές για όλες τις περιπτώσεις, όχι μόνο βρόχους FOR, και μπορούν όλες να φωλιάσουν η μία μέσα στην άλλη ανάλογα με τις απαιτήσεις της αποστολής. Μπορούμε να βάλουμε ένα βρόχο REPeat μέσα σ' ένα βρόχο FOR. Το παρακάτω πρόγραμμα παράγει τα αποτελέσματα δύο ζαριών σε κάθε σειρά μέχρι να τύχει ένα επτά, αντί για χ.

```
100 REMark Dice rows
110 FOR row = 1 TO 4
120   PRINT 'Row number '! row
130   REPeat throws
140     LET die1 = RND(1 TO 6)
150     LET die2 = RND(1 TO 6)
160     LET score = die1 + die2
170     PRINT score!
180     IF score = 7 THEN EXIT throws
190   END REPeat throws
200   PRINT'End of row' ! row
210 END FOR row
```

Αποτέλεσμα:

```
Row number 1
8 11 6 3 7
End of row number 1
Row number 2
4 6 2 9 4 55 12 7
End of row number 2
Row number 3
7
End of row number 3
Row number 4
6 2 4 9 9 7
End of row number 4
```

Το τρίτο χαρακτηριστικό των βρόχων στη SuperBASIC επιτρέπει περισσότερη ευελιξία στην επιλογή της κλίμακας των τιμών ενός βρόχου FOR. Το ακόλουθο πρόγραμμα το επιδεικνύει τυπώνοντας όλους τους μη πρώτους αριθμούς από 1 ως 20. 'Ενας μη πρώτος αριθμός είναι αυτός που διαιρείται και από άλλον αριθμό εκτός από τον εαυτό του και τη μονάδα.

```
100 REMark Divisible numbers
110 FOR num = 4,6,8 TO 10,12,14 TO 16,18,20
120   PRINT ! num!
130 END FOR num
```

θα ειπωθούν περισσότερα για το χειρισμό των επαναλήψεων σε επόμενο κεφάλαιο αλλά τα χαρακτηριστικά που περιγράφηκαν παραπάνω, θα αντιπαρέλθουν όλες τις καταστάσεις εκτός από μερικές ασυνήθιστες ή προχωρημένες.

## ΛΗΨΗ ΑΠΟΦΑΣΕΩΝ

θα έχετε παρατηρήσει τον απλό τύπο απόφασης:

```
IF die = 6 THEN EXIT throws.
```

Αυτός είναι διαθέσιμος στις περισσότερες BASIC αλλά η SuperBASIC προσφέρει επεκτάσεις αυτής της δομής καθώς και μία εντελώς νέα δομή για το χειρισμό καταστάσεων με περισσότερες από δύο εναλλακτικές περιπτώσεις ενεργειών.

Παρόλα αυτά, ίσως βρείτε τις ακόλουθες μεγάλες μορφές των IF...THEN χρήσιμες. Εξηγούνται από μόνες τους:

i.
```
100 REMark Long form IF ...END IF
110 LET sunny = RND(0 TO 1)
120 IF sunny THEN
130   PRINT 'Wear sunglasses'
140   PRINT 'Go for walk'
150 END IF
```

ii.
```
100 REMark Long form IF...END IF
110 LET sunny = RND(0 TO 1)
120 IF sunny THEN
130   PRINT 'Wear sunglasses'
140   PRINT 'Go for walk'
150 ELSE
160   PRINT 'Wear coat'
170   PRINT 'Go to cinema'
180 END IF
```

Ο διαχωριστής THEN είναι προαιρετικός στις μεγάλες μορφές ή μπορεί να αντικατασταθεί από άνω - κάτω τελεία στις μικρές μορφές. Οι μεγάλες μορφές των δομών απόφασης, έχουν την ίδια ιδιότητα όπως οι βρόχοι. Μπορείτε να τις φωλιάσετε ή να βάλετε άλλες δομές μέσα τους. 'Οταν μία απλή μεταβλητή εμφανίζεται εκεί που θα περιμένατε έναν όρο, τότε η τιμή μηδέν θα παίρνεται σαν ψευδής και οι άλλες τιμές ως αληθείς.

## ΥΠΟΡΟΥΤΙΝΕΣ ΚΑΙ ΔΙΑΔΙΚΑΣΙΕΣ

Οι περισσότερες BASIC έχουν μία εντολή GOSUB που μπορεί να χρησιμοποιηθεί για να ενεργοποιήσει συγκεκριμένα τμήματα κώδικα που ονομάζονται υπορουτίνες. Η δήλωση GOSUB δεν είναι ικανοποιητική για πολλούς λόγους και η SuperBASIC προσφέρει σωστά ονομασμένες διαδικασίες με μερικά πολύ χρήσιμα χαρακτηριστικά.

Κοιτάξτε τα ακόλουθα προγράμματα που και τα δύο σχεδιάζουν ένα πράσινο "τετράγωνο" μήκους πλευράς 50 μονάδων οθόνης pixels, στη θέση 200 κατά μήκος και 100 κάτω σε κόκκινο φόντο.

α. Χρησιμοποιώντας το GOSUB

```
100 LET colour = 4 : background = 2
110 LET across = 200
120 LET down = 100
130 LET side = 50
140 GOSUB 800
150 PRINT 'END'
155 STOP
160 REMark
170 PAPER background : CLS
180 BLOCK side, side, across, down, colour
190 RETURN
```

β. Χρησιμοποιώντας μία διαδικασία με παραμέτρους.

```
10 square 4, 50, 20, 100, 2
20 PRINT 'END'
30 DEFine PROCedure square(colour,side,across,
   down,background)
```

```
40   PAPER background : CLS
50   BLOCK side, side, across, down
60 END DEFine
```

Στο πρώτο πρόγραμμα οι τιμές των colour, across, down και side τοποθετούνται με δηλώσεις LET πριν η δήλωση GOSUB ενεργοποιήσει τις γραμμές 170 και 180. Ο έλεγχος μετά επιστρέφεται με τη δήλωση RETURN.

Στο δεύτερο πρόγραμμα οι τιμές δίνονται στην πρώτη γραμμή σαν παράμετροι στην κλήση της διαδικασίας square, που ενεργοποιεί τη διαδικασία και ταυτόχρονα της δίνει τις τιμές που χρειάζεται.

Στην απλούστερη μορφή της μια διαδικασία δεν έχει παραμέτρους. Απλώς χωρίζει ένα συγκεκριμένο τμήμα κώδικα, αν και σ' αυτή την απλή χρήση η διαδικασία έχει το πλεονέκτημα απέναντι στη GOSUB, ότι είναι σωστά ονομασμένη και σωστά απομονωμένη σα μία μονάδα.

Η ισχύς και τα απλοποιητικά αποτελέσματα των διαδικασιών είναι πιο προφανή όσο τα προγράμματα γίνονται μεγαλύτερα. Αυτό που κάνουν οι διαδικασίες καθώς τα προγράμματα μεγαλώνουν είναι όχι τόσο να κάνουν τον προγραμματισμό ευκολώτερο όσο να τον εμποδίσουν να γίνει δυσκολότερος καθώς το πρόγραμμα αυξάνει σε μέγεθος. Το παραπάνω παράδειγμα απλώς επιδεικνύει τον τρόπο λειτουργίας τους σε μία απλή έκφραση.

**ΠΑΡΑΔΕΙΓΜΑΤΑ**

Τα ακόλουθα παραδείγματα δείχνουν την κλίμακα του λεξιλογίου και της σύνταξης της SuperBASIC που παρουσιάστηκε σ' αυτό και στα προηγούμενα κεφάλαια, και θα αποτελέσει τη βάση πάνω στην οποία θα αναπτυχθεί το δεύτερο μέρος αυτού του εγχειριδίου.
Τα γράμματα μιας καρκινικής επιγραφής δίνονται σαν απλά στοιχεία σε εντολές DATA. Το θέμα της λήξης είναι ένας αστερίσκος και δε χρειάζεται να γνωρίζετε τον αριθμό των γραμμάτων της καρκινικής επιγραφής. Κάντε READ τα γράμματα σ' έναν πίνακα και τυπώστε τα από το τέλος προς την αρχή. Μερικές καρκινικές επιγραφές όπως το "MADAM I'M ADAM" δουλεύουν μόνο αν τα διαστήματα και η στίξη αγνοηθούν. Αυτή που χρησιμοποιείται εδώ δουλεύει σωστά.

```
100 REMark Palindromes
110 DIM text$(30)
120 LET text$ = FILL$ (' ',30)
130 LET count = 30
140 REPeat get_letters
150   READ character$
160   IF character$ = '*' THEN EXIT get_letters
170   LET count = count-1
180 LET text$(count) = character$
190 END REPeat get_letters
200 PRINT text$
210 DATA 'A','B','L','E' ,'W','A','S' ,'I' ,'E','R'
220 DATA 'E' ,'I' ,'S','A','W' ,'E','L','B','A','*'
```

Το ακόλουθο πρόγραμμα δέχεται σαν είσοδο αριθμούς από 1 ως 3999 και τους μετατρέπει στο λατινικό αντίστοιχο αριθμό. Δεν παράγει την πιο κομψή μορφή. Παράγει IIII αντί για IV.

```
100 REMark Roman numbers
110 INPUT number
120 RESTORE 210
130 FOR type = 1 TO 7
140 READ letter$, value
150   REPeat output
160     IF number < value : EXIT output
170     PRINT letter$;
180     LET number = number - value
190   END REPeat output
200 END FOR type
210 DATA 'M',1000,'D',500,'C',100,'L',50,'X',10,'V',5,'I',1
```

Πρέπει να μελετήσετε τα παρακάτω παραδείγματα προσεκτικά, χρησιμοποιώντας δοκιμαστικά τρεξίματα αν χρειαστεί, μέχρι να είστε σίγουροι ότι τα καταλαβαίνετε.

**ΣΥΜΠΕΡΑΣΜΑ**

Στη SuperBASIC υπάρχουν πλήρη χαρακτηριστικά δόμησης έτσι ώστε τα στοιχεία του προγράμματος να πηγαίνουν με σειρά ή να παιριάζουν το ένα μέσα στο άλλο ωραία. 'Ολες οι δομές πρέπει να γίνουν γνωστοί στο σύστημα και να ονομαστούν. Υπάρχουν πολλά ενοποιητικά και απλοποιητικά χαρακτηριστικά και πολλές επί πλέον ευκολίες.

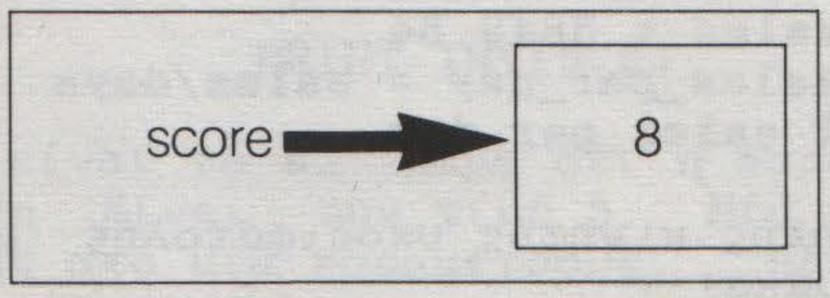
Τα περισσότερα από αυτά εξηγούνται και επιδεικνύονται στα απομένοντα κεφάλαια του εγχειριδίου αυτού, που θα πρέπει να είναι πιο εύκολο στο διάβασμα από τα τμήματα των Δεσμευμένων Λέξεων και των Εννοιών. 'Ομως είναι πιο εύκολο γιατί δε δίνει όλες τις τεχνικές λεπτομέρειες και δεν εξαντλεί κάθε θέμα με το οποίο ασχολείται. Επομένως θα υπάρχουν μερικές περιστάσεις όπου θα χρειαστείτε τη συμβουλή των τμημάτων αναφοράς. Από την άλλη μεριά μερικά προχωρημένα θέματα συζητώνται στα επόμενα κεφάλαια. Λίγοι από εσάς θα τα χρειαστείτε όλα και ίσως προτιμήσετε να παραλείψετε μερικά τμήματα, τουλάχιστον στην πρώτη ανάγνωση.

# ΚΕΦΑΛΑΙΟ 9

# ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ, ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΤΕΣ

θα έχετε παρατηρήσει ότι ένα πρόγραμμα (μία σειρά εντολών) συχνά χρησιμοποιεί δεδομένα για να δουλέψει (είσοδος) και παράγει κάποιο είδος αποτελέσματος (έξοδος). θα έχετε επίσης κατανοήσει ότι υπάρχει εσωτερικός διακανονισμός για την αποθήκευση των δεδομένων. Για να αποφύγουμε τις περιττές τεχνικές εξηγήσεις, συστήσαμε να φαντάζεστε φωλιές και να διαλέγετε ονόματα με νόημα γι' αυτές τις φωλιές. Για παράδειγμα αν χρειάζεται να αποθηκεύσετε έναν αριθμό που αντιπροσωπεύει το αποτέλεσμα της προσομοιωμένης ρίψης δύο ζαριών, φαντάζεστε μία φωλιά με όνομα score που θα περιέχει έναν αριθμό όπως το 8.

Εσωτερικά οι φωλιές είναι αριθμημένες και το σύστημα κρατάει ένα λεξικό όπου συνδέει συγκεκριμένα ονόματα με συγκεκριμένες αριθμημένες φωλιές. Λέμε ότι το όνομα score, δείχνει στη συγκεκριμένη φωλιά (μέσω του εσωτερικού λεξικού).




Η όλη διευθέτηση λέγεται μεταβλητή.

Αυτό που βλέπετε είναι η λέξη score. Λέμε ότι η λέξη score είναι ένας αναγνωριστής. Είναι αυτό που βλέπουμε και αναγνωρίζει την ιδέα που θέλουμε, σ' αυτή την περίπτωση το αποτέλεσμα, 8, της ρίψης δύο ζαριών. Επειδή ο αναγνωριστής είναι αυτό που βλέπουμε, γίνεται αυτό για το οποίο μιλάμε ή γράφουμε ή σκεφτόμαστε. Γράφουμε για το score και την τιμή του σε κάθε συγκεκριμένη στιγμή.

Υπάρχουν τέσσερις τύποι δεδομένων που λέγονται κινητής υποδιαστολής, ακεραίοι, αλφαριθμητικοί και λογικοί και εξηγούνται παρακάτω. Μιλάμε για τύπους δεδομένων παρά για τύπους μεταβλητών επειδή τα δεδομένα μπορούν να υπάρξουν και μόνα τους, για παράδειγμα το 3, 4 ή "Blue hat" σαν τιμή μιας μεταβλητής. Αλλά αν καταλαβαίνετε τους διαφορετικούς τύπους μεταβλητών, θα πρέπει να καταλαβαίνετε και τους διαφορετικούς τύπους δεδομένων.

## ΑΝΑΓΝΩΡΙΣΤΕΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ

[1] 'Ενας αναγνωριστής της SuperBASIC πρέπει να αρχίζει με γράμμα και είναι μία σειρά από

- μικρά ή κεφαλαία γράμματα
- ψηφία ή υπογράμμιση

[2] 'Ενας αναγνωριστής μπορεί να είναι μέχρι 255 χαρακτήρες σε μήκος ώστε πρακτικά δεν υπάρχει όριο.

[3] 'Ενας αναγνωριστής δεν μπορεί να είναι ίδιος με μία λέξη δεσμευμένη της SuperBASIC.

[4] Το όνομα μιας ακέραιης μεταβλητής είναι ένας αναγνωριστής με % στο τέλος.

[5] Το όνομα μιας αλφαριθμητικής μεταβλητής είναι ένας αναγνωριστής με $ στο τέλος.

[6] Οι άλλοι αναγνωριστές δεν πρέπει να χρησιμοποιούν τα σύμβολα % ή $.

[7] 'Ενας αναγνωριστής πρέπει συνήθως να διαλέγεται έτσι ώστε να σημαίνει κάτι σ' έναν αναγνώστη, αλλά για τη SuperBASIC δεν έχει κανένα ιδιαίτερο νόημα εκτός του ότι αναγνωρίζει ορισμένα πράγματα.

## ΜΕΤΑΒΛΗΤΕΣ ΚΙΝΗΤΗΣ ΥΠΟΔΙΑΣΤΟΛΗΣ

Παραδείγματα χρήσης μεταβλητών κινητής υποδιαστολής:

```
100 LET days = 24
110 LET sales = 3649.84
120 LET sales_per_day = sales/days
130 PRINT sales_per_day
```

Η τιμή μιας μεταβλητής κινητής υποδιαστολής μπορεί να είναι οτιδήποτε στην κλίμακα:

-10 εις την 615 έως +10 εις την 615 με 8 σημαντικά ψηφία.

Υποθέτουμε ότι στο παραπάνω πρόγραμμα το sales ήταν κατ' εξαίρεση μόνο 3 πέννες. Αλλάξτε τη γραμμή 110 σε:

```
110 LET sales = 0.03
```

Το σύστημα θα το αλλάξει αυτό σε:

```
110 LET sales = 3E-2
```

Για να το μεταφράσετε αυτό αρχίστε με 3 ή 3.0 και μετακινήστε την υποδιαστολή κατά δύο ψηφία προς τα αριστερά. Αυτό δείχνει ότι:

3E-2 είναι το ίδιο με 0.03

Μετά το τρέξιμο του προγράμματος οι μέσες ημερήσιες πωλήσεις είναι:

1.25E-3 που είναι το ίδιο με 0,00125.

Οι αριθμοί με E λέμε ότι είναι σε εκθετική μορφή:

(βάση) E (εκθέτης) = (βάση) χ 10 εις τον (εκθέτη)

**ΑΚΕΡΑΙΕΣ ΜΕΤΑΒΛΗΤΕΣ**

Οι ακέραιες μεταβλητές μπορούν να έχουν μόνο ακέραιες τιμές από -32768 ως 32767. Τα ακόλουθα είναι παραδείγματα δεκτών ονομάτων ακέραιων μεταβλητών που πρέπει να τελειώνουν με %.

```
LET count% = 10
LET six_tally% = RND(10)
LET number_3% = 3
```

Το μόνο μειονέκτημα των ακέραιων μεταβλητών, όταν χρειάζονται ακέραιοι αριθμοί, είναι το κάπως παραπλανητικό σύμβολο % στο τέλος του αναγνωριστή. Δεν έχει καμιά σχέση με την έννοια της ποσοστιαίας αναλογίας. Είναι απλώς ένα βολικό σύμβολο σαν ετικέτα για να δείχνει ότι η μεταβλητή είναι ακέραια.

**ΑΡΙΘΜΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ**

Η χρήση των συναρτήσεων είναι σαν να φτιάχνουμε μία ομελέτα. Βάζουμε ένα αβγό που κατεργάζεται σύμφωνα με ορισμένους κανόνες (τη συνταγή) και παίρνουμε μία ομελέττα. Για παράδειγμα η συνάρτηση INT παίρνει οποιοδήποτε αριθμό σαν είσοδο, και εξάγει το ακέραιο μέρος τους αριθμού. 'Οτι εισάγεται σε μία συνάρτηση λέγεται παράμετρος ή όρισμα. Η INT είναι μία συνάρτηση που δίνει το ακέραιο μέρος μιας παράστασης. Μπορείτε να γράψετε:

```
PRINT INT(5.6)
```

και η έξοδος θα είναι το 5. Λέμε ότι η παράμετρος είναι 5.6 και η συνάρτηση δίνει την τιμή 5. Μία συνάρτηση μπορεί να έχει περισσότερες από μία παραμέτρους. 'Ηδη συναντήσατε:

```
RND(1 TO 6)
```

που είναι μία συνάρτηση με δύο παραμέτρους. Αλλά οι συναρτήσεις πάντα επιστρέφουν ακριβώς μία τιμή. Αυτό πρέπει να είναι έτσι γιατί μπορείτε να βάλετε συναρτήσεις μέσα σε παραστάσεις. Για παράδειγμα:

```
PRINT 2 * INT(5.6)
```

θα δώσει την έξοδο 10. Είναι πολύ σημαντική ιδιότητα των συναρτήσεων το ότι μπορείτε να τις χρησιμοποιείτε σε παραστάσεις. 'Ετσι πρέπει να επιστρέφουν μία μόνο τιμή που μετά χρησιμοποιείται στην παράσταση. Οι INT και RND είναι συναρτήσεις του συστήματος, είναι μαζί με το σύστημα, αλλά αργότερα θα δείτε πως να γράφετε τις δικές σας.

Το ακόλουθο παράδειγμα δείχνει μία συνήθη χρήση της συνάρτησης INT.

```
100 REMark Rounding
110 INPUT decimal
120 PRINT INT(decimal + 0.5)
```

Στο παράδειγμα εισάγετε έναν δεκαδικό αριθμό και η έξοδος βγαίνει στρογγυλοποιημένη. 'Ετσι το 4.7 θα γίνει 5 αλλά το 4.3 θα γίνει 4. Μπορείτε να καταφέρετε το ίδιο πράγμα χρησιμοποιώντας μία ακέραια μεταβλητή και εξαναγκασμό.

Με τις τριγωνομετρικές συναρτήσεις θα ασχοληθούμε σε επόμενο τμήμα αλλά μερικές συνήθεις αριθμητικές συναρτήσεις δίνονται στον παρακάτω πίνακα:

| Συνάρτηση | Αποτέλεσμα | Παραδείγματα | Επιστρεφόμενες τιμές |
|---|---|---|---|
| ABS | Απόλυτη τιμή | ABS(7) | 7 |
| | ή τιμή χωρίς πρόσημο | ABS(-4.3) | 4.3 |
| INT | Ακέραιο μέρος ενός | INT(2.4) | 2 |
| | αριθμού κινητής | INT(0.4) | 0 |
| | υποδιαστολής | INT(-2.7) | -3 |
| SQRT | Τετραγωνική ρίζα | SQRT(2) | 1.414214 |
| | | SQRT(16) | 4 |
| | | SQRT(2.6) | 1.612452 |

Υπάρχει ένας τρόπος υπολογισμού τετραγωνικών ριζών που είναι ευκολονόητος. Για να υπολογίσετε την τετραγωνική ρίζα του 8, πρώτα κάντε μία υπόθεση. Δεν έχει σημασία πόσο λάθος μπορεί να είναι η υπόθεσή σας. Ας πούμε ότι παίρνετε απλά το μέσο του 8 σαν την πρώτη υπόθεση, που είναι το 4.

Επειδή το 4 είναι μεγαλύτερο από την τετραγωνική ρίζα του 8 τότε το 8/4 πρέπει να είναι μικρότερο απ' αυτήν. Το αντίστροφο επίσης ισχύει. Αν είχατε υποθέσει το 2 που είναι μικρότερο από την τετραγωνική ρίζα, τότε το 8/2 πρέπει να είναι μεγαλύτερο απ' αυτήν.

Επομένως αν πάρουμε οποιαδήποτε υπόθεση και υπολογίσουμε το αριθμός/υπόθεση τότε έχουμε δύο αριθμούς, έναν πολύ μικρό και έναν πολύ μεγάλο. Παίρνουμε το μέσο όρο αυτών των αριθμών σαν την επόμενη μας προσέγγιση κι έτσι πλησιάζουμε στη σωστή απάντηση.

Επαναλαμβάνουμε αυτή τη διαδικασία μέχρι που οι διαδοχικές προσεγγίσεις να είναι τόσο κοντά ώστε να διαφέρουν ελάχιστα.

```
100 REMark Square Roots
110 LET number = 8
120 LET approx = number/2
130 REPeat root
140   LET newval = (approx + number/approx) /2
150   IF newval == approx then EXIT root
160   LET approx = newval
170 END REPeat root
180 PRINT 'Square root of' ! number ! 'is' ! newval
```

παράδειγμα εξόδου

```
Square root 8 is 2.828427
```

Σημειώστε ότι το υπό όρους EXIT από το βρόχο πρέπει να είναι στη μέση. Οι παραδοσιακές δομές δεν τα καταφέρνουν τόσο καλά μ' αυτή την περίπτωση όπως η SuperBASIC.

Το σύμβολο == στη γραμμή 150 σημαίνει "κατά προσέγγιση ίσο με", και γίνεται με προσέγγιση .0000001 στις τιμές που συγκρίνονται.

## ΑΡΙΘΜΗΤΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

Η SuperBASIC επιτρέπει τις συνήθεις μαθηματικές λειτουργίες. θα παρατηρήσετε ότι είναι σαν συναρτήσεις με ακριβώς δύο ορίσματα η κάθε μία. Είναι συμβατικό σ' αυτές τις περιπτώσεις να μπαίνει ένα όρισμα σε κάθε πλευρά του συμβόλου. Μερικές φορές η λειτουργία παριστάνεται μ' ένα οικείο σύμβολο όπως + και *. Μερικές φορές η λειτουργία παριστάνεται με μία δεσμευμένη λέξη όπως DIN ή MOD αλλά δεν υπάρχει πραγματική διαφορά. Οι αριθμητικές λειτουργίες έχουν μία σειρά προτεραιότητας. Για παράδειγμα το αποτέλεσμα του:

```
PRINT 7 + 3*2
```

είναι 13 επειδή ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα. 'Ομως:

```
PRINT (7 + 3)*2
```

θα εξάγει 20 επειδή η παρένθεση ξεπερνά τη συνήθη προτεραιότητα. 'Οπως θα δείτε αργότερα, μπορούν να γίνουν τόσο πολλά πράγματα με τις εκφράσεις της SuperBASIC, ώστε μία πλήρης δήλωση σχετικά με την προτεραιότητα δεν μπορεί να γίνει σ' αυτό το στάδιο (δέστε τις 'Εννοιες αν θέλετε) αλλά οι λειτουργίες με τις οποίες ασχολούμαστε τώρα έχουν την ακόλουθη προτεραιότητα.

ψηλότερη: ύψωση σε δύναμη

πολλαπλασιασμός και διαίρεση (συμπεριλαμβανομένων των DIV, MOD)

χαμηλότερη: πρόσθεση και αφαίρεση

Τα σύμβολα + και - επίσης χρησιμοποιούνται με ένα μόνο όρισμα που απλώς δηλώνεται θετικό ή αρνητικό. Τα σύμβολα που χρησιμοποιούνται μ' αυτόν τον τρόπο, έχουν την ψηλότερη προτεραιότητα απ' όλα και μπορούν να ξεπεραστούν μόνο με τη χρήση παρενθέσεων.

Τελικά αν δύο σύμβολα έχουν την ίδια προτεραιότητα, τότε η λειτουργία που βρίσκεται πιο αριστερά, εκτελείται πρώτη ώστε:

```
PRINT 7-2 + 5
```

θα κάνει την αφαίρεση πριν από την πρόσθεση. Αυτό ίσως να είναι σημαντικό αν ποτέ ασχοληθείτε με πολύ μεγάλους ή πολύ μικρούς αριθμούς.

| Λειτουργία | Σύμβολο | Παράδειγμα | Αποτέλεσμα | Σημειώσεις |
|---|---|---|---|---|
| Πρόσθεση | + | 7.66 | 13.3 | |
| Αφαίρεση | - | 7-66 | 0.4 | |
| Πολλαπλα- | * | 3*2.1 | 6.3 | |
| σιασμός | | 2.1*(-3) | -6.3 | |
| Διαίρεση | / | 7/2 | 3.5 | Μη διαιρείτε με το μηδέν |
| | | -17/5 | -3.4 | |
| 'Υψωση σε δύναμη | ^ | 4^1.5 | 8 | |
| 'Υψωση σε δύναμη (ακέραιοι) | (^) | 3(^)2 | 9 | μόνο ακέραιοι |
| διαίρεση ακεραίων | DIV | -8 DIV 2 | -4 | μόνο ακέραιοι |
| | | 7 DIV 2 | 3 | μη διαιρείτε με το μηδέν |
| | | -8 DIV 2 | -4 | |
| ισουπόλοιπο (modulo) | MOD | 13 MOD 5 | 3 | |
| | | 21 MOD 7 | 0 | |
| | | -17 MOD 8 | -7 | |

Το MOD επιστρέφει το υπόλοιπο μιας ακέραιας διαίρεσης. Κάθε προσπάθεια διαίρεσης με μηδέν θα παράγει ένα λάθος και θα σταματήσει την εκτέλεση του προγράμματος.

## ΑΡΙΘΜΗΤΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

Μιλώντας αυστηρά, μία αριθμητική παράσταση είναι μία παράσταση που εκτιμάται μ' έναν αριθμό και υπάρχουν περισσότερες δυνατότητες απ' όσες χρειάζεται να συζητήσουμε εδώ. Η SuperBASIC σας επιτρέπει να κάνετε περίπλοκα πράγματα αν θέλετε αλλά επίσης σας επιτρέπει να κάνετε απλά πράγματα με απλούς τρόπους. Σ' αυτό το τμήμα συγκεντρώνουμε την προσοχή μας στις συνήθεις άμεσες χρήσεις των μαθηματικών χαρακτηριστικών.

Βασικά, οι αριθμητικές εκφράσεις στη SuperBASIC είναι οι ίδιες με αυτές των μαθηματικών αλλά πρέπει να βάλετε ολόκληρη την έκφραση σε μορφή μιας σειράς.

$$\frac{5 + 3}{6-4}$$

γίνεται στη SuperBASIC (ή σε άλλες BASIC):

```
(5 + 3)/(6-4)
```

ΠΑΡΑΔΕΙΓΜΑ 1

Στην άλγεβρα υπάρχει μία παράσταση για μία λύση εξίσωσης δευτέρου βαθμού:

$$ax^2 + bx + c = 0$$

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Μία λύση σε μαθηματική γραφή είναι:

$$2x^2 - 3x + 1 = 0$$

Αν ξεκινήσουμε με την εξίσωση:

ΠΑΡΑΔΕΙΓΜΑ 2

Το ακόλουθο πρόγραμμα θα βρει μία λύση.

```
100 READ a,b,c
110 PRINT 'Root is' ! (-b + SQRT(B ^ 2 - 4*a*c))/(2*a)
120 DATA 2,-3,1
```

Σε προβλήματα όπου χρειάζεται να προσομοιώσουμε το μοίρασμα τραπουλόχαρτων, μπορείτε να ορίσετε αντιστοιχία μεταξύ των χαρτιών και των αριθμών 1 ως 52 όπως παρακάτω:

```
1 ως 13      Ace, two.....king of hearts
14 ως 26     Ace, two.....king of clubs
27 ως 39     Ace, two.....king of diamonds
40 ως 52     Ace, two.....king of spades
```

'Ενα συγκεκριμένο χαρτί μπορεί να αναγνωριστεί όπως φαίνεται:

```
100 REM Card identification
110 LET card = 23
120 LET suit = (card-1) DIV 13
130 LET value = card MOD 13
140 IF value = 0 THEN LET value = 13
150 IF value = 1 THEN PRINT "Ace of ";
160 IF value >= 2 AND value <= 10 THEN PRINT value ! "of ";
170 IF value = 11 THEN PRINT "Jack of ";
180 IF value = 12 THEN PRINT "Queen of ";
190 IF value = 13 THEN PRINT "King of ";
200 IF suit = 0 THEN PRINT "hearts"
210 IF suit = 1 THEN PRINT "clubs"
220 IF suit = 2 THEN PRINT "diamonds"
230 IF suit = 3 THEN PRINT "spades"
```

Υπάρχει μία νέα ιδέα στο πρόγραμμα. Είναι στη γραμμή 160. Το νόημα είναι ότι ο αριθμός τυπώνεται μόνο όταν και οι δύο λογικές δηλώσεις είναι αληθείς. Αυτές είναι:

η τιμή μεγαλύτερη ή ίση του 2
και η τιμή είναι μικρότερη ή ίση του 10

Τα χαρτιά έξω απ' αυτήν την κλίμακα είναι είτε άσσοι ή φιγούρες και πρέπει να μεταχειριστούν διαφορετικά.

Σημειώστε επίσης τη χρήση του ! στη δήλωση PRINT για να δώσει ένα διάστημα, και του ; για να συνεχίσει η έξοδος στην ίδια γραμμή.

Υπάρχουν δύο ομάδες μαθηματικών συναρτήσεων που δεν τις έχουμε συζητήσει ακόμη. Είναι οι τριγωνομετρικές και οι λογαριθμικές. Μπορεί να χρειαστείτε τις πρώτες για την οργάνωση απεικονίσεων στην οθόνη. Οι τύποι των συναρτήσεων είναι επίσης πλήρως ορισμένοι στις Δεσμευμένες Λέξεις.

## ΛΟΓΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ

Μιλώντας αυστηρά, η SuperBASIC δεν επιτρέπει λογικές μεταβλητές αλλά σας επιτρέπει να χρησιμοποιείτε άλλες μεταβλητές σαν λογικές. Για παράδειγμα μπορείτε να τρέξετε το ακόλουθο πρόγραμμα:

```
100 REMark Logical Variable
110 LET hungry = 1
120 IF hungry THEN PRINT "Have a bun"
```

θα περιμένατε μία λογική παράσταση στη γραμμή 120 αλλά εκεί υπάρχει μόνο η αριθμητική μεταβλητή hungry. Το σύστημα μεταφράζει την τιμή 1 του hungry σαν αληθή και η έξοδος είναι:

```
Have a bun
```

Αν η γραμμή ήταν:

```
LET hungry = 0
```

τότε δεν θα υπάρχει έξοδος. Το σύστημα μεταφράζει το μηδέν ως ψευδές και όλες τις άλλες τιμές ως αληθείς. Αυτό είναι χρήσιμο, αλλά μπορείτε να μεταμφιέσετε την αριθμητική μορφή του hungry γράφοντας

```
100 REMark Logical Variable
110 LET true = 1 : false = 0
120 LET hungry = true
130 IF hungry THEN PRINT "Have a bun"
```

## ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ

Υπάρχουν πολλά να πούμε για το χειρισμό των σειρών χαρακτήρων ή των αλφαριθμητικών μεταβλητών κι έτσι τα αφήνουμε για ένα χωριστό κεφάλαιο.

## ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 9

1. 'Ενας πλούσιος πωλητής πετρελαίου, στοιχηματίζει στρίβοντας ένα νόμισμα με τον ακόλουθο τρόπο. Αν τύχει κεφάλι, τότε παίρνει 1. Αν τύχει γράμματα τότε το

ξαναστρίβει αλλά η πιθανή αμοιβή διπλασιάζεται. Αυτό επαναλαμβάνεται έτσι ώστε οι αμοιβές είναι όπως φαίνονται:

ΡΙΞΙΜΟ: 1 2 3 4 5 6 7
ΑΜΟΙΒΕΣ: 1 22 3 8 16 32 64

Προσομοιώνοντας το παιχνίδι, προσπαθήστε να αποφασίσετε ποια θα ήταν μία δίκαιη αρχική πληρωμή για ένα τέτοιο παιχνίδι:

α. αν ο παίκτης περιορίζεται σ' ένα μέγιστο επτά ριξιών ανά παιχνίδι

β. αν δεν υπάρχει μέγιστος αριθμός ριξιών.

2. Ο Bill και ο Ben συμφωνούν να σχηματίσουν με τον ακόλουθο τρόπο. Μόλις δοθεί ένα σήμα, ο καθένας διαιρεί τα χρήματα του στα δύο και δίνει τα μισά στον άλλο παίκτη. Μετά καθένας διαιρεί το νέο του σύνολο και δίνει τα μισά στον άλλο. Δείξτε τι συμβαίνει καθώς το παιχνίδι συνεχίζεται αν ο Bill ξεκινά με 16 πέννες και ο Ben ξεκινά με 64 πέννες.

3. Τι συμβαίνει αν το παιχνίδι αλλάξει έτσι ώστε καθένας να δίνει στον άλλο ένα ποσό ίσο με μισά απ' αυτά που έχει ο άλλος;

4. Γράψτε ένα πρόγραμμα που να σχηματίζει τυχαίες λέξεις τριών γραμμάτων που επιλέγονται από τα A, B, C, D και τις τυπώνει μέχρι να εμφανιστεί το "BAD".

5. Τροποποιήστε το τελευταίο πρόγραμμα ώστε να τελειώνει όταν εμφανιστεί οποιαδήποτε πραγματική λέξη τριών γραμμάτων.

# ΚΕΦΑΛΑΙΟ 10
# ΛΟΓΙΚΗ

Αν έχετε διαβάσει προηγούμενα κεφαλαία μάλλον θα συμφωνείτε ότι η επανάληψη, η λήψη αποφάσεων και ο χωρισμός των αποστολών σε υποαποστολές είναι βασικές ιδέες στην ανάλυση, το σχεδιασμό και τη γραφή των προγραμμάτων. Δύο από αυτές τις ιδέες, η επανάληψη και η λήψη αποφάσεων χρειάζονται λογικές παραστάσεις όπως αυτές στις ακόλουθες γραμμές προγράμματος:

```
IF score = 7 THEN EXIT throws
IF suit = THEN PRINT "spades"
```

Η πρώτη κάνει EXIT από ένα βρόχο REPeat. Η δεύτερη απλώς αποφασίζει να κάνει κάτι ή να μην το κάνει. Μία μαθηματική παράσταση εκτιμάται σε μία από εκατομμύρια πιθανές αριθμητικές τιμές. 'Ομοια μία αλφαριθμητική παράσταση μπορεί να εκτιμηθεί σε εκατομμύρια πιθανές σειρές χαρακτήρων. Μπορεί να σας φανεί παράξενο ότι οι λογικές παραστάσεις, που λέμε ότι έχουν μεγάλη σημασία, μπορούν να εκτιμηθούν σε μία από δύο μόνο πιθανές τιμές: αληθής ή ψευδής.

Στην περίπτωση του:

```
score = 7
```

αυτό είναι προφανώς σωστό. Το score είτε είναι 7 είτε δεν είναι! Η παράσταση πρέπει να είναι αληθής ή ψευδής - υποθέτοντας ότι έχει νόημα. Μπορεί να τύχει να μην ξέρετε κάποτε την τιμή αλλά αυτό θα συζητηθεί παρακάτω.

Πρέπει να είστε λίγο πιο προσεκτικοί στις παραστάσεις που συμπεριλαμβάνουν δεσμευμένες λέξεις όπως OR, AND, NOT αλλά αξίζει τον κόπο να τις ερευνήσετε, γιατί είναι ουσιώδεις στον καλό προγραμματισμό. θα γίνουν ακόμη πιο σημαντικές με την τάση προς άλλα είδη γλωσσών που βασίζονται περισσότερο στις ακριβείς περιγραφές αυτού που απαιτείται παρά αυτού που πρέπει να κάνει ο υπολογιστής.

## AND

Η λέξη AND στη SuperBASIC είναι όπως η λέξη "και" στη γλώσσα μας. Δέστε το ακόλουθο πρόγραμμα.

```
100 REMark AND
110 PRINT "Enter two values"/"1 for TRUE or 0 for FALSE"
120 INPUT raining, hole_in_roof
130 IF raining AND hole_in_roof THEN PRINT "Get wet"
```

'Οπως στην αληθινή ζωή, θα βραχείτε μόνο αν βρέχει και υπάρχει τρύπα στη στέγη. Αν μία (ή δύο) από τις απλές λογικές μεταβλητές

raining
hole_in_roof

είναι ψευδής, τότε η σύνθετη λογική παράσταση:

raining AND hole_in_roof

είναι επίσης ψευδής. Χρειάζονται δύο αληθείς τιμές για να κάνουν ολόκληρη την έκφραση αληθή. Αυτό μπορείτε να το δείτε από τους παρακάτω κανόνες. Μόνο όταν η σύνθετη παράσταση είναι αληθής τότε θα βραχείτε.

| raining | hole__in__roof | raining AND hole__in__roof | effect |
|---|---|---|---|
| FALSE | FALSE | FALSE | DRY |
| FALSE | TRUE | FALSE | DRY |
| TRUE | FALSE | FALSE | DRY |
| TRUE | TRUE | TRUE | WET |

Κανόνες του AND

**OR**

Στην καθημερινή ζωή η λέξη "ή" χρησιμοποιείται με δύο τρόπους. Μπορούμε να επιδείξουμε τη συμπεριληπτική (inclusive) χρήση του αν σκεφτούμε έναν αρχηγό του παιχνιδιού κρίκετ που ψάχνει για παίκτες. θα ρωτούσε "Μπορείς να χτυπάς ή να κυλάς τη σφαίρα;". θα ήταν ευχαριστημένος αν ένας παίκτης μπορούσε να κάνει μόνο ένα πράγμα καλά, αλλά θα ήταν επίσης ευχαριστημένος αν κάποιος μπορούσε και τα δύο. 'Ετσι στον προγραμματισμό: μία σύνθετη έκφραση που χρησιμοποιεί είναι αληθής αν και οι δύο ή μόνο η μία από τις απλές εντολές ή μεταβλητές είναι αληθής. Δοκιμάστε το ακόλουθο πρόγραμμα:

```
100 REMark OR test
110 PRINT "Enter two values /1 for TRUE or 0 for FALSE"
120 INPUT "Can you bat?", batsman
130 INPUT "Can you bowl?", bowler
140 IF batsman OR bowler THEN PRINT "In the team"
```

| **batsman** | **bowler** | **batsman OR bowler** | **αποτέλεσμα** |
|---|---|---|---|
| ΨΕΥΔΗΣ | ΨΕΥΔΗΣ | ΨΕΥΔΗΣ | έξω από την ομάδα |
| ΨΕΥΔΗΣ | ΑΛΗΘΗΣ | ΑΛΗΘΗΣ | μέσα στην ομάδα |
| ΑΛΗΘΗΣ | ΨΕΥΔΗΣ | ΑΛΗΘΗΣ | μέσα στην ομάδα |
| ΑΛΗΘΗΣ | ΑΛΗΘΗΣ | ΑΛΗΘΗΣ | μέσα στην ομάδα |

Κανόνες του OR

'Οταν το συμπεριληπτικό χρησιμοποιείται τότε μία αληθής τιμή σε μία από τις απλές εντολές θα δώσει αληθή τιμή στη σύνθετη

παράσταση. Αν ο Ian Botham, ο Αγγλος παίκτης όλων των γύρων, επρόκειτο να απαντήσει στις ερωτήσεις και σαν bowler και σαν batsman, τότε και οι δύο απλές δηλώσεις θα ήταν αληθής και μαζί μ' αυτές και η σύνθετη παράσταση. 'Ετσι θα έμπαινε στην ομάδα.

Αν γράψετε 0 για την ψευδή και 1 για την αληθή θα πάρετε όλους τους δυνατούς συνδυασμούς, μετρώντας σε δυαδικό:

00
01
10
11

**NOT και παρένθεση**

Η λέξη NOT έχει τις προφανείς σημασίες:

- NOT αληθής είναι το ίδιο όπως ψευδής

- NOT ψευδής είναι το ίδιο όπως αληθής

'Ομως πρέπει να προσέχετε. Υποθέτουμε ότι κρατάτε ένα κόκκινο τρίγωνο και λέτε ότι είναι:

NOT red AND square

Αυτό στη γλώσσα μας μπορεί να είναι διφορούμενο. Αν εννοείτε:

(NOT red) AND square

τότε για ένα κόκκινο τρίγωνο η παράσταση είναι ψευδής. Αν εννοείτε:

NOT(red AND square)

Για ένα κόκκινο τρίγωνο η συνολική παράσταση είναι αληθής.

Πρέπει να υπάρχει ένας κανόνας στον προγραμματισμό για να φαίνεται καθαρά τι σημαίνει. Ο κανόνας είναι ότι το NOT προηγείται του AND κι έτσι η μετάφραση:

(NOT red) AND square

είναι η σωστή. Είναι το ίδιο όπως:

NOT red AND square

Για να πάρετε την άλλη μετάφραση πρέπει να χρησιμοποιήσετε παρένθεση. Αν πρέπει να χρησιμοποιήσετε μία πολύπλοκη λογική παράσταση, είναι καλύτερα να χρησιμοποιείτε παρενθέσεις και NOT αν η χρήση τους φυσιολογικά αντανακλά αυτό που θέλετε. Αλλά μπορείτε πάντα να αφαιρέσετε τις παρενθέσεις με τη χρήση των ακόλουθων νόμων (αποδίδονται στον Augustus De Morgan)

NOT (a AND b) είναι το ίδιο με NOT a OR NOT b
NOT (a OR b) είναι το ίδιο με NOT a AND NOT b

Για παράδειγμα:

NOT (tall AND fair) είναι το ίδιο με
NOT tall OR NOT fair

NOT (hungry OR thirsty) είναι το ίδιο με
NOT hungry AND NOT thirsty

Δοκιμάστε το εισάγοντας:

```
100 REMark NOT and brackets
110 PRINT "Enter two values"/" 1 for TRUE or 0 for FALSE"
120 INPUT "tall"; tall
130 INPUT "fair"; fair
140 IF NOT (tall AND fair) THEN PRINT "FIRST"
150 IF NOT tall OR NOT fair THEN PRINT "SECOND"
```

'Οποιο συνδυασμό αριθμών αν δώσετε για είσοδο, η έξοδος θα είναι πάντα ή δύο λέξεις ή καμμία, ποτέ μία. Αυτό δείχνει ότι οι δύο σύνθετες λογικές εκφράσεις είναι ισοδύναμες.

**XOR - αποκλειστικό OR**

Υποθέτουμε ότι ένας επαγγελματικός παίκτης του γκολφ ήθελε ένα βοηθό είτε για να κρατά το μαγαζί του είτε για να δίνει μαθήματα γκολφ. Αν εμφανίζοταν κάποιος υποψήφιος και με τις δύο ικανότητες μπορεί να μην έπαιρνε τη δουλειά γιατί ο επαγγελματίας θα φοβόταν ότι ένας τέτοιος ικανός βοηθός ίσως προσπαθούσε να του πάρει τον έλεγχο. θα δεχόνταν έναν καλό παίκτη που δεν θα μπορούσε να κρατήσει το μαγαζί. Επίσης θα δεχόταν έναν κακό παίκτη που θα μπορούσε να κρατήσει το μαγαζί. Αυτή είναι μία κατάσταση αποκλειστικού κάθε ένα είναι δεκτό αλλά όχι και τα δύο. Το ακόλουθο πρόγραμμα θα δοκιμάσει τους υποψήφιους:

```
100 REMark XOR test
110 PRINT "ENTER 1 for yes or 0 for no."
120 INPUT "Can you run a shop?", shop
130 INPUT "Can you teach golf?", golf
140 IF shop XOR golf THEN PRINT "Suitable"
```

Οι μόνοι συνδυασμοί απαντήσεων που θα προκαλούσαν την έξοδο "Suitable" είναι (0 και 1) ή (1 και 0). Οι κανόνες του XOR δίνονται παρακάτω:

| Able to run shop | Able to teach | Shop XOR teach | effect |
|---|---|---|---|
| FALSE | FALSE | FALSE | no job |
| FALSE | TRUE | TRUE | gets the job |
| TRUE | FALSE | TRUE | gets the job |
| TRUE | TRUE | FALSE | no job |

Κανόνες του XOR

Προτεραιότητες

Η σειρά προτεραιότητας των λογικών χειριστών είναι (πρώτα η μεγαλύτερη):

NOT
AND
OR, XOR

Για παράδειγμα η παράσταση:

```
rich tall AND fair
```

σημαίνει το ίδιο με:

```
rich (tall AND fair)
```

Η λειτουργία AND γίνεται πρώτη. Για να δείτε ότι οι δύο λογικές παραστάσεις έχουν το ίδιο αποτέλεσμα, τρέξτε το ακόλουθο πρόγραμμα:

```
100 REMark Priorities
110 PRINT "Enter three values"\"Type 1 for Yes and 0 for No"!
120 INPUT rich,tall,fair
130 IF rich OR tall AND fair THEN PRINT "YES"
140 IF rich OR (tall AND fair) THEN PRINT "AYE"
```

Οποιοδήποτε συνδυασμό τριών μηδενικών ή μονάδων εισάγετε στη γραμμή 110, η έξοδος θα είναι είτε τίποτα ή:

```
YES
AYE
```

Για να είστε σίγουροι ότι δοκιμάσατε όλες τις πιθανότητες, εισάγετε δεδομένα που να σχηματίζουν οκτώ τριψήφιους δυαδικούς αριθμούς από 000 ως 111:

```
000 001 010 011 100 101 110 111
```

## ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 10

1. Βάλτε δέκα αριθμούς σε μία δήλωση DATA. Κάντε READ κάθε αριθμό και αν είναι μεγαλύτερος του 20, τότε τυπώστε τον.

2. Δοκιμάστε όλους τους αριθμούς από 1 ως 100 και τυπώστε μόνο αυτούς που είναι τέλεια τετράγωνα ή διαιρετοί με το 7.

3. Τα παιχνίδια περιγράφονται σαν Safe (ασφαλή) (S) ή Unsafe (επικίνδυνα) (U), Expensive (ακριβά) (E) ή Cheap (φθηνά) (C) και είτε για Girls (κορίτσια) (G), ή Boys (αγόρια) (B) ή Anyone (για όλους) (A). Μία τριάδα γραμμάτων κωδικοποιεί τις ιδιότητες κάθε παιχνιδιού. Τοποθετήστε πέντε τέτοιες τριάδες σε μία δήλωση DATA και μετά ψάξτε την τυπώνοντας μόνο τα παιχνίδια που είναι ασφαλή και κατάλληλα για κορίτσια.

4. Τροποποιήστε το πρόγραμμα 3 για να τυπώνει τα παιχνίδια που είναι ακριβά και επικίνδυνα.

5. Τροποποιήστε το πρόγραμμα 3 για να τυπώνει αυτά που είναι ασφαλή, φθηνά και κατάλληλα για όλους.

# ΚΕΦΑΛΑΙΟ 11
# ΧΕΙΡΙΣΜΟΣ ΚΕΙΜΕΝΟΥ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ

'Εχετε χρησιμοποιήσει αλφαριθμητικές μεταβλητές για να αποθηκεύσετε σειρές χαρακτήρων και ξέρετε ότι οι κανόνες χειρισμού των αλφαριθμητικών μεταβλητών ή σταθερών δεν είναι ίδιοι μ' αυτούς των αριθμητικών μεταβλητών ή σταθερών. Η SuperBASIC προσφέρει μία πλήρη κλίμακα ευκολιών για αποδοτικό χειρισμό σειρών χαρακτήρων. Ειδικώτερα η ιδέα του τεμαχισμού μιας αλφαριθμητικής μεταβλητής, επεκτείνει και απλοποιεί τη διαδικασία χειρισμού των τεμαχίων μιας αλφαριθμητικής.

## ΟΡΙΣΜΟΣ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ

Ο χώρος για τις αλφαριθμητικές μεταβλητές κατανέμεται όπως απαιτεί το πρόγραμμα. Για το παράδειγμα, οι γραμμές:

```
100 LET words$ = "LONG"
110 LET words$ = "LONGER"
120 PRINT words$
```

θα έχουν αποτέλεσμα να τυπωθεί η λέξη έξι γραμμάτων LONGER. Η πρώτη γραμμή είχε αποτέλεσμα να κατανεμηθεί χώρος για τέσσερα γράμματα αλλά αυτή η κατανομή ξεπερνάται από τη δεύτερη γραμμή που απαιτεί χώρο για έξι χαρακτήρες.

Είναι όμως δυνατό να ορίσουμε τη διάσταση (δηλαδή να κρατήσουμε χώρο) της αλφαριθμητικής μεταβλητής, οπότε το μέγιστο μήκος ορίζεται και η μεταβλητή συμπεριφέρεται σε όλες τις περιπτώσεις σαν πίνακας.

## ΕΝΩΣΗ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ

'Ισως θέλετε να κατασκευάσετε αρχεία για επεξεργασία δεδομένων από διάφορες πηγές. Υποθέτουμε για παράδειγμα, ότι είστε δάσκαλος και θέλετε να αποθηκεύσετε μία ομάδα τριών βαθμών για κάθε μαθητή στη Λογοτεχνία, Ιστορία και Γεωγραφία. Οι βαθμοί βρίσκονται σε μεταβλητές όπως φαίνονται:

```
lit$      62      hist$     56      geog$     71
```

Σαν ένα μέρος του χειρισμού του αρχείου, ίσως θέλετε να συνδυάσετε τις τρεις τιμές των μεταβλητών σε μία σειρά έξι χαρακτήρων, με όνομα marks$. Απλώς γράφετε:

```
LET mark$ = lit$ & hist$ & geog$
```

Δημιουργήσατε ακόμη μία μεταβλητή όπως φαίνεται:

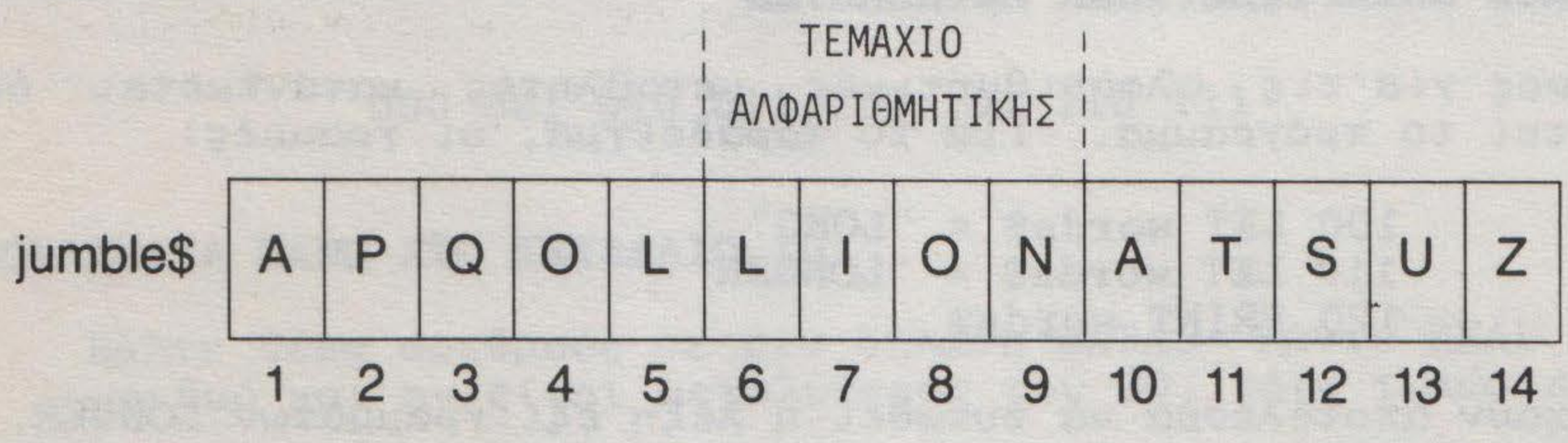
```
mark$      625671
```

Να θυμάστε όμως ότι πρόκειται για μια σειρά χαρακτήρων που τυχαίνει να είναι αριθμοί και όχι για τον ίδιο τον αριθμό.

## ΑΝΤΙΓΡΑΦΗ ΤΕΜΑΧΙΟΥ ΑΛΦΑΡΙΘΜΗΤΙΚΗΣ ΜΕΤΑΒΛΗΤΗΣ

'Ενα τεμάχιο αλφαριθμητικής μεταβλητής είναι ένα μέρος της αλφαριθμητικής. Μπορεί να είναι οτιδήποτε από έναν απλό χαρακτήρα μέχρι ολόκληρη την αλφαριθμητική. Για να αναγνωρίσετε το τεμάχιο πρέπει να γνωρίζετε τις θέσεις των απαιτούμενων χαρακτήρων.

Υποθέτουμε ότι κατασκευάζετε ένα παιδικό παιχνίδι στο οποίο πρέπει να αναγνωρίζουν μια λέξη κρυμμένη μέσα σε πολλά γράμματα. Κάθε γράμμα έχει έναν εσωτερικό αριθμό - ένα δείκτη - που ανταποκρίνεται στη θέση του μέσα στην αλφαριθμητική μεταβλητή. Υποθέτουμε ότι ολόκληρη η αλφαριθμητική μεταβλητή αποθηκεύεται στη μεταβλητή jumble$ με υπόδειξη λιοντάρι (LION).




Μπορείτε να δείτε ότι η απάντηση ορίζεται από τους αριθμούς 6 ως 9 που δείχνουν το σημείο όπου βρίσκεται. Μπορείτε να εξάγετε την απάντηση όπως δείχνεται:

```
100 LET jumble$ = "APQOLLIONATSUZ"
110 LET an$ = jumble$(6 TO 9)
120 PRINT an$
```

## ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΤΕΜΑΧΙΟΥ ΑΛΦΑΡΙΘΜΗΤΙΚΗΣ

Τώρα υποθέτουμε ότι θέλετε να κάνετε το κρυμμένο ζώο, ταύρο. Μπορείτε να γράψετε δύο επιπλέον γραμμές:

```
130 LET jumble$(6 TO 9) = "BULL"
140 PRINT jumble$
```

Η έξοδος από ολόκληρο το πρόγραμμα των πέντε γραμμών θα είναι:

```
LION
APQOLBULLATSUZ
```

'Ολες οι αλφαριθμητικές μεταβλητές είναι αρχικά κενές, έχουν μήκος μηδέν. Αν επιχειρήσετε να αντιγράψετε μία αλφαριθμητική, σε ένα τεμάχιο αλφαριθμητικής που δεν έχει αρκετό μήκος τότε η ανάθεση αυτή μπορεί να μην αναγνωριστεί από τη SuperBASIC.

Αν θέλετε να αντιγράψετε μία αλφαριθμητική σ' ένα τεμάχιο αλφαριθμητικής τότε είναι καλύτερα να βεβαιωθείτε ότι η παραλαμβάνουσα αλφαριθμητική έχει αρκετό μήκος, γεμίζοντας την πρώτα με κενά διαστήματα.

```
100 LET subject$ = "ENGLISH MATHS COMPUTING"
110 LET student$ = "                "
120 LET student$(9 TO 13) = subject$(9 TO 13)
```

Λέμε ότι το "BULL" είναι ένα τεμάχιο της αλφαριθμητικής "APQOLBULLATSUZ". Η φράση ορισμού:

```
(6 TO 9)
```

λέγεται τεμαχιστής. 'Εχει και άλλες χρήσεις. Σημειώστε ότι η ίδια σημειολογία μπορεί να χρησιμοποιηθεί και στις δύο πλευρές της δήλωσης LET. Αν θέλετε να αναφερθείτε σ' ένα μόνο χαρακτήρα θα ήταν άχαρο αν γράφατε:

```
jumble$(6 TO 6)
```

μόνο για να βγάλετε το "B" (πιθανώς σαν συμπέρασμα), έτσι μπορείτε να γράψετε:

```
jumble$(6)
```

## ΕΞΑΝΑΓΚΑΣΜΟΣ

Υποθέτουμε ότι έχετε μία μεταβλητή, τη mark$, που κρατάει ένα αρχείο βαθμών εξετάσεων. Το τεμάχιο που δίνει το βαθμό της ιστορίας μπορεί να εξαχθεί και να αυξηθεί, ίσως επειδή ο καθηγητής της ιστορίας ήταν πολύ αυστηρός στη βαθμολόγηση. Οι ακόλουθες γραμμές θα εξάγουν το βαθμό της ιστορίας:

```
100 LET mark$ = "625671"
110 LET hist$ = mark$(3 TO 4)
```

Το πρόβλημα τώρα είναι ότι η τιμή "56" της μεταβλητής hist$ είναι μία σειρά χαρακτήρων, όχι αριθμητικό δεδομένο. Αν θέλετε να το αυξήσετε, πολλαπλασιάζοντας το ας πούμε επί 1.125, τότε η τιμή της hist$ πρέπει να μετατραπεί πρώτα σε αριθμητικό δεδομένο και η SuperBASIC θα το κάνει αυτόματα αν πληκτρολογήσουμε:

```
120 LET num = 1.125 * hist$
```

Η γραμμή 120 μετατρέπει την αλφαριθμητική "56" στον αριθμό 56 και τον πολλαπλασιάζει επί 1.125 δίνοντας 63.

Τώρα πρέπει να αντικαταστήσουμε τον παλιό βαθμό με το νέο αλλά τώρα ο νέος βαθμός είναι ακόμη ο αριθμός 63 και πριν μπορέσει να εισαχθεί στην αρχική αλφαριθμητική, πρέπει να μετατραπεί στην αλφαριθμητική "63". Και πάλι η SuperBASIC θα μετατρέψει τον αριθμό αυτόματα όταν πληκτρολογήσουμε:

```
130 LET mark$(3 TO 4) = num
140 PRINT mark$
```

Η έξοδος από ολόκληρο το πρόγραμμα είναι:

```
626371
```

που δείχνει ότι ο βαθμός της ιστορίας αυξήθηκε σε 63.

Μιλώντας αυστηρά, είναι παράνομο να αναμιγνύουμε τύπους δεδομένων στις εντολής LET. θα ήταν αφελές να γράψετε:

```
LET num = "LION"
```

και θα παίρνατε ένα μήνυμα λάθους αν προσπαθούσατε, αλλά αν γράφατε:

```
LET num = "65"
```

τότε το σύστημα θα κατέληγε στο ότι θέλετε να γίνει ο αριθμός 65 η τιμή του num, και θα το έκανε αυτό. Το πλήρες πρόγραμμα είναι:

```
100 LET mark$ = "625671"
110 LET hist$ = mark$(3 TO 4)
120 LET num = 1.125 * hist$
130 LET mark$(3 TO 4) = num
140 PRINT marks
```

Και πάλι η έξοδος είναι η ίδια!

Στη γραμμή 120 μία αλφαριθμητική τιμή μετατράπηκε σε αριθμητική μορφή για να μπορέσει να πολλαπλασιαστεί στη γραμμή 130 ένας αριθμός μετατράπηκε σε αλφαριθμητική μορφή. Αυτή η μετατροπή των τύπων δεδομένων είναι γνωστή σαν εξαναγκασμός τύπων.

Μπορείτε να γράψετε το πρόγραμμα πιο οικονομικά αν κατανοήσετε και τον τεμαχισμό των αλφαριθμητικών και τον εξαναγκασμό.

```
100 LET mark$ = "625671"
110 LET mark$(3 TO 4) = 1.125 * mark$(3 TO 4)
120 PRINT mark$
```

Αν έχετε δουλέψει με άλλες BASIC θα εκτιμήσετε την απλότητα και την ισχύ του τεμαχισμού αλφαριθμητικών και του εξαναγκασμού.

**ΨΑΧΝΟΝΤΑΣ ΜΙΑ ΑΛΦΑΡΙΘΜΗΤΙΚΗ**

Μπορείτε να ψάξετε μία αλφαριθμητική για ένα δεδομένο τεμάχιο. Το ακόλουθο πρόγραμμα απεικονίζει ένα ανακάτωμα γραμμάτων και σας καλεί να βρείτε το κρυμμένο ζώο.

```
100 REM Animal Spotting
110 LET jumble$ = "SYNDICATE"
120 PRINT jumble$
130 INPUT "What is the animal?" ! an$
140 IF an$ INSTR jumble$ AND an$(1) = "C"
150    PRINT "Correct"
150 ELSE
170    PRINT "Not correct"
180 END IF
```

Ο τελεστής INSTR, επιστρέφει μηδέν αν η υπόθεση είναι λάθος. Αν η υπόθεση είναι σωστή τότε ο INSTR επιστρέφει τον αριθμό που είναι η αρχή του τεμαχίου, σ' αυτή την περίπτωση το 6.

Επειδή η έκφραση:

an$, INSTR jumble$,

μπορεί να μεταχειριστεί σαν λογική παράσταση, η θέση της αλφαριθμητικής σ' ένα επιτυχές ψάξιμο μπορεί να ληφθεί ως αληθής, ενώ ένα ανεπιτυχές ψάξιμο μπορεί να ληφθεί σαν ψευδές.

**ΑΛΛΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ**

'Εχετε ήδη συναντήσει το LEN που επιστρέφει το μήκος (τον αριθμό των χαρακτήρων) μιας αλφαριθμητικής.

'Ισως θέλετε να επαναλάβετε μία συγκεκριμένη σειρά χαρακτήρων, πολλές φορές. Για παράδειγμα αν θέλετε να εξάγετε μία σειρά αστερίσκων, τότε αντί να εισάγετε σαράντα αστερίσκους σε μία δήλωση PRINT ή να οργανώσετε ένα βρόχο, μπορείτε απλώς να πληκτρολογήσετε:

PRINT FILL$ ("*",40)

Τελικά είναι δυνατόν να χρησιμοποιήσουμε τη συνάρτηση CHR$ για να μετατρέψουμε τους κώδικες ASCII σε χαρακτήρες. Για παράδειγμα:

PRINT CHR$(65)

θα δώσει το A.

**ΣΥΓΚΡΙΝΟΝΤΑΣ ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ**

'Ενα μεγάλο μέρος του υπολογισμού ασχολείται με την οργάνωση δεδομένων έτσι ώστε να μπορούν να ψάχνονται γρήγορα. Μερικές φορές είναι απαραίτητο να τα ταξινομήσουμε με αλφαβητική σειρά. Η βάση διαφόρων διαδικασιών ταξινόμησης είναι η ευκολία να συγκρίνουμε δύο αλφαριθμητικές για να δούμε ποια έρχεται πρώτη. Επειδή τα γράμματα A, B, C... είναι εσωτερικά κωδικοποιημένα σαν 65, 66, 67... είναι φυσικό να λαμβάνουμε ως αληθείς τις ακόλουθες δηλώσεις:

- το A μικρότερο του B

- το B μικρότερο του C

και επειδή η εσωτερική σύγκριση χαρακτήρα με χαρακτήρα γίνεται αυτόματα:

- το CAT είναι μικρότερο από το DOG

- το CAN είναι μικρότερο από το CAT

Μπορείτε να γράψετε για παράδειγμα:

```
IF "CAT" < "DOG" THEN PRINT "MEOW"
```

και η έξοδος θα είναι:

```
MEOW
```

'Ομοια:

```
IF "DOG" > "CAT" THEN PRINT "WOOF"
```

θα δώσει την έξοδο:

```
WOOF
```

Χρησιμοποιήσαμε τα σύμβολα σύγκρισης των μαθηματικών για σύγκριση αλφαριθμητικών. 'Ολες οι ακόλουθες λογικές δηλώσεις - παραστάσεις είναι και επιτρεπτές και αληθείς:

```
"ALF" < "BEN"
"KIT" > "BEN"
"KIT" <= "LEN"
"KIT" >= "KIT"
"PAT" >= "LEN"
"LEN" <= "LEN"
"PAT" <> "PET"
```

Ως εδώ, οι συγκρίσεις που βασίζονταν απλώς στους εσωτερικούς κώδικες, είχαν νόημα, αλλά τα δεδομένα δεν περιορίζονται πάντα σε κεφαλαία γράμματα. θα θέλαμε για παράδειγμα:

- το CAT να είναι μικρότερο του COT

- και το K2N να είναι μικρότερο του K27N

Μία απλή σύγκριση χαρακτήρα με χαρακτήρα, βασισμένη στους εσωτερικούς κώδικες, δεν θα έδινε αυτά τα αποτελέσματα, γι' αυτό η SuperBASIC συμπεριφέρεται με πιο έξυπνο τρόπο. Το ακόλουθο πρόγραμμα με προτεινόμενη είσοδο και την έξοδο που θα προκύψει, επιδεικνύει τους κανόνες της σύγκρισης αλφαριθμητικών.

```
100 REMark comparisons
110 REPeat comp
```

```
120    INPUT "input a string" ! first$
130    INPUT "input another string" ! second$
140    IF first$ < second$ THEN PRINT "Less"
150    IF first$ > second$ THEN PRINT "Greater"
160    IF first$ = second$ THEN PRINT "Equal"
170 END REPeat comp
```

| ΕΙΣΟΔΟΣ | ΕΞΟΔΟΣ |
|---|---|
| Cat COT | Μικρότερο |
| CAT CAT | 'Ισο |
| PET PETE | Μικρότερο |
| K6 K7 | Μικρότερο |
| K66 K7 | Μεγαλύτερο |
| K12N K6N | Μεγαλύτερο |

`>` Μεγαλύτερο από - σύγκριση που εξαρτάται από την περίπτωση, οι αριθμοί συγκρίνονται αριθμητικά.

`<` Μικρότερο από - εξαρτάται από την περίπτωση, οι αριθμοί συγκρίνονται αριθμητικά.

`=` 'Ισον - εξαρτάται από την περίπτωση. Οι αλφαριθμητικές πρέπει να είναι ίδιες.

`==` Ισοδύναμο - οι αλφαριθμητικές πρέπει να είναι "σχεδόν" ίδιες. Ανεξάρτητο από την περίπτωση, οι αριθμοί συγκρίνονται αριθμητικά.

`>=` Μεγαλύτερο ή ίσο - εξαρτάται από την περίπτωση, οι αριθμοί συγκρίνονται αριθμητικά.

`<=` Μικρότερο ή ίσο - εξαρτάται από την περίπτωση, οι αριθμοί συγκρίνονται αριθμητικά.

## ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 11

1. Τοποθετήστε 12 γράμματα, όλα διαφορετικά, σε μία αλφαριθμητική μεταβλητή και άλλα έξη γράμματα σε μία δεύτερη αλφαριθμητική μεταβλητή. Ψάξτε την πρώτη αλφαριθμητική για καθένα από τα έξη γράμματα με σειρά λέγοντας σε κάθε περίπτωση αν βρέθηκε ή δε βρέθηκε.

2. Επαναλάβετε χρησιμοποιώντας πίνακες αλφαριθμητικών αντί για αλφαριθμητικές μεταβλητές. Τοποθετήστε είκοσι τυχαία γράμματα σε μία αλφαριθμητική και τυπώστε αυτά που επαναλαμβάνονται.

3. Γράψτε ένα πρόγραμμα που να διαβάζει ένα δείγμα κειμένου, ολόκληρο σε κεφαλαία. Μετρήστε τη συχνότητα κάθε γράμματος και τυπώστε τα αποτελέσματα.

> "GOVERNMENT IS A TRUST, AND THE OFFICERS OF THE GOVERNMENT ARE TRUSTEES; AND BOTH THE TRUST AND THE TRUSTEES ARE CREATED FOR THE BENEFIT OF THE PEOPLE. - HENRY CLAY, 1829."

4. Γράψτε ένα πρόγραμμα που να μετρήσει τον αριθμό των λέξεων στο ακόλουθο κείμενο. Μία λέξη αναγνωρίζεται επειδή αρχίζει με γράμμα και ακολουθείται από διάστημα, τελεία ή άλλο σημείο στίξης.

"THE REPORTS OF MY DEATH ARE GREATLY EXAGGERATED.
- CABLE FROM MARK TWAIN TO THE ASSOCIATED PRESS,
LONDON 1896."

5. Ξαναγράψτε το τελευταίο πρόγραμμα επιδεικνύοντας τη χρήση λογικών μεταβλητών και διαδικασιών.

# ΚΕΦΑΛΑΙΟ 12

# ΕΞΟΔΟΣ ΣΤΗΝ ΟΘΟΝΗ

Η SuperBASIC έχει επεκτείνει τόσο πολύ την έκταση και την ποικιλία των ευκολιών για την παρουσίαση της οθόνης ώστε περιγράφουμε τα χαρακτηριστικά σε δύο τμήματα: Απλό Τύπωμα και Οθόνη.

Το πρώτο τμήμα περιγράφει την έξοδο απλού κειμένου. Εδώ εξηγούμε τις απλούστερες τυποποιημένες μεθόδους απεικόνισης μηνυμάτων, κειμένου ή αριθμητικής εξόδου. Ακόμη και σ' αυτό το mundane τμήμα, υπάρχει καινοτομία στην ιδέα του "έξυπνου" διαστήματος - ένα παράδειγμα συνδυασμού ευκολίας χρήσης και πολύ χρήσιμων αποτελεσμάτων.

Το δεύτερο τμήμα είναι πολύ μεγαλύτερο επειδή έχει πολλά να πει. Η μεγάλη κλίμακα των χαρακτηριστικών κάνει τα πράγματα ευκολότερα. Για παράδειγμα, μπορείτε να σχεδιάσετε έναν κύκλο γράφοντας απλώς την εντολή CIRCLE ακολουθούμενη από μερικές λεπτομέρειες που ορίζουν στοιχεία όπως η θέση και το μέγεθος. Πολλά άλλα συστήματα απαιτούν να καταλάβετε γεωμετρία και τριγωνομετρία για να κάνετε κάτι, σαν έννοια, απλό.

Κάθε δεσμευμένη λέξη έχει επιλεγεί προσεκτικά για να αντανακλά το αποτέλεσμα που προκαλεί. Το WINDOW ορίζει την περιοχή της οθόνης. Το BORDER βάζει ένα περιθώριο γύρω της. Το PAPER ορίζει το χρώμα φόντου. Το INK καθορίζει το χρώμα αυτών που θα βάλετε στο χαρτί.

Αν μελετήσετε αυτό το κεφάλαιο και εξασκηθείτε λίγο θα μπορείτε εύκολα να θυμάστε ποια δεσμευμένη λέξη προκαλεί ποιο αποτέλεσμα. θα προσθέσετε αυτή τη νέα ποιότητα στον προγραμματισμό σας σχετικά εύκολα. Με την πείρα θα δείτε γιατί τα γραφικά των υπολογιστών γίνονται μια νέα μορφή τέχνης.

## ΑΠΛΟ ΤΥΠΩΜΑ

Η δεσμευμένη λέξη PRINT μπορεί να ακολουθείται από μια σειρά στοιχείων τυπώματος. 'Ενα στοιχείο τυπώματος μπορεί να είναι:

- κείμενο όπως: "This is text"
- μεταβλητές όπως: num, word$
- παραστάσεις όπως: 3 * num, weekday$ & week$

Τα στοιχεία τυπώματος μπορούν να αναμιχθούν σε οποιαδήποτε εντολή PRINT αλλά πρέπει να υπάρχουν ένας ή περισσότεροι διαχωριστές τυπώματος ανάμεσα στα διαδοχικά στοιχεία. Οι διαχωριστές τυπώματος είναι:

; Χωρίς αποτέλεσμα - απλώς χωρίζει δύο θέματα τυπώματος

! 'Εξυπνο διάστημα που φυσιολογικά εισάγει ένα διάστημα ανάμεσα στα στοιχεία που εξάγονται. Αν ένα στοιχείο δε χωράει στην τρέχουσα γραμμή τότε συμπεριφέρεται σαν σύμβολο νέας γραμμής. Αν ένα στοιχείο είναι στην αρχή της γραμμής τότε δεν παράγεται διάστημα.

, Πινακοποιητής που προκαλεί την έξοδο σε μορφή πίνακα με στήλες 8 χαρακτήρων.

\ Σύμβολο νέας γραμμής που προκαλεί μία νέα γραμμή.

TO Επιτρέπει τεμαχισμό.

**ΠΑΡΑΔΕΙΓΜΑΤΑ**

Οι αριθμοί 1,2,3 είναι νόμιμα στοιχεία τυπώματος και είναι βολικοί για την επίδειξη των αποτελεσμάτων των διαχωριστών τυπώματος.

| Εντολή | Αποτέλεσμα |
|---|---|
| `100 PRINT 1,2,3` | 1 2 3 |
| `100 print 1!2!3!` | 1 2 3 |
| `100 PRINT 1\2\3` | 1<br>2<br>3 |
| `100 PRINT 1;2;3` | 123 |
| `100 PRINT "This is text"` | This is text |
| `100 LET word$ = " "`<br>`110 PRINT word$` | moves print position |
| `100 LET num = 13`<br>`110 PRINT num` | 13 |
| `100 LET an$ = "yes"`<br>`110 PRINT "I say" ! an$` | I say yes |
| `110 PRINT "Sum is" ! 4 + 2` | Sum is 6 |

**ΑΠΛΗ ΕΜΦΑΝΙΣΗ**

Μπορείτε να τοποθετήσετε έξοδο τυπώματος οπουδήποτε στην τρέχουσα γραμμή ή οπουδήποτε στην οθόνη με μια εντολή AT.

Για παράδειγμα:

```
AT 10,15 : PRINT "This is on the 10th column at row 15"
```

Η εντολή CURSOR μπορεί να χρησιμοποιηθεί για τοποθέτηση της εξόδου τυπώματος οπουδήποτε στο πλαίσιο των 256,512 pixels. Για παράδειγμα:

```
CURSOR 100,150 : PRINT "this is 100 pixel grid units across
                        and 150 down"
```

Αν διαβάσετε τις Δεσμευμένες Λέξεις, θα δυσκολευτείτε να συμβιβάσετε το τμήμα για το PRINT με την παραπάνω περιγραφή. Δύο από τις δυσκολίες εξαφανίζονται αν κατανοήσετε ότι:

> Κείμενο εντός εισαγωγικών, μεταβλητές και αριθμοί είναι, μιλώντας αυστηρά, παραστάσεις. Είναι οι απλούστερες (εκφυλισμένες) μορφές.
>
> Οι διαχωριστές τυπώματος είναι αυστηρά ταξινομημένοι ως στοιχεία τυπώματος.

**ΟΘΟΝΗ**

Αυτό το τμήμα παρουσιάζει γενικά αποτελέσματα που συμβαίνουν είτε θέλετε να εξάγετε κείμενο είτε γραφικά. Η δήλωση:

```
MODE 8 or MODE 256
```

θα επιλέξει τη MODE 8 στην οποία υπάρχουν:

- 256 pixels κατά μήκος αριθμημένα 0-511 (δύο αριθμοί ένα pixel)
- 256 pixels κάτω αριθμημένα 0-255
- 8 χρώματα

Το pixel είναι η μικρότερη περιοχή χρώματος που μπορεί να απεικονιστεί. Χρησιμοποιούμε τον όρο σταθερό χρώμα επειδή τα χρώματα ξεκινούν από συνήθη σταθερά χρώματα, τα οποία είναι μόνο τέσσερα. 'Ομως χρησιμοποιώντας διάφορους τρόπους μπορούμε να πετύχουμε μία μεγάλη ποικιλία σκιων και αναμίξεων. Αν χρησιμοποιείτε το QL σας με κοινή τηλεόραση τότε η τηλεόρασή σας δεν θα μπορέσει να αναπαράγει αυτά τα επί πλέον αποτελέσματα.

Η εντολή:

```
MODE 4 or MODE 512
```

θα επιλέξει τη MODE 4 στην οποία υπάρχουν:

- 512 pixels κατά μήκος αριθμημένα από 0 ως 511
- 256 pixels κάτω αριθμημένα από 0 ως 255
- 4 χρώματα

## ΧΡΩΜΑΤΑ

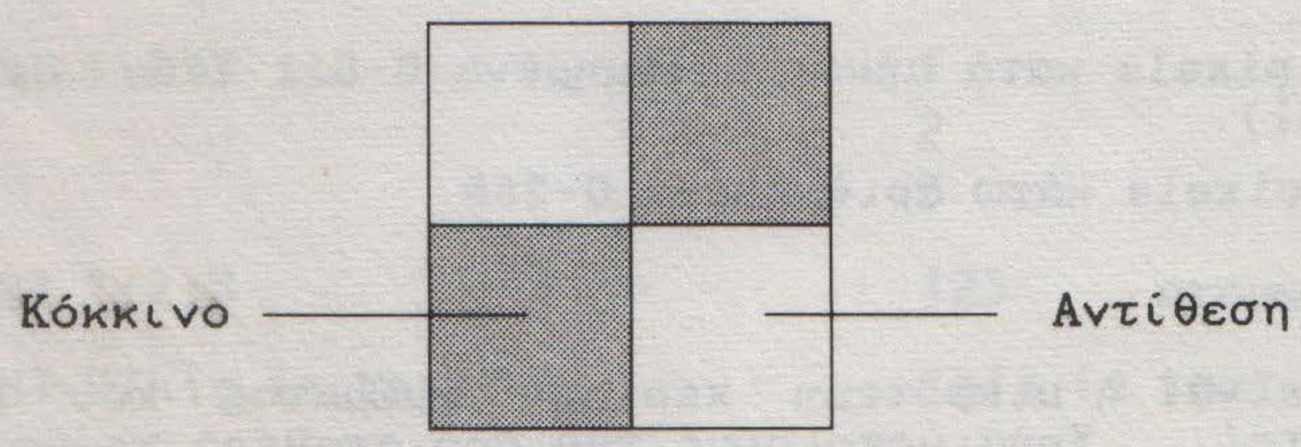
Μπορείτε να επιλέξετε ένα χρώμα χρησιμοποιώντας τον ακόλουθο κώδικα σε συνδυασμό με κατάλληλες δεσμευμένες λέξεις όπως PAPER, INK κ.λ.π. Σημειώστε ότι οι αριθμοί από μόνοι τους δε σημαίνουν τίποτα. Οι αριθμοί μεταφράζονται σε χρώματα μόνο όταν χρησιμοποιούνται με το PAPER, INK κ.λ.π.

Κώδικοί χρωμάτων

| Μορφή 8 Χρωμάτων | Κώδικας | Μορφή 4 Χρωμάτων |
|---|---|---|
| μαύρο | 0 | μαύρο |
| μπλε | 1 | μαύρο |
| κόκκινο | 2 | κόκκινο |
| μωβ | 3 | κόκκινο |
| πράσινο | 4 | πράσινο |
| γαλάζιο | 5 | πράσινο |
| κίτρινο | 6 | άσπρο |
| άσπρο | 7 | άσπρο |

Για παράδειγμα το INK 3 θα δώσει μωβ στο MODE 8.

Μπορείτε αν θέλετε να ορίσετε δύο χρώματα με μία κατάλληλη δήλωση. Για παράδειγμα 2,4 θα δώσει ένα σχήμα όπως φαίνεται. Σε κάθε ομάδα τεσσάρων pixels δύο θα είναι κόκκινα (κώδικας 2) ανάλογα με το χρώμα που επιλέχθηκε πρώτο. Τα άλλα δύο pixels θα έχουν χρώμα αντίθετο. Πραγματικά δεν είναι δυνατό να το δείτε αυτό σε κοινή τηλεόραση.




Αν γράψετε:

INK 2,4

το χρώμα της ανάμιξης θα σχηματιστεί από τους δύο κώδικες 2 και 4. θα λέμε αυτές τις επιλογές χρώμα και αντίθεση!

INK colour, contrast

## ΣΧΗΜΑ ΣΚΑΚΙΕΡΑΣ

Μπορείτε να βρείτε ποια είναι τα χρωματικά αποτελέσματα δοκιμάζοντα τα, αλλά δίνουμε περισσότερες τεχνικές λεπτομέρειες παρακάτω.

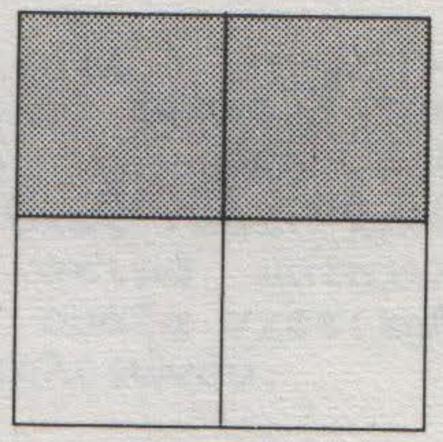
```
100 REMark Colour/Contrast
110 FOR colour = 0 TO 7 STEP 2
120   PAPER colour : CLS
130   FOR contrast = 0 TO 7 STEP 2
140     BLOCK 100,50,40,50, colour,contrast
150     PAUSE 50
160   END FOR contrast
170 END FOR colour
```

**ΑΛΛΑ ΣΧΕΔΙΑ**

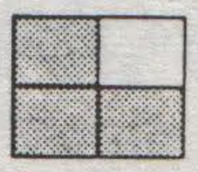
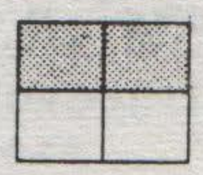
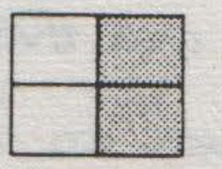
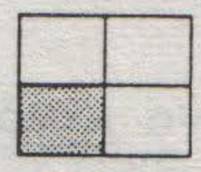
Αν θέλετε να δοκιμάσετε διαφορετικά σχέδια, μπορείτε να προσθέσετε ένα τρίτο κωδικό αριθμό στον ορισμό του χρώματος. Για παράδειγμα:

```
INK 2,4,1
```

θα ορίσει ένα κόκκινο και πράσινο οριζόντιο αποτέλεσμα. 'Ενα τετράγωνο τεσσάρων στιγμάτων θα είναι:



Τα πιθανά αποτελέσματα δείχνονται χρησιμοποιώντας κόκκινο και αντίθεση

| | Ονομα | Αποτέλεσμα |
|---|---|---|
| 0 | 'Ενα pixel αντίθεσης |  |
| 1 | Οριζόντιο σχέδιο |  |
| 2 | κατακόρυφο σχέδιο |  |
| 3 | σκακιέρα |  |

**ΧΡΩΜΑΤΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ**

Μπορείτε να ορίσετε ένα χρωματικό αποτέλεσμα όπως περιγράφηκε παραπάνω χρησιμοποιώντας τρεις αριθμούς. Για παράδειγμα:

INK χρώμα,αντίθεση,σχέδιο

μπορούν να χρησιμοποιηθούν με:

- το χρώμα στην κλίμακα από 0 ως 7
- η αντίθεση στην κλίμακα από 0 ως 7
- το σχέδιο από 0 ως 3

Μπορείτε να φτάσετε στο ίδιο αποτέλεσμα με ένα μόνο αριθμό αν θέλετε, όμως δεν είναι τόσο εύκολο να τον κατασκευάσετε. Δέστε το 'Εννοιες.

Το ακόλουθο πρόγραμμα θα απεικονίσει όλα τα δυνατά χρωματικά αποτελέσματα.

```
100 REMARK Colour Effects
110 FOR num = 0 TO 255
120   BLOCK 100, 50, 40,50, num
130   PAUSE 50
140 END FOR num
```

**PAPER**

Το PAPER ακολουθούμενο από ένα, δύο ή τρεις αριθμούς καθορίζει το φόντο. Για παράδειγμα:

PAPER 2 {κόκκινο}

PAPER 2,4 {κόκκινη πράσινη σκακιέρα}

PAPER 2,4,1 {κόκκινο πράσινο οριζόντιο σχέδιο}

Το χρώμα δεν θα είναι ορατό μέχρι να γίνει κάτι άλλο, για παράδειγμα, να καθαριστεί η οθόνη.

**INK**

Το INK ακολουθούμενο από ένα, δύο ή τρεις αριθμούς καθορίζει το χρώμα τυπώματος των χαρακτήρων, των γραμμών ή των άλλων γραφικών. Το χρώμα και τα αποτελέσματα είναι τα ίδια όπως στο INK. Για παράδειγμα:

INK {κόκκινο μελάνι}

INK 2,4. {κόκκινο πράσινο μελάνι σε σχήμα σκακιέρας)

INK 2,4,1 {κόκκινο πράσινο μελάνι σε οριζόντιο σχέδιο}

Το χρώμα θα αλλάξει για κάθε επόμενη έξοδο.

**CLS**

Το CLS σημαίνει καθάρισε το παράθυρο στο τρέχον χρώμα φόντου - όπως ένας δάσκαλος καθαρίζει τον πίνακα, μόνο που αυτός είναι ηλεκτρονικός και πολύχρωμος.

**ΑΝΑΒΟΣΒΗΣΜΑ**

Μπορείτε να κάνετε το χρώμα μελάνης να αναβοσβήνει στη mode 8 μόνο. Για να ξεκινήσει το αναβόσβησμα πληκτρολογήστε:

FLASH 1

και για να σταματήσει:

FLASH 0

Αλληλεπικαλύπτοντας χαρακτήρες που αναβοσβήνουν μπορεί να δώσει αποτελέσματα συναγερμού.

**ΑΡΧΕΙΑ**

θα έχετε χρησιμοποιήσει microdrives για αποθήκευση προγραμμάτων και θα έχετε χρησιμοποιήσει τις εντολές LOAD και SAVE. Οι μικροκασέτες μπορούν να χρησιμοποιηθούν για αποθήκευση δεδομένων όπως και προγραμμάτων. Η λέξη αρχείο συνήθως σημαίνει μια σειρά καταχωρήσεων δεδομένων, όπου καταχώρηση είναι μια ομάδα σχετικών πληροφοριών όπως όνομα, διεύθυνση και αριθμός τηλεφώνου.

Δύο από τους πιο ευρέως χρησιμοποιούμενους τύπους αρχείων είναι τα σειριακά και τα άμεσης προσπέλασης. Τα θέματα σ' ένα σειριακό αρχείο, συνήθως διαβάζονται με σειρά αρχίζοντας από το πρώτο. Αν θέλετε την πεντηκοστή καταχώρηση τότε πρέπει να διαβάσετε τις πρώτες σαρανταεννιά για να τη βρείτε. Από την άλλη μεριά, η πεντηκοστή καταχώρηση σ' ένα αρχείο άμεσης προσπέλασης, μπορεί να βρεθεί γρήγορα γιατί το σύστημα δε χρειάζεται να κοιτάξει τις προηγούμενες καταχωρήσεις για να τη βρει. Η μουσική ποπ σε μία κασέτα είναι όπως ένα σειριακό αρχείο αλλά οκτώ κομμάτια σ' ένα δίσκο σχηματίζουν ένα αρχείο άμεσης προσπέλασης. Μπορείτε να βάλετε το βραχίονα του πικάπ σε οποιοδήποτε από τα οκτώ κομμάτια.

Ο απλούστερος δυνατός τύπος αρχείου είναι μία σειρά αριθμών. Για να επειδείξουμε την ιδέα θα τοποθετήσουμε τους αριθμούς 1 ως 100 σ' ένα αρχείο με όνομα numbers. 'Ομως το πλήρες όνομα του αρχείου αποτελείται από δύο μέρη.

- όνομα συσκευής

- προσαρτημένες πληροφορίες

Υποθέτουμε ότι θέλουμε να δημιουργήσουμε το αρχείο numbers, σε μικροκασέτα στο microdrive 1. Το όνομα της συσκευής είναι:

MDV1_

και οι προσαρτημένες πληροφορίες είναι απλώς το όνομα του αρχείου:

numbers

'Ετσι το ολοκληρωμένο όνομα του αρχείου είναι:

MDV1_numbers

**ΚΑΝΑΛΙΑ**

'Ενα πρόγραμμα μπορεί να χρησιμοποιεί πολλά αρχεία ταυτόχρονα, αλλά είναι πιο βολικό να αναφερόμαστε σ' ένα αρχείο μ' ένα συνδεμένο αριθμό καναλιού. Αυτός μπορεί να είναι οποιοσδήποτε ακέραιος στην κλίμακα 0 εως 15. 'Ενα αρχείο συνδέεται μ' έναν αριθμό καναλιού, χρησιμοποιώντας τη δήλωση OPEN ή αν είναι καινούργιο αρχείο, την OPEN-NEW. Για παράδειγμα μπορεί να διαλέξετε το κανάλι 7 για το αρχείο numbers και να γράψετε:

```
OPEN_NEW #7 ,mdv1_numbers
    ¦       ¦     ¦    ¦____ όνομα αρχείου
    ¦       ¦     ¦_________ συσκευή
    ¦       ¦_______________ αριθμός καναλιού
    ¦_______________________ δεσμευμένη λέξη
```

Τώρα μπορείτε να αναφέρεστε στο αρχείο απλώς εισάγοντας τον αριθμό #7. Το πλήρες πρόγραμμα είναι:

```
100  REMark Simple file
110  OPEN_NEW #7, MDV1_numbers
120  FOR number = 1 to 100
130 PRINT #7, number
140  END FOR number
150  CLOSE #7
```

Η εντολή PRINT έχει αποτέλεσμα το "τύπωμα" των αριθμών στο αρχείο της μικροκασέτας επειδή το 7 έχει συνδεθεί μ' αυτό. Η δήλωση CLOSE #7 είναι απαραίτητη επειδή το σύστημα κάνει κάποια εσωτερική δουλειά όταν το αρχείο έχει χρησιμοποιηθεί. Επίσης ελευθερώνει το κανάλι 7 για άλλες πιθανές χρήσεις. Αφού το πρόγραμμα έχει εκτελεστεί, πληκτρολογήστε:

DIR MDV1_

και ο κατάλογος θα δείξει ότι ο αρχείο numbers υπάρχει στη μικροκασέτα του microdrive mdv1_.

Επίσης χρειάζεται να ξέρετε ότι το αρχείο είναι σωστό και μπορείτε να είστε σίγουροι γι' αυτό μόνο αν το αρχείο διαβαστεί και ελεχθεί. Η απαραίτητη δεσμευμένη λέξη είναι OPEN_IN, κατά τα άλλα το πρόγραμμα ανάγνωσης δεδομένων από αρχείο είναι όμοιο με το προηγούμενο.

```
100 REMark Reading a file
110 OPEN_IN #6, MDV1_numbers
120 FOR item = 1 TO 100
130   INPUT #6, number
140   PRINT ! number !
150 END FOR item
160 CLOSE #6
```

Το πρόγραμμα θα εξάγει τους αριθμούς 1 έως 100, αλλά μόνο αν η μικροκασέτα που περιέχει το αρχείο numbers βρίσκεται ακόμη στο microdrive mdv1_.

## ΣΥΣΚΕΥΕΣ ΚΑΙ ΚΑΝΑΛΙΑ

'Εχετε δει ένα παράδειγμα συσκευής, ένα αρχείο δεδομένων σε microdrive. Μπορεί να λέμε συνήθως ότι ένα αρχείο ανοίγεται αλλά αυστηρά, εννοούμε ότι η συσκευή συνδέθηκε μ' ένα συγκεκριμένο κανάλι. 'Ολες οι περαιτέρω πληροφορίες έχουν επίσης δοθεί. Ορισμένες συσκευές έχουν κανάλια συνδεμένα μόνιμα μαζί τους από το σύστημα.

| κανάλι | χρήση |
|---|---|
| #0 | ΕΞΟΔΟΣ - παράθυρο διαταγών<br>ΕΙΣΟΔΟΣ - πληκτρολόγιο |
| #1 | ΕΞΟΔΟΣ - παράθυρο τυπώματος |
| #2 | ΛΙΣΤΑ - έξοδος λίστας |

Μπορείτε να δημιουργήσετε ένα παράθυρο οποιουδήποτε μεγέθους, οπουδήποτε στην οθόνη. Το όνομα της συσκευής για το παράθυρο είναι:

SCR_

## ΠΑΡΑΘΥΡΑ

```
360 150a80x40
 ¦   ¦  ¦  ¦_____________ τιμή κάτω
 ¦   ¦  ¦________________ τιμή κατά μήκος
 ¦   ¦___________________ ύψος
 ¦_______________________ πλάτος
```

Το ακόλουθο πρόγραμμα δημιουργεί ένα παράθυρο στον αριθμό καναλιού 5 και το γεμίζει με χρώμα πράσινο (κώδικας 4) και μετά το κλείνει:

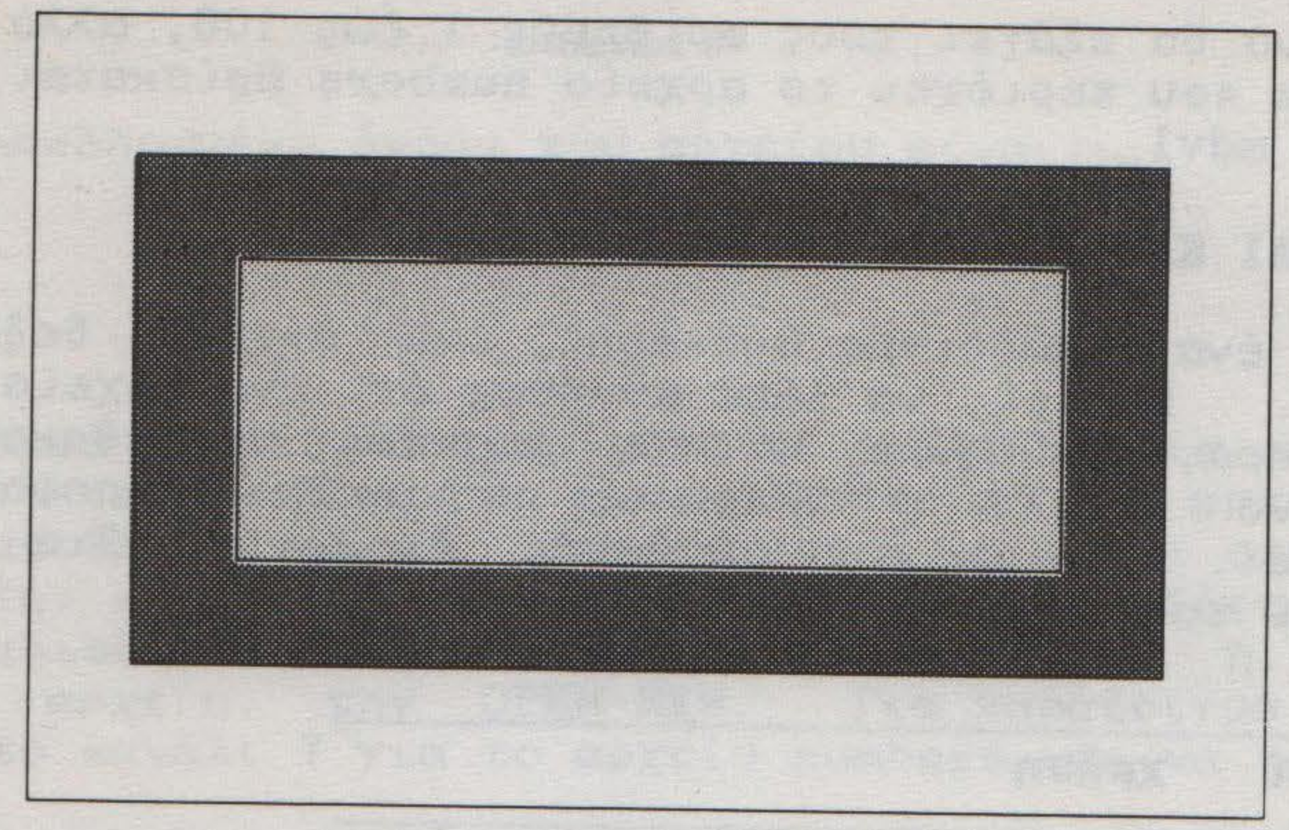
```
100 REMark Create a window
110 OPEN #5, SCR_360x150a80x40
120 PAPER #5,4 : CLS #5
130 CLOSE #5
```

Σημειώστε ότι κάθε παράθυρο έχει τα δικά του χαρακτηριστικά όπως χρώμα φόντου, μελάνης κ.τ.λ. Το γεγονός ότι ένα παράθυρο έχει ανοιχθεί δε σημαίνει ότι είναι το τρέχον παράθυρο παράλειψης. (default)

Μπορείτε να αλλάξετε τη θέση ή το σχήμα ενός ανοιγμένου παραθύρου χωρίς να το κλείσετε και να το ξαναανοίξετε. Δοκιμάστε να προσθέσετε δύο γραμμές στο προηγούμενο πρόγραμμα:

```
124 WINDOW #5,300,110,100,65
126 PAPER #5,2 : CLS #5
```

Ξανατρέξτε το πρόγραμμα και θα βρείτε ένα κόκκινο παράθυρο μέσα στο αρχικό πράσινο. Αυτό το κόκκινο παράθυρο είναι συνδεμένο τώρα με το κανάλι 5 όπως στο σχήμα:



**BORDER (ΠΕΡΙΘΩΡΙΟ)**

Μπορείτε να βάλετε ένα περιθώριο γύρω από την άκρη της οθόνης ή ενός παράθυρου. Για παράδειγμα:

BORDER #5,6

θα δημιουργήσει ένα περιθώριο γύρω από το παράθυρο του καναλιού #5. θα έχει πλάτος 6 μονάδες και το μέγεθος του παραθύρου θα μειωθεί ανάλογα. Το περιθώριο θα είναι διαφανές, προστατεύοντας ότι ήταν από κάτω. Μπορείτε να ορίσετε ένα χρωματιστό περιθώριο με τη συνήθη μέθοδο:

BORDER #5,6,2

θα παράγει ένα κόκκινο περιθώριο. Μπορείτε να φτιάξετε περιθώρια άλλων χρωμάτων και αναμίξεων με τις συνήθεις μεθόδους. Για παράδειγμα:

BORDER 10

θα προσθέσει ένα διαφανές περιθώριο πλάτους 10 pixels στο τρέχον παράθυρο (διαφανές επειδή δεν ορίστηκε χρώμα) και

BORDER 2,0,7,0

θα προσθέσει ένα περιθώριο ασπρόμαυρο με σχέδιο πλάτους 2 pixels.

**BLOCK**

Μπορείτε να ορίσετε το μέγεθος, τη θέση και το χρώμα ενός τετράπλευρου με μία μόνο εντολή. Τοποθετείται στο σύστημα συντεταγμένων που αναφέρεται στο τρέχον παράθυρο ή στην οθόνη. Για παράδειγμα:

BLOCK #5,10,20,50,100,2

θα δημιουργήσει ένα τετράπλευρο στο παράθυρο #5 στη θέση 50

μονάδες κατά μήκος και 100 μονάδες κάτω. Θα έχει 10 μονάδες πλάτος και 20 μονάδες ύψος. Το χρώμα του θα είναι κόκκινο.

Αξίζει να σημειώσουμε ότι οι εντολές WINDOW και BLOCK δουλεύουν χωρίς αλλαγή στις μορφές 8 και 4 χρωμάτων (παρόλο που τα χρώματα μπορεί να διαφέρουν) επειδή οι τιμές κατά μήκος είναι πάντα από 0 έως 511 και υπάρχουν πάντα 256 θέσεις pixels προς τα κάτω.

**ΕΙΔΙΚΟ ΤΥΠΩΜΑ**

**CSIZE**

Μπορείτε να αλλάξετε το μέγεθος των χαρακτήρων. Για παράδειγμα:

CSIZE 3,1

θα δώσει τους μεγαλύτερους δυνατούς χαρακτήρες και:

CSIZE 0,0

θα δώσει τους μικρότερους. Ο πρώτος αριθμός πρέπει να είναι 0,1,2, ή 3 και καθορίζει το πλάτος. Ο δεύτερος πρέπει να είναι 0 ή 1 και καθορίζει το ύψος. Τα καινούρια μεγέθη είναι:

MODE 4 CSIZE 0,0 25 γραμμές των 84 χαρακτήρων

MODE 8 CSIZE 1,0 25 γραμμές των 42 χαρακτήρων

Ο αριθμός των σειρών και των στηλών που είναι διαθέσιμες για κάθε μέγεθο χαρακτήρων εξαρτάται από το αν χρησιμοποιούμε μονιτορ ή τηλεόραση. Τα μεγέθη σειρών και στηλών που δόθηκαν είναι για μόνιτορ. Αυτά σε τηλεόραση θα είναι μικρότερα και θα διαφέρουν ανάμεσα σε διάφορες τηλεοράσεις.

Αν χρησιμοποιείτε τη μορφή χαμηλής διακριτικότητας, το QL δεν θα σας επιτρέψει να διαλέξετε μέγεθος χαρακτήρων μικρότερο από το μέγεθος παράλειψης.

**STRIP**

Μπορείτε να δώσετε ένα ειδικό φόντο στους χαρακτήρες για να τους κάνετε να ξεχωρίζουν. Για παράδειγμα:

STRIP 7

θα δώσει μια λευκή ταινία ενώ:

STRIP 2,4,2

θα δώσει μία κόκκινη/πράσινη ταινία με κατακόρυφο σχέδιο. 'Ολοι οι φυσιολογικοί συνδυασμοί χρωμάτων είναι δυνατοί.

**OVER**

Το φυσιολογικό τύπωμα γίνεται στο τρέχον χρώμα φόντου. Μπορείτε να το αλλάξετε αυτό, χρησιμοποιώντας το STRIP.

Μπορείτε να παράγετε κι άλλα αποτελέσματα χρησιμοποιώντας:

OVER 1 το 1 τυπώνει με μελάνι πάνω σε διάφανη ταινία

OVER -1 το -1 τυπώνει με μελάνι πάνω στην απεικόνιση που ήδη υπάρχει στην οθόνη

Για να επιστρέψετε στο κανονικό τύπωμα στο τρέχον STRIP χρησιμοποιήστε:

OVER 0

**UNDER**

Μπορείτε να υπογραμμίσετε χαρακτήρες.

UNDER 1 υπογραμμίζει οποιαδήποτε ακόλουθη έξοδο με το τρέχον μελάνι

UNDER 0 σταματάει την υπογράμμιση.

**ΓΡΑΦΙΚΑ ΜΕ ΚΛΙΜΑΚΑ**

Αν θέλετε να σχεδιάσετε φυσιολογικά γεωμετρικά σχήματα σε μία οθόνη TV ή Video, δεν μπορείτε να το κάνετε εύκολα με το σύστημα που βασίζεται στα pixels. Το σύστημα θα κάνει την απαραίτητη δουλειά ώστε να μπορείτε σχετικά εύκολα να σχεδιάσετε κανονικούς κύκλους, τετράγωνα και άλλα σχήματα.

Η κλίμακα παράλειψης του συστήματος γραφικών συντεταγμένων είναι 100 στην κατακόρυφη διεύθυνση και όσο χρειάζεται στην κατά μήκος διεύθυνση ώστε τα σχήματα που σχεδιάζονται με τις ειδικές γραφικές δεσμευμένες λέξεις (PLOT, DRAW, CIRCLE) να είναι κανονικά.

Η αρχή των γραφικών δεν είναι η ίδια με την αρχή των pixels που χρησιμοποιείται για τον ορισμό της θέσης παραθύρων και τετραπλεύρων. Η αρχή των γραφικών βρίσκεται στην κάτω αριστερή γωνιά της τρέχουσας οθόνης ή παράθυρου.

**ΣΗΜΕΙΑ ΚΑΙ ΓΡΑΜΜΕΣ**

Είναι εύκολο να σχεδιάσετε σημεία και γραμμές χρησιμοποιώντας γραφικά με κλίμακα. Χρησιμοποιώντας κατακόρυφη κλίμακα 100, μπορείτε να να σχεδιάσετε ένα σημείο κοντά στο κέντρο παραθύρου με:

POINT 60,50

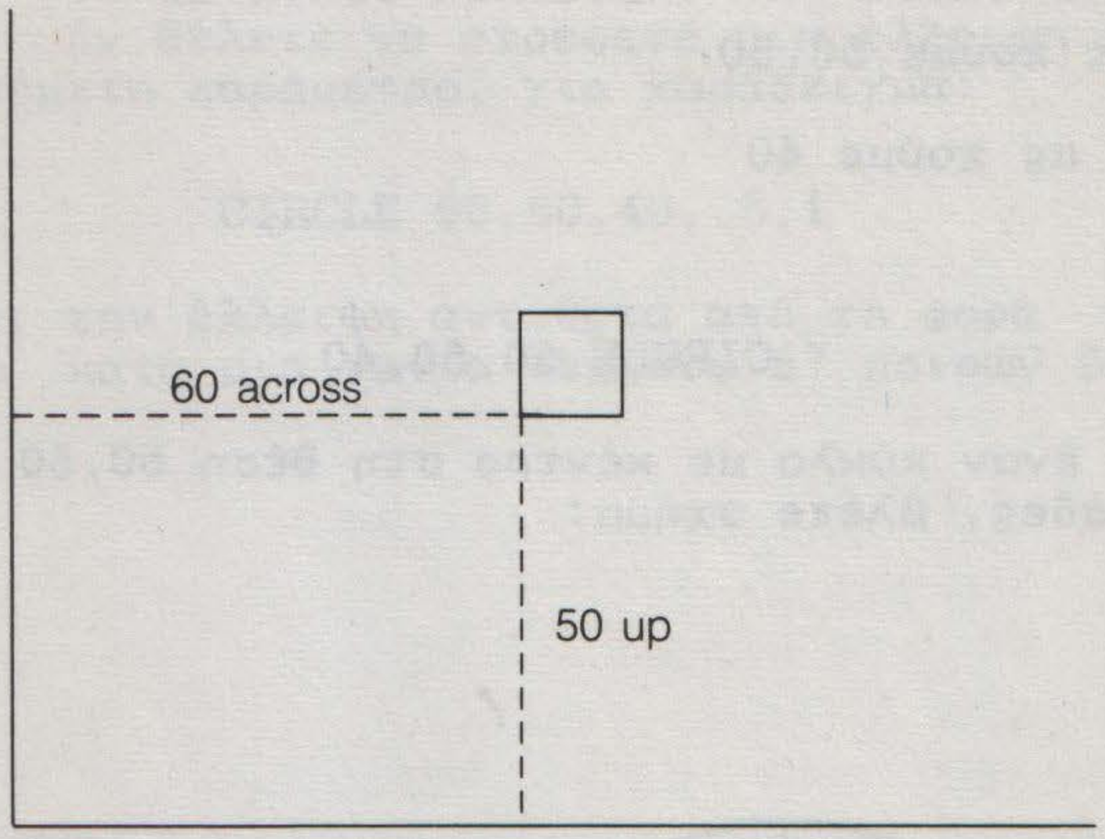
Το σημείο (60 μονάδες κατά μήκος και 50 μονάδες πάνω) θα σχεδιαστεί στο τρέχον χρώμα μελάνης.

'Ομοια μπορεί να σχεδιαστεί μια γραμμή με τη εντολή:

LINE 60,50 TO 80,90

Περισσότερα στοιχεία μπορούν να προστεθούν. Για παράδειγμα το ακόλουθο θα σχεδιάσει ένα τετράγωνο:

```
LINE 60,50 TO 70,50 TO 70,60 TO 60,60 TO 60,50
```




**ΣΧΕΤΙΚΗ ΜΟΡΦΗ**

Δύο συντεταγμένες όπως,

κατά μήκος, πάνω

φυσιολογικά ορίζουν ένα σημείο σε σχέση με την αρχή 0,0 της κάτω αριστερής γωνίας ενός παραθύρου (ή όπου αλλού διαλέξετε). Είναι μερικές φορές πιο βολικό να ορίζουμε σημεία σε σχέση με την τρέχουσα θέση του δείκτη. Για παράδειγμα το παραπάνω τετράγωνο μπορεί να σχεδιαστεί και με άλλον τρόπο χρησιμοποιώντας τη εντολή LINE_R που σημαίνει:

"Κάνε όλες τις συντεταγμένες σχετικές με την τρέχουσα θέση δείκτη."

```
POINT 60,50
LINE_R 0,0 TO 10,0 TO 0,10 TO -10,0 TO 0, - 10
```

Κατ' αρχήν το σημείο 60,50 γίνεται η αρχή και μετά καθώς οι γραμμές σχεδιάζονται, το τέλος μιας γραμμής γίνεται η αρχή για την επόμενη.

Το ακόλουθο πρόγραμμα θα κάνει ένα σχέδιο από τυχαία τοποθετημένα χρωματιστά τετράγωνα.

```
100 REMark Coloured Squares
110 PAPER 7 : CLS
120 FOR sq = 1 TO 100
130   INK RND(1 TO 6)
140   POINT RND(90), RND(90)
150   LINE_R 0,0 TO 10,0 TO 0,10 TO - 10,0 TO 0, -10
160 END FOR sq
```

Τα ίδια αποτελέσματα μπορούσαμε να πάρουμε και με απόλυτα γραφικά αλλά θα απαιτούσε λίγη προσπάθεια παραπάνω.

**ΚΥΚΛΟΙ ΚΑΙ ΕΛΛΕΙΨΕΙΣ**

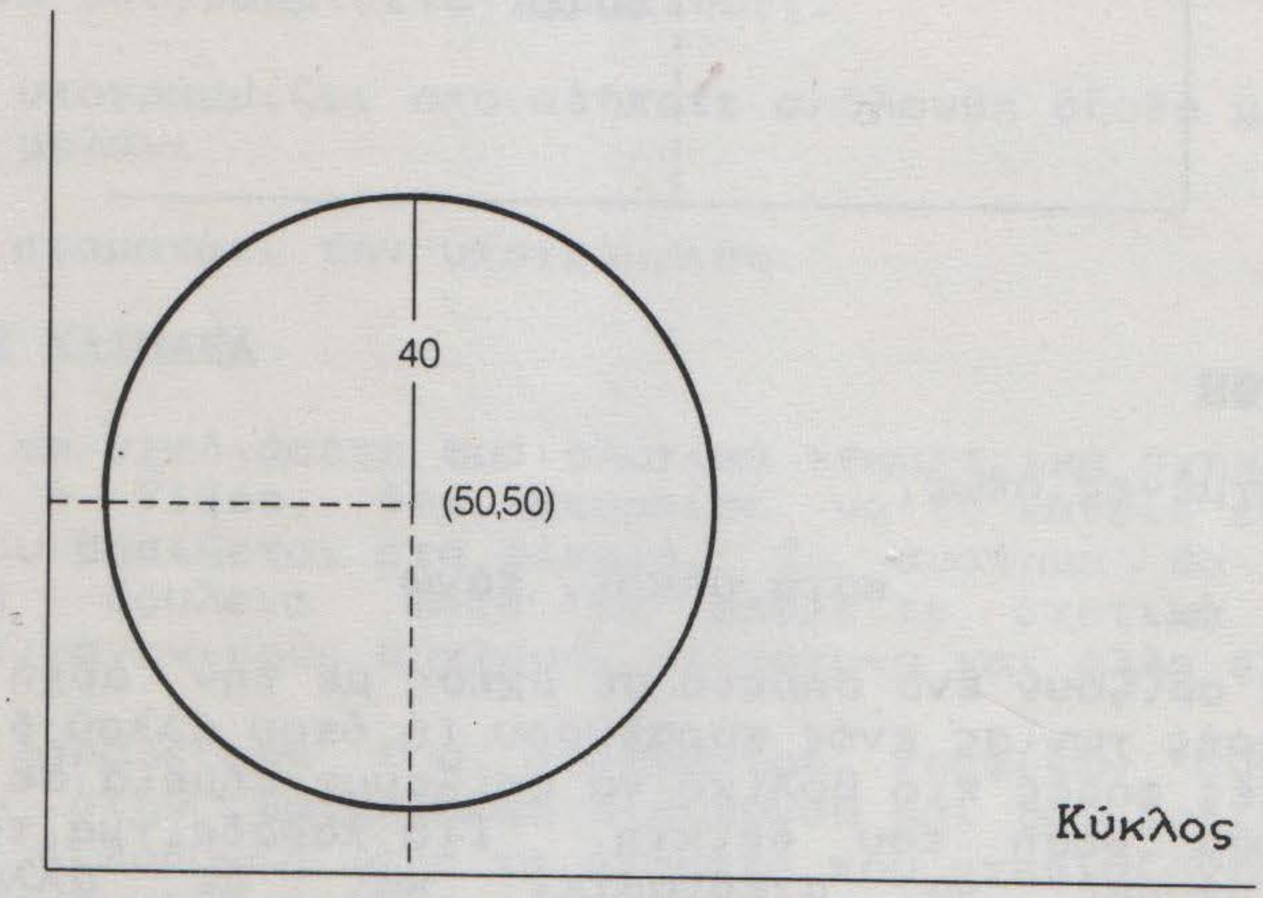
Αν θέλετε να σχεδιάσετε έναν κύκλο πρέπει να ορίσετε:

- θέση, ας πούμε 50,50
- ακτίνα, ας πούμε 40

Η εντολή

```
CIRCLE 50,50,40
```

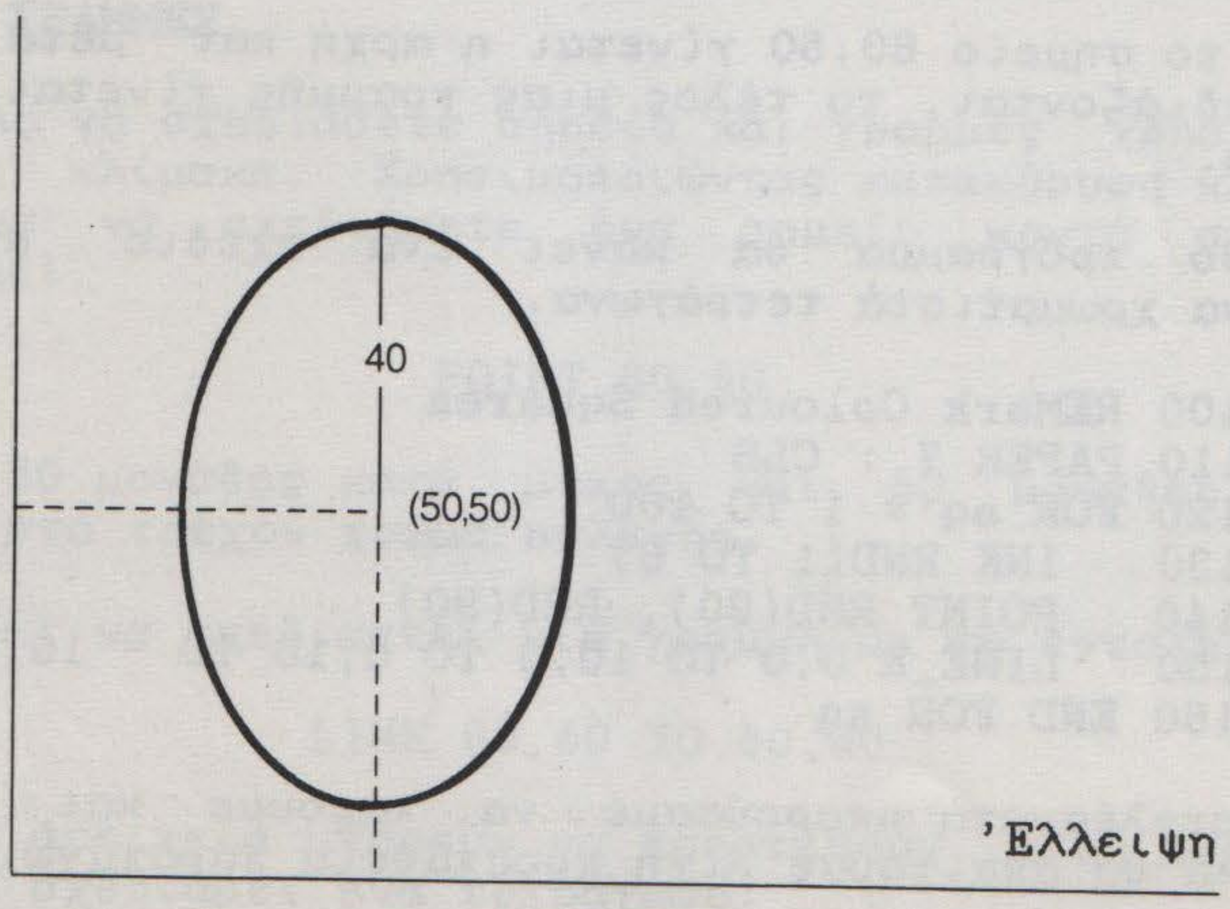
θα σχεδιάσει έναν κύκλο με κέντρο στη θέση 50,50 και ακτίνα (ή ύψος) 40 μονάδες, βλέπε σχήμα:




Αν προσθέσετε δύο παραμέτρους:

```
CIRCLE 50,50,40,.5,0
```

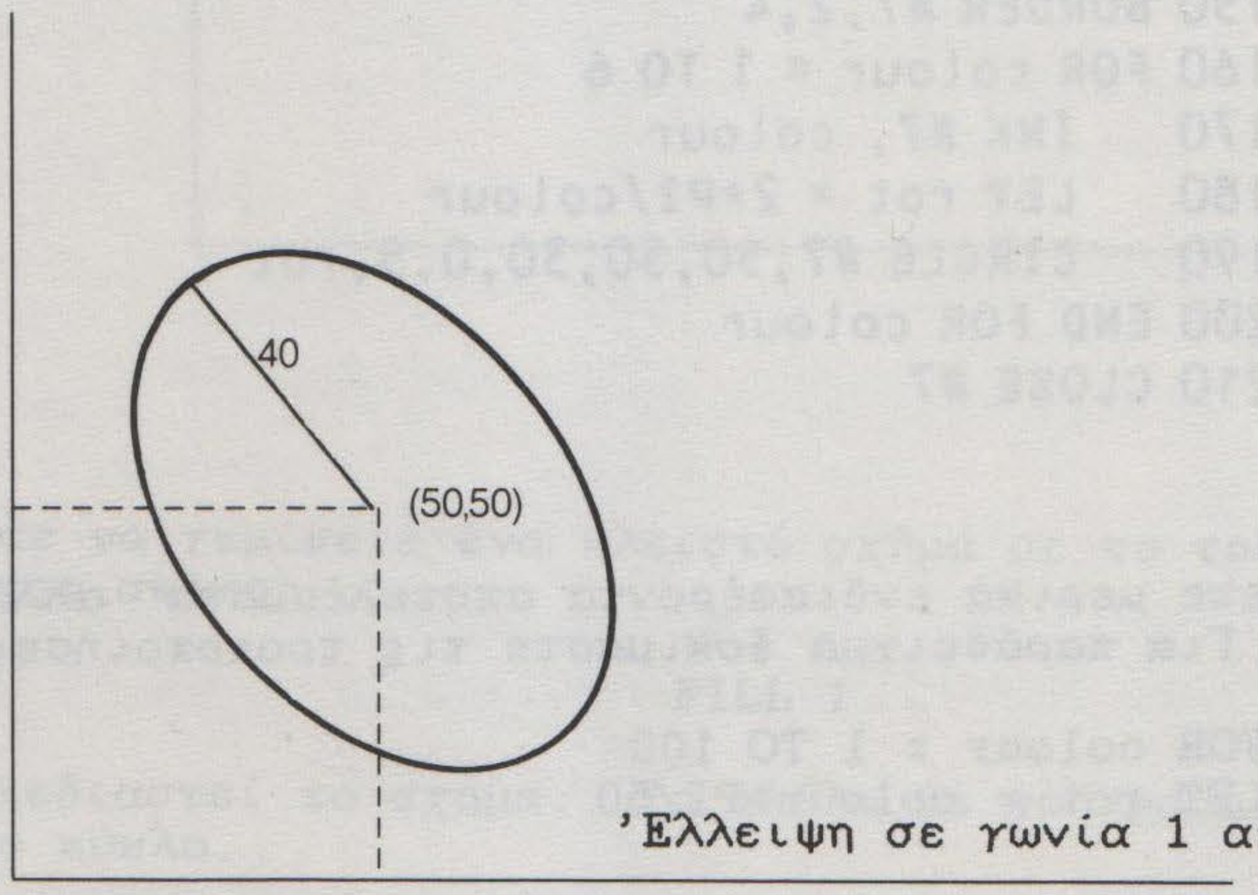
θα πάρετε μια έλλειψη.



Το ύψος της έλλειψης είναι 40 όπως πριν αλλά η οριζόντια "ακτίνα" είναι τώρα μόνο 0.5 του ύψους. Ο αριθμός 0.5 λέγεται εκκεντρότητα. Αν η εκκεντρότητα είναι 1 τότε παίρνετε κύκλο, αν δεν είναι 1 παίρνετε έλλειψη. Αν είναι μηδέν παίρνετε ευθεία γραμμή. Αν θέλετε να στρέψετε μία έλλειψη μπορείτε να αλλάξετε την πέμπτη παράμετρο, για παράδειγμα:

```
CIRCLE 50,50,40,.5,1
```

Αυτό θα στρέψει την έλλειψη αντίθετα από τη φορά των δεικτών του ωρολογιού κατά μια γωνία περίπου 57 μοιρών όπως φαίνεται στο σχήμα:




'Ελλειψη σε γωνία 1 ακτινίου

Μια ευθεία γωνία είναι 180 μοίρες ή PI, (π), ακτίνια κι έτσι μπορείτε να κάνετε ένα σχέδιο από ελλείψεις με το πρόγραμμα:

```
100 FOR rot = 0 TO 2*PI STEP PI/6
110   CIRCLE 50,50,40,0.5,rot
120 END FOR rot
```

Η σειρά των παραμέτρων για έναν κύκλο ή έλλειψη είναι:

κέντρο_κατά_μήκος, κέντρο_πάνω, ύψος, [εκκεντρότητα], [γωνία]

Οι τελευταίες δύο παράμετροι είναι προαιρετικές και αυτό δείχνεται επειδή βρίσκονται μέσα σε αγκύλες ([]).

**ΠΑΡΑΔΕΙΓΜΑΤΑ**

Γράψτε ένα πρόγραμμα που να κάνει τα ακόλουθα:

[1] Να ανοίγει ένα παράθυρο (στο 350,180).

[2] Να βάζει κλίμακα 100 στη μορφή 8.

[3] Να επιλέγει μαύρο χρώμα φόντου και να καθαρίζει το παράθυρο.

[4] Να φτιάχνει πράσινο περιθώριο πλάτους 2 μονάδων.

[5] Να κάνει ένα σχέδιο έξη χρωματιστών ελλείψεων.

[6] Να κλείνει το παράθυρο.

```
100 REMark pattern
110 MODE 8
120 OPEN #7, scr_100x100a 100x50
130 SCALE #7,100,0,0
140 PAPER #7,0 : CLS #7
150 BORDER #7,2,4
160 FOR colour = 1 TO 6
170    INK #7, colour
180    LET rot = 2*PI/colour
190    CIRCLE #7,50,50,30,0.5,rot
200 END FOR colour
210 CLOSE #7
```

Μπορείτε να έχετε μερικά ενδιαφέροντα αποτελέσματα αλλάζοντας το πρόγραμμα. Για παράδειγμα δοκιμάστε τις τροποποιήσεις:

```
160 FOR colour = 1 TO 100
180 LET rot = colour*PI/50
```

**ΤΟΞΑ**

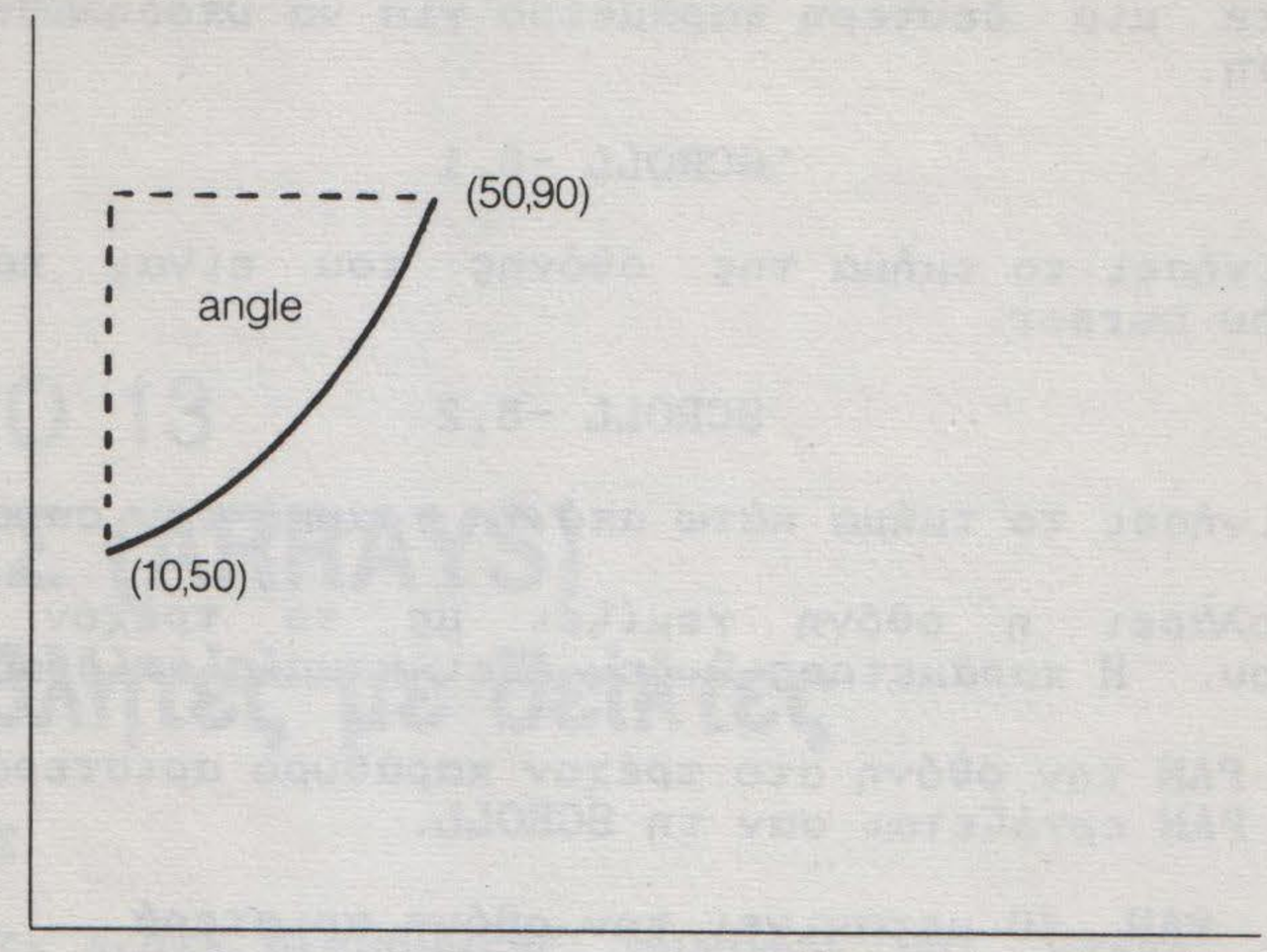
Αν θέλετε να σχεδιάσετε ένα τόξο πρέπει να αποφασίσετε:

- το αρχικό σημείο
- το τελικό σημείο
- την καμπυλότητα

Τα πρώτα δύο θέματα είναι απλά αλλά το ποσό καμπυλότητας δεν είναι τόσο εύκολο. Μπορείτε να το κάνετε σχεδιάζοντας με ακρίβεια ή δοκιμάζοντας αλλά πρέπει να αποφασίσετε ποια γωνία βλέπει το τόξο και μετά να ορίσετε τη γωνία σε ακτίνια. Μια γωνία 1.5 ακτινίων θα δώσει μικρό λύγισμα και μια μικρή γωνία θα δώσει μεγάλη καμπυλότητα. Δοκιμάστε για παράδειγμα:

```
ARC 10,50 TO 50,90, 1
```

που δίνει μια γωνία καμπυλότητας στο τρέχον χρώμα μελάνης.




**FILL**

Μπορείτε να γεμίσετε ένα κλειστό σχήμα με το τρέχον χρώμα INK γράφοντας απλώς:

```
FILL 1
```

πριν σχεδιαστεί το σχήμα. Το ακόλουθο πρόγραμμα παράγει έναν πράσινο κύκλο.

```
INK 4
FILL 1
CIRCLE 50,50,30
```

Η εντολή FILL δουλεύει σχεδιάζοντας διαδοχικές οριζόντιες γραμμές ανάμεσα στα κατάλληλα σημεία.

Η εντολή:

```
FILL 0
```

θα σταματήσει το αποτέλεσμα του FILL.

**SCROLL (ΡΟΛΑΡΙΣΜΑ) ΚΑΙ PAN (ΠΛΑΓΙΑ ΚΙΝΗΣΗ)**

Μπορείτε να κάνετε SCROLL και PAN στην απεικόνιση ενός παραθύρου όπως ένας κάμερμαν. Κανονίζετε το SCROLL με όρους pixels. 'Ενας θετικός αριθμός δείχνει SCROLL προς τα πάνω, έτσι:

```
SCROLL 10
```

μετακινεί το κείμενο στο τρέχον παράθυρο επάνω κατά 10 pixels.

```
SCROLL -8
```

Μετακινεί την οθόνη επάνω κατά 8 pixels. Μπορείτε να προσθέσετε μια δεύτερη παράμετρο για να υποδηλωθεί τμηματική μετακίνηση.

```
SCROLL -8,1
```

θα μετακινήσει το τμήμα της οθόνης που είναι πάνω από τη γραμμή του cursor.

```
SCROLL -8,2
```

θα μετακινήσει το τμήμα κάτω από τη γραμμή του cursor.

Καθώς ρολάρει η οθόνη γεμίζει με το τρέχον χρώμα του περιθωρίου. Η παράμετρος 0 δεν έχει καμμία επίδραση.

Μπορείτε PAN την οθόνη στο τρέχον παράθυρο αριστερά ή δεξιά. Η εντολή PAN εργάζεται σαν τη SCROLL.

```
PAN  40 μετακινεί την οθόνη αριστερά
PAN -40 μετακινεί την οθόνη δεξιά
```

Μια δεύτερη παράμετρος καθορίζει τμηματική PAN.

0 ολόκληρη οθόνη

3 ολόκληρη γραμμή στην οποία είναι ο cursor

4 η δεξιά πλευρά της γραμμής που βρίσκεται ο cursor.

**ΠΡΟΒΛΗΜΑΤΑ ΣΤΟ ΚΕΦΑΛΑΙΟ 12**

[1] Γράψτε ένα πρόγραμμα που να ζωγραφίζει ένα πλέγμα από 10 σειρές με δέκα τετράγωνα.

[2] Βάλτε τους αριθμούς από το 1 έως το εκατό στο κάτω αριστερά τετράγωνο και βάλτε F (τέλος) στο τελευταίο τετράγωνο.

[3] Ζωγραφίστε ένα στόχο στην οθόνη. θα αποτελείται από ένα εξωτερικό κύκλο στον οποίο θα υπάρχουν οι αριθμοί.

# ΚΕΦΑΛΑΙΟ 13

# ΠΙΝΑΚΕΣ (ARRAYS)
# = Μεταβλητές με δείκτες

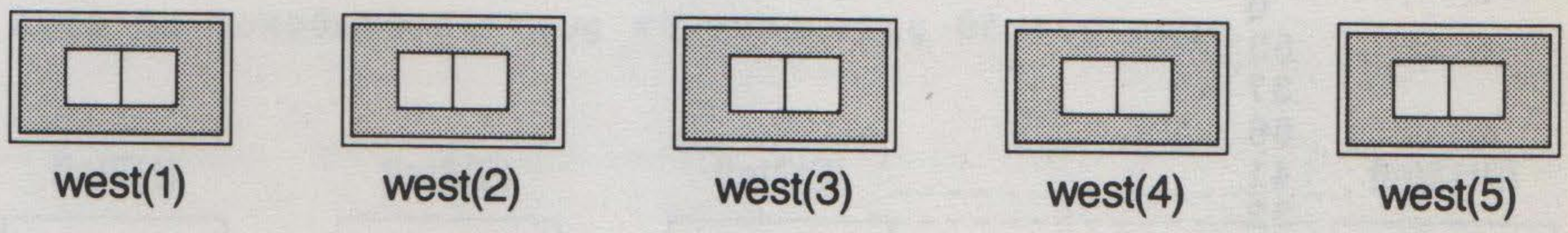
**ΓΙΑΤΙ ΠΙΝΑΚΕΣ**

Υποθέτουμε ότι είστε διευθυντής φυλακών και έχετε ένα νέο τμήμα στη φυλακή που λέγεται West Block. Είναι έτοιμο να δεχτεί 5 νέους κρατούμενους. Χρειάζεται να ξέρετε ποιος κρατούμενος (γνωστός από τον αριθμό του) βρίσκεται σε κάθε κελί. θα μπορούσατε να δώσετε σε κάθε κελί ένα όνομα αλλά είναι απλούστερο να τους δώσετε αριθμούς από 1 έως 5.

Σε μια υπολογιστική προσομοίωση θα φανταστούμε 5 κρατούμενους με αριθμούς που μπορούμε να τους βάλουμε σε μία εντολή DATA.

```
DATA 50, 37, 86, 41, 32
```

Δημιουργούμε έναν πίνακα μεταβλητών που μοιράζονται το όνομα west και διαχωρίζονται από έναν προσαρτημένο αριθμό σε παρένθεση:




Είναι απαραίτητο να ορίζουμε έναν πίνακα δίνοντας τις διαστάσεις του με μια εντολή DIM.

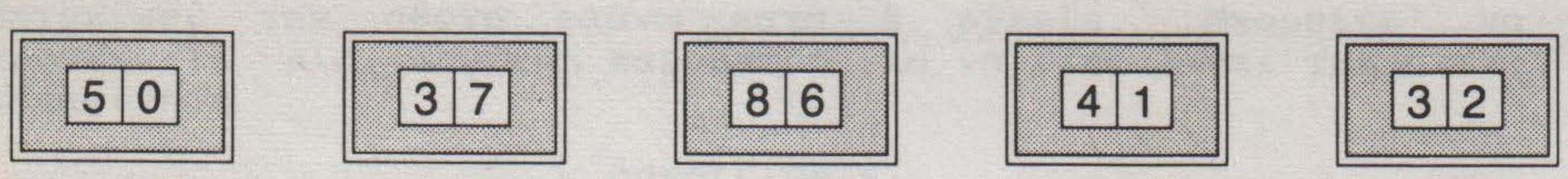
```
DIM west(5)
```

Αυτό κάνει τη SuperBASIC να κατανείμει χώρο, ο οποίος μπορεί να είναι μεγάλος. Αφού εκτελεστεί η εντολή DIM, οι πέντε μεταβλητές μπορούν να χρησιμοποιηθούν.

Οι κατάδικοι μπορούν να διαβαστούν με READ από τις εντολές DATA στις πέντε μεταβλητές του πίνακα.

```
FOR cell = 1 TO 5 : READ west(cell)
```

Μπορούμε να προσθέσουμε άλλον ένα βρόχο FOR με εντολή PRINT για να αποδείξουμε ότι οι κατάδικοι βρίσκονται στα κελλιά:




Το ολοκληρωμένο πρόγραμμα φαίνεται παρακάτω:

```
100 REMark Prisoners
110 DIM west(5)
120 FOR cell = 1 TO 5 : READ west(cell)
130 FOR cell = 1 TO 5 : PRINT cell! west(cell)
140 REMark End of Program
150 DATA 50, 37, 86, 41, 32
```

Η έξοδος από το πρόγραμμα είναι:

```
1 50
2 37
3 86
4 41
5 32
```

Οι αριθμοί 1 ως 5 λέγονται δείκτες του ονόματος πίνακα west. Ο πίνακας west είναι ένας αριθμητικός πίνακας που αποτελείται από πέντε αριθμητικά στοιχεία του πίνακα.

Μπορείτε να αντικαταστήσετε τη γραμμή 130 με:

```
130 PRINT west
```

Αυτό θα εξάγει μόνο τις τιμές:

```
 0
50
37
86
41
32
```

Το μηδέν στην κορυφή της λίστας εμφανίζεται επειδή οι δείκτες αρχίζουν από μηδέν ως το δηλωμένο αριθμό. Θα δείξουμε αργότερα πόσο χρήσιμα είναι τα μηδενικά στοιχεία στους πίνακες.

Σημειώστε επίσης ότι όταν ένας αριθμητικός πίνακας δηλώνεται, τα στοιχεία του έχουν όλα την τιμή μηδέν.

## ΑΛΦΑΡΙΘΜΗΤΙΚΟΙ ΠΙΝΑΚΕΣ

Οι αλφαριθμητικοί πίνακες είναι όμοιοι με τους αριθμητικούς εκτός από μια επί πλέον διάσταση στη εντολή DIM που ορίζει το μήκος κάθε αλφαριθμητικής μεταβλητής στον πίνακα. Υποθέστε ότι δέκα από τους κορυφαίους παίκτες στο Royal Birkdale για το Βρετανικό πρωτάθλημα γκολφ 1982 μπορούν να δηλωθούν με τα μικρά τους ονόματα και να τοποθετηθούν σε εντολές DATA.

```
DATA "Tom", "Graham", "Sevvy", "Jack", "Lee"
DATA "Nick", "Bernard", "Ben", "Gregg", "Hal"
```

θα χρειαστείτε δέκα διαφορετικά ονόματα μεταβλητών, αλλά αν υπήρχαν εκατό ή χίλιοι παίκτες τότε η δουλειά θα γινόταν υπερβολικά κουραστική. Ο πίνακας είναι μία ομάδα μεταβλητών σχεδιασμένη για να ξεπερνά προβλήματα αυτού του είδους. Κάθε όνομα μεταβλητής αποτελείται από δύο μέρη:

- ένα όνομα σύμφωνο με τους συνήθης κανόνες
- ένα αριθμητικό μέρος που λέγεται δείκτης

Γράψτε τα ονόματα των μεταβλητών έτσι:

```
flat$(1), flat$(2), flat$(3).... κ.λ.π.
```

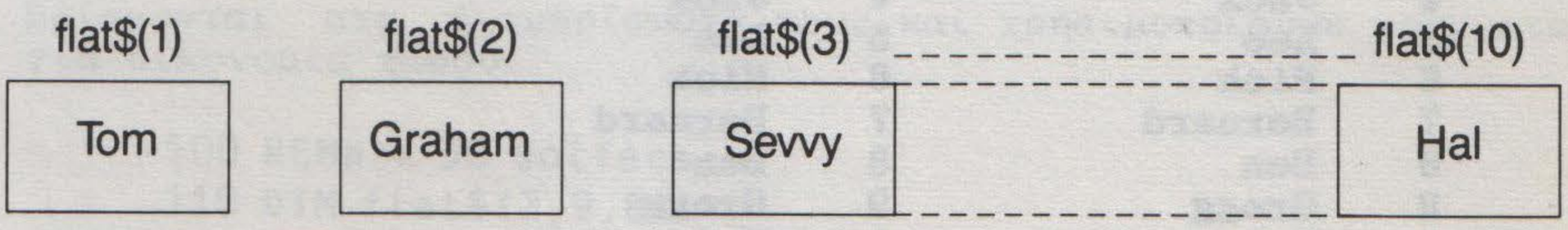
Πριν χρησιμοποιήσετε τις μεταβλητές του πίνακα, πρέπει να πείτε στο σύστημα σχετικά με τον πίνακα και τις διαστάσεις του:

```
DIM flat$(10,8)
```

Αυτό έχει αποτέλεσμα να κρατηθούν δέκα μεταβλητές για χρήση στο πρόγραμμα. Κάθε αλφαριθμητική μεταβλητή στον πίνακα μπορεί να έχει μέχρι οκτώ χαρακτήρες. Οι εντολές DIM συνήθως πρέπει να τοποθετούνται όλες μαζί κοντά στην αρχή του προγράμματος. Αφού ο πίνακας έχει δηλωθεί με τη εντολή DIM, όλα τα στοιχεία του μπορούν να χρησιμοποιηθούν. 'Ενα σημαντικό πλεονέκτημα είναι ότι μπορείτε να δώσετε το αριθμητικό μέρος (το δείκτη) σαν μία αριθμητική μεταβλητή. Μπορείτε να γράψετε:

```
FOR number = 1 TO 10 : READ flat$(number)
```

Αυτό θα τοποθετήσει τους πίνακες στις θέσεις τους.




Μπορείτε να αναφέρεστε στη μεταβλητή με το συνήθη τρόπο αλλά να θυμάστε να χρησιμοποιείτε το σωστό δείκτη. Υποθέτουμε ότι ο Tom και ο Sevvy θέλουν να ανταλλάξουν θέσεις. Σε υπολογιστικούς όρους, ένας απ' αυτούς, ας πούμε ο Tom, πρέπει να παέι σε μια προσωρινή θέση για να δώσει χρόνο στον Sevvy να μετακινηθεί. Μπορείτε να γράψετε

```
LET temp$ = flat$(1)       Tom into temporary
LET flat$(1) = flat$(3)    Sevvy into flat$(1)
LET flat$(3) = temp$       Tom into flat$(3)
```

Το ακόλουθο πρόγραμμα τοποθετεί τους δέκα παίκτες σ' έναν πίνακα με όνομα flat$ και τα ονόματα των παικτών μαζί με τον "αριθμό θέσης" τους (τους δείκτες του πίνακα) για να δείξει ότι είναι τοποθετημένοι. Μετά οι παίκτες των θέσεων 1 και 3, αλλάζουν θέσεις. Η λίστα των παικτών ξανατυπώνεται για να δείξει ότι έγινε η ανταλλαγή.

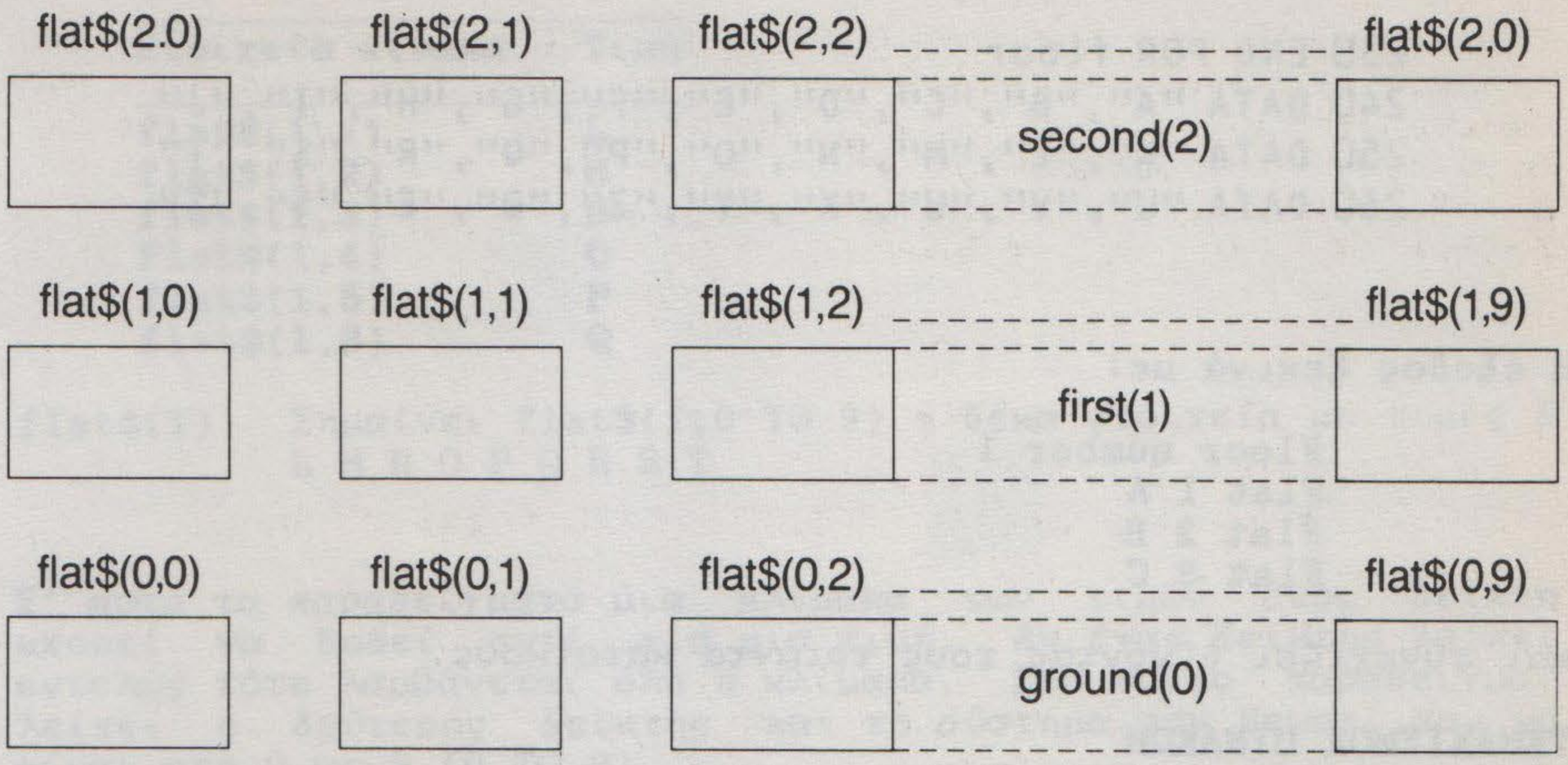
```
100 REMark Golfers' Flats
110 DIM flat$(10,8)
120 FOR number = 1 TO 10 : READ flat$(number)
130 printlist
140 exchange
150 printlist
160 REMark End of main program
170 DEFine PROCedure printlist
180    FOR num = 1 TO 10 : PRINT num, flat$(num)
190 END DEFine
200 DEFine PROCedure exchange
210    LET temp$ = flat$(1)
220    LET flat$(1) = flat$(3)
230    LET flat$(3) = temp$
240 END DEFine
250 DATA "Tom", "Graham", "Sevvy", "Jack", "Lee"
260 DATA "Nick", "Bernard", "Ben", "Greg", "Hal"
```

| Εξοδος (γραμμή 130) | | Εξοδος (γραμμή 150) | |
|---|---|---|---|
| 1 | Tom | 1 | Sevvy |
| 2 | Graham | 2 | Graham |
| 3 | Sevvy | 3 | Tom |
| 4 | Jack | 4 | Jack |
| 5 | Lee | 5 | Lee |
| 6 | Nick | 6 | Nick |
| 7 | Bernard | 7 | Bernard |
| 8 | Ben | 8 | Ben |
| 9 | Gregg | 9 | Gregg |
| 10 | Hal | 10 | Hal |

## ΠΙΝΑΚΕΣ ΔΥΟ ΔΙΑΣΤΑΣΕΩΝ

Μερικές φορές η φύση ενός προβλήματος συνιστά δύο διαστάσεις όπως 3 όροφοι των 10 διαμερισμάτων παρά μία απλή σειρά των 30.

Υποθέτουμε ότι 20 ή περισσότεροι παίκτες του γκολφ χρειάζονται διαμερίσματα και υπάρχει μία πολυκατοικία 30 διαμερισμάτων χωρισμένων σε 3 ορόφους των 10 διαμερισμάτων. Μια ρεαλιστική μέθοδος αναπαράστασης της πρλυκατοικίας θα ήταν με ένα διδιάστατο πίνακα. Μπορείτε να φανταστείτε τις τριάντα μεταβλητές όπως φαίνονται παρακάτω:




Υποθέτουμε ότι υπάρχουν εντολές DATA με 30 ονόματα, ένας κατάλληλος τρόπος να τοποθετήσουμε τα ονόματα στα διαμερίσματα θα ήταν:

```
120 FOR floor = 0 TO 2
130    FOR num = 0 TO 9
140       READ flat$(floor, num)
150    END FOR num
160 END FOR floor
```

Χρειάζεστε επίσης μία εντολή DIM.

```
110 DIM flat$(2,9,8)
```

που δείχνει ότι ο πρώτος δείκτης μπορεί να είναι από 0 ως 2 (αριθμός ορόφου) και ο δεύτερος δείκτης μπορεί να είναι από 0 ως 9 (αριθμός δωματίου). Ο τρίτος αριθμός δηλώνει το μέγιστο αριθμό χαρακτήρων σε κάθε στοιχείο του πίνακα. Προσθέτουμε μια ρουτίνα τυπώματος για να δείξουμε ότι οι παίκτες βρίσκονται στα διαμερίσματα τους και χρησιμοποιούμε γράμματα για οικονομία χώρου.

```
100 REMark 30 Golfers
110 DIM flat$(2,9,8)
120 FOR floor = 0 TO 2
130    FOR num = 0 TO 9
140       READ flat$(floor,num) : REMark Golfer goes in
150    END FOR num
160 END FOR floor
170 REMark End of input
180 FOR floor = 0 TO 2
190    PRINT "Floor number" ! floor
200    FOR num = 0 TO 9
210       PRINT 'Flat' ! num ! flat$(floor,num)
220    END FOR num
```

```
230 END FOR floor
240 DATA "A","B","C","D","E","F","G","H","I","J"
250 DATA "K","L","M","N","O","P","Q","R","S","T"
260 DATA "U","V","W","X","Y","Z","@","£","$","%"
```

Η έξοδος ξεκινά με:

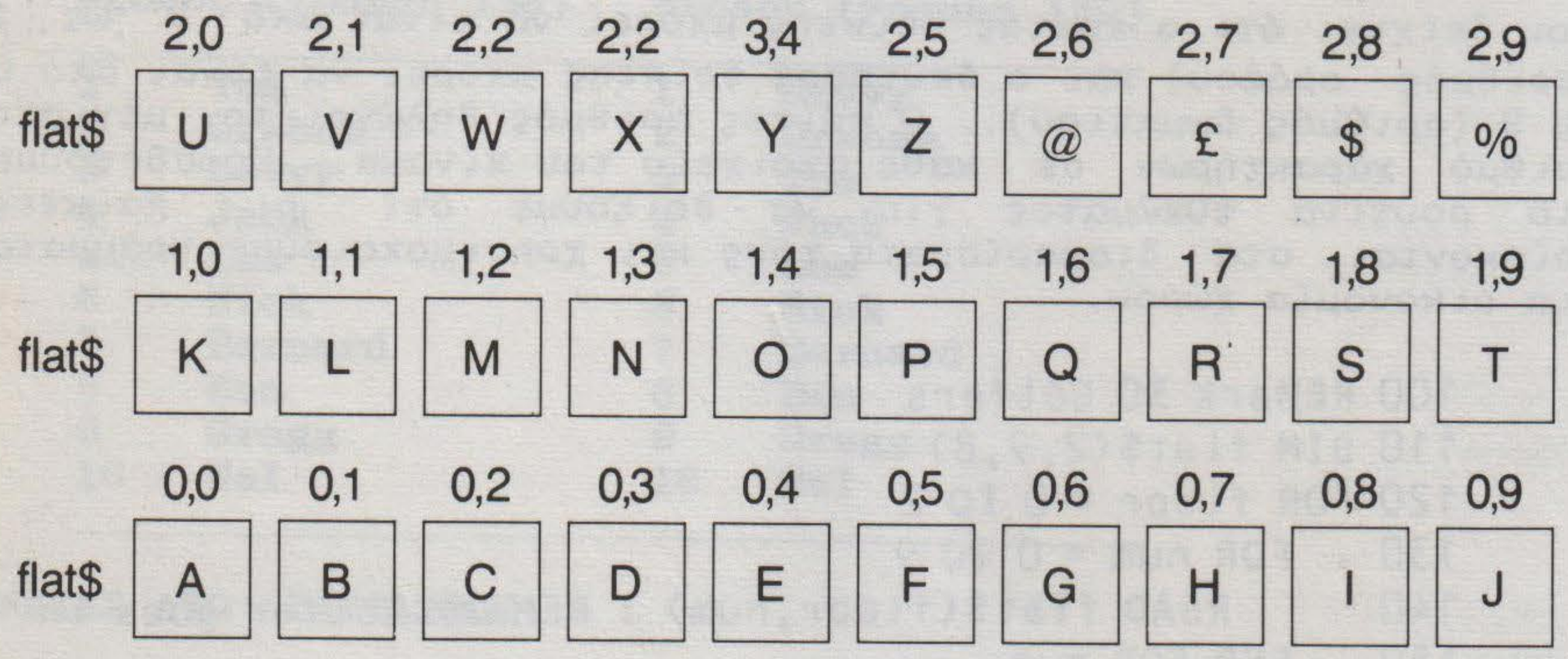
```
Floor number 1
Flat 1 A
Flat 2 B
Flat 3 C
```

και συνεχίζει δίνοντας τους τριάντα κατοίκους.

**ΤΕΜΑΧΙΣΜΟΣ ΠΙΝΑΚΩΝ**

'Ισως βρείτε αυτό το τμήμα δύσκολο, παρόλο που ουσιαστικά είναι η ίδια ιδέα με τον τεμαχισμό αλφαριθμητικών. θα σας χρειαστεί ο τεμαχισμός αλφαριθμητικών αν προχωρήσετε πέρα από το στάδιο εκμάθησης του προγραμματισμού. Η ανάγκη για τεμαχισμό πινάκων είναι πολύ σπανιότερη και ίσως προτιμάτε να παραλείψετε αυτό το τμήμα, ιδίως στην πρώτη ανάγνωση.

Τώρα απλοποιούμε το πρόβλημα των παικτών στα διαμερίσματα για να επιδείξουμε την ιδέα του τεμαχισμού πίνακα. Τα διαμερίσματα θα είναι αριθμημένα από 0 ως 9 για να είναι μονοψήφια και τα ονόματα θα είναι απλοί χαρακτήρες για οικονομία χώρου.




'Εχοντας δεδομένες τις παραπάνω τιμές, τα ακόλουθα είναι τεμάχια πίνακα.

| | |
|---|---|
| flat$(1,3) | σημαίνει ένα απλό στοιχείο πίνακα με τιμή N |
| flat$(1,1 TO 6) | σημαίνει έξι στοιχεία με τιμές L M N O P Q |

| Στοιχείο πίνακα | Τιμή |
|---|---|
| flat$(1,1) | L |
| flat$(1,2) | M |
| flat$(1,3) | N |
| flat$(1,4) | O |
| flat$(1,5) | P |
| flat$(1,6) | Q |

flat$(1) Σημαίνει flat$(1,0 TO 9) - δέκα στοιχεία με τιμές K L M N O P Q R S T

Σ' αυτά τα παραδείγματα μια κλίμακα των τιμών ενός δείκτη μπορεί να δοθεί αντί για μια τιμή. Αν ένας δείκτης λείπει εντελώς τότε λαμβάνεται όλη η κλίμακα. Στο τρίτο παράδειγμα, λείπει ο δεύτερος δείκτης και το σύστημα τον θεωρεί σαν να είναι από 0 ως 9 (0 TO 9).

Οι τεχνικές του τεμαχισμού πίνακα και του τεμαχισμού αλφαριθμητικής είναι όμοιες αν και η τελευταία βρίσκει πιο πλατιές εφαρμογές.

## ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 13

1. ΤΑΞΙΝΟΜΗΣΗ

Τοποθετήστε δέκα αριθμούς σ' έναν πίνακα διαβάζοντας τους από μια εντολή DATA. Ψάξτε τον πίνακα να βρείτε τον μικρότερο αριθμό. Κάντε αυτόν τον μικρότερο αριθμό, την τιμή του πρώτου στοιχείου ενός νέου πίνακα. Αντικαταστήστε τον στο πρώτο πίνακα μ' έναν πολύ μεγάλο αριθμό. Επαναλάβετε τη διαδικασία κάνοντας το δεύτερο μικρότερο αριθμό, τη δεύτερη τιμή στο νέο πίνακα και συνεχίστε μέχρι να έχετε έναν ταξινομημένο πίνακα από αριθμούς που μετά θα τους τυπώσετε.

2. ΦΙΔΑΚΙΑ ΚΑΙ ΣΚΑΛΕΣ

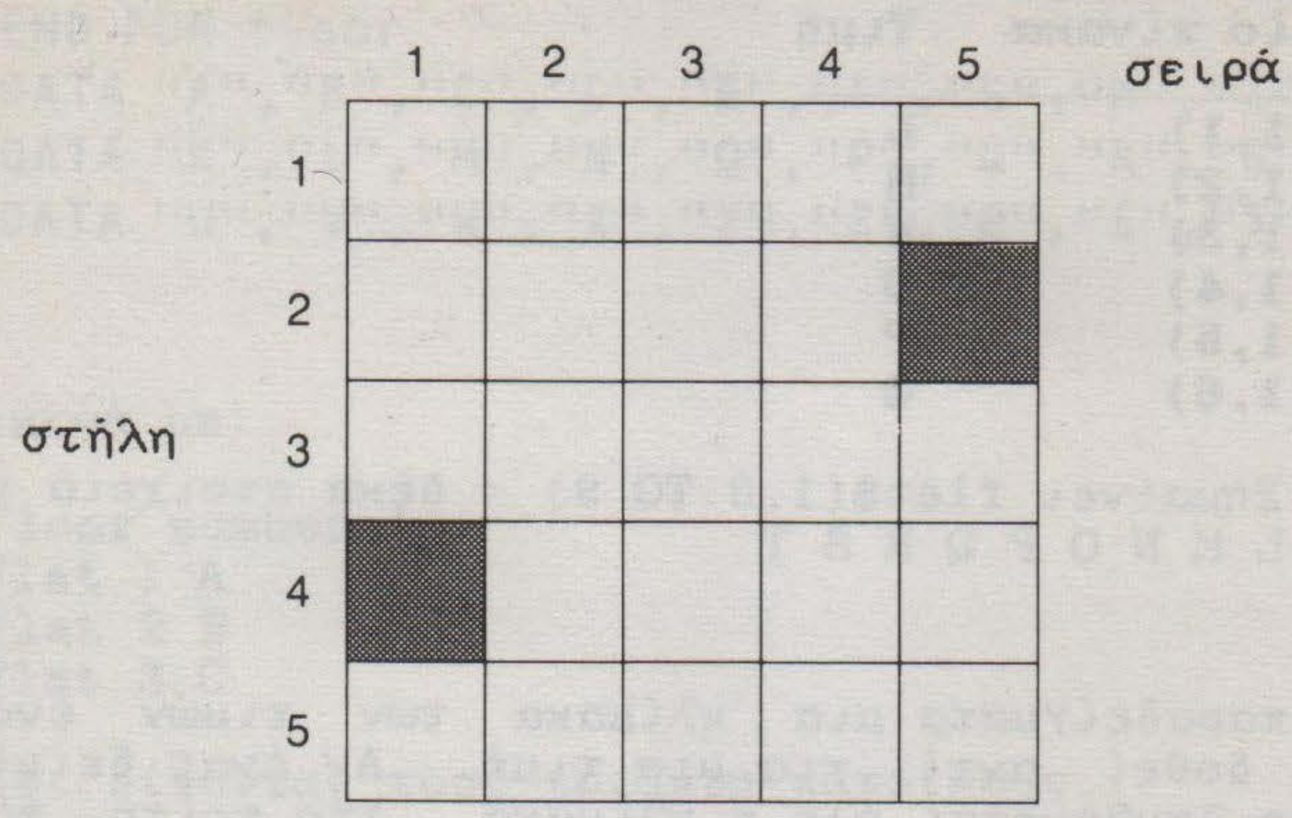
Παρατηρήστε ένα παιχνίδι "φιδάκια και σκάλες" με έναν αριθμητικό πίνακα 100 στοιχείων. Κάθε στοιχείο πρέπει να περιέχει:

μηδέν

ή έναν αριθμό από 10 ως 90 που σημαίνει ότι ο παίκτης πρέπει να μεταφερθεί σ' αυτόν τον αριθμό είτε "ανεβαίνοντας μια σκάλα", είτε "κατεβαίνοντας ένα φίδι".

ή τα ψηφία 1,2,3 κ.τ.λ. που δηλώνουν μια συγκεκριμένη θέση ενός παίκτη. Δημιουργήστε έξι φίδια και έξι σκάλες τοποθετώντας αριθμούς στον πίνακα και προσομοιώστε ένα "σόλο" τρέξιμο από έναν παίκτη για να δοκιμάσετε το παιχνίδι.

3. ΚΕΝΑ ΣΤΑΥΡΟΛΕΞΑ




Τα σταυρόλεξα συνήθως έχουν περιττό αριθμό σειρών και στηλών στις οποίες τα μαύρα τετράγωνα έχουν συμμετρικό σχήμα. Το σχέδιο λέμε ότι έχει περιστροφική συμμετρία επειδή η περιστροφή 180 μοιρών δεν το αλλάζει.
Σημειώστε ότι μετά από περιστροφή 180 μοιρών το τετράγωνο στη σειρά 4, στήλη 1 θα γίνει το τετράγωνο στη σειρά 2 στήλη 5. Αυτό σημαίνει ότι η σειρά 4, στήλη 1 γίνεται σειρά 6-4, στήλη 6-1 σ' ένα πλαίσιο 5χ5.

Γράψτε ένα πρόγραμμα που να δημιουργεί και να απεικονίζει συμμετρικά σχέδια τέτοιου είδους.

4. Τροποποιήστε το σχέδιο του σταυρόλεξου ώστε να μην υπάρχουν σειρές λιγότερων από τέσσερα άσπρα τετράγωνα κατά μήκος ή κάτω.

5. ΑΝΑΚΑΤΕΜΑ ΤΡΑΠΟΥΛΟΧΑΡΤΩΝ

Τα τραπουλόχαρτα δηλώνονται με τους αριθμούς 1-52 αποθηκευμένους σ' έναν πίνακα. Μπορούν να μετατραπούν εύκολα σε πραγματικές τιμές τραπουλόχαρτων όταν χρειαστεί. Τα χαρτιά πρέπει να "ανακατευτούν" με τον ακόλουθο τρόπο:

- Διαλέξτε μια θέση στην κλίμακα 1-15 π.χ. 17.

- Τοποθετήστε το χαρτί της θέσης αυτής σε μια προσωρινή αποθήκευση.

- Φέρτε όλα τα χαρτιά των θέσεων 52 ως 18 στις θέσεις 51 ως 17.

- Τοποθετήστε το αποθηκευμένο χαρτί στη θέση 52.

- Κάντε το ίδιο με τις κλίμακες 1-50, 1-49 ... μέχρι 1-2 ώστε η τράπουλα να ανακατευτεί καλά.

- Εξάγετε το αποτέλεσμα της ανάμιξης.

6. Δημιουργήστε έξι εντολές DATA όπου κάθε μια θα περιέχει ένα επώνυμο, αρχικά και ένα τηλεφωνικό αριθμό (αριθμό τηλεφώνου και αυτόματο περιοχής). Βρέστε μια κατάλληλη δομή πινάκων για να αποθηκεύσετε αυτές τις πληροφορίες και μεταφέρετε τις με READ στους πίνακες.

   Τυπώστε τα δεδομένα χρησιμοποιώντας έναν ξεχωριστό βρόχο FOR και εξηγείστε πως η μορφή των δεδομένων εισόδων (DATA), η εσωτερική μορφή (πίνακες) και η μορφή εξόδων δεν είναι απαραίτητα η ίδια.

# ΚΕΦΑΛΑΙΟ 14

# ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Σ' αυτό το κεφάλαιο επανερχόμαστε στη δομή προγράμματος: βρόχοι και αποφάσεις ή επιλογές. Προσπαθήσαμε να παρουσιάσουμε τα πράγματα όσο γίνεται με πιο απλό τρόπο αλλά η SuperBASIC είναι σχεδιασμένη έτσι ώστε να τα βγάζει πέρα και με τα πολύπλοκα και με όλα τα επίπεδα μεταξύ αυτών. Μερικά τμήματα του κεφαλαίου αυτού είναι δύσκολα και αν είστε νέος στον προγραμματισμό ίσως προτιμάτε να παραλείψετε μερικά. Τα θέματα που καλύπτονται είναι:

- Βρόχοι
- Φωλιασμένοι βρόχοι
- Δυαδικές αποφάσεις
- Πολλαπλές αποφάσεις

Τα τελευταία μέρη του πρώτου τμήματος, των βρόχων, γίνονται δύσκολα καθώς δείχνουμε πως η SuperBASIC τα βγάζει πέρα με προβλήματα που οι άλλες γλώσσες απλώς αγνοούν. Ξεπεράστε αυτά τα τμήματα αν θέλετε, πάντως τα άλλα μέρη είναι πιο ευκολονόητα.

## ΒΡΟΧΟΙ

Σ' αυτό το τμήμα επιχειρούμε να επιδείξουμε τα γνωστά προβλήματα του χειρισμού επαναλήψεων προσομοιώνοντας μερικές σκηνές της 'Αγριας Δύσης. Η ιδέα μπορεί να είναι χρησιμοποιημένη και απλή αλλά προσφέρει μια απλή βάση για συζήτηση και και επιδεικνύει δυσκολίες που συμβαίνουν σ' όλη την κλίμακα των εφαρμογών του προγραμματισμού.

### ΠΑΡΑΔΕΙΓΜΑ 1

'Ενας ληστής έχει καταφύγει στο παλιό σχολείο. Ο σερίφης έχει έξι σφαίρες στο όπλο του. Προσομοιώστε τους έξι πυροβολισμούς.

### ΠΡΟΓΡΑΜΜΑ 1

```
100 REMark Western FOR
110 FOR bullets = 1 TO 6
120   PRINT "Take aim"
130   PRINT "Fire shot"
140 END FOR bullets
```

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 REMark Western REPeat
110 LET bullets = 6
120 REPeat bandit
130   PRINT "Take aim"
140   PRINT "fire shot"
150   LET bullets = bullets - 1
160   IF bullets = 0 THEN EXIT bandit
170 END REPeat bandit
```

Και τα δύο αυτά προγράμματα παράγουν την ίδια έξοδο:

```
Take aim
Fire a shot
```

που τυπώνει έξι φορές.

Αν σε κάθε πρόγραμμα το 6 αλλάξει σε οποιοδήποτε αριθμό μέχρι 1 τότε και τα δύο προγράμματα δουλεύουν όπως θα περιμένατε. Αλλά τι γίνεται αν το όπλο είναι άδειο πριν ριχθούν οι πυροβολισμοί;

**ΠΑΡΑΔΕΙΓΜΑ 2**

Υποθέτουμε ότι κάποιος έχει βγάλει κρυφά όλες τις σφαίρες από το όπλο του σερίφη. Τι θα συμβεί αν αλλάξετε απλώς το 6 σε 0 σε κάθε πρόγραμμα;

**ΠΡΟΓΡΑΜΜΑ 1**

```
100 REMark Western FOR Zero Case
110 FOR bullets = 1 to 0
120   PRINT "Take aim"
130   PRINT "Fire a shot"
140 END FOR bullets
```

Αυτό δουλεύει σωστά. Δεν υπάρχει έξοδος. Η "μηδενική περίπτωση" συμπεριφέρεται σωστά στη SuperBASIC.

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 REMark Western REPeat Fails
110 LET bullets = 0
120 REPeat bandit
130   PRINT "Take aim"
140   PRINT "Fire shot"
150   LET bullets = bullets -1
160   IF bullets = 0 THEN EXIT bandit
170 END REPeat bandit
```

Το πρόγραμμα αποτυγχάνει σε δύο περιπτώσεις:

[1] Take aim
Fire a shot
εκτυπώνεται ενώ δεν υπάρχουν σφαίρες.

[2] 'Οταν η μεταβλητή bullets ελέγχεται στη γραμμή 70 έχει τιμή -1 και δε γίνεται ποτέ μηδέν. Το πρόγραμμα

ανακυκλώνει επ' άπειρον. Η άπειρη ανακύκλωση θα διορθωθεί αν γράψετε:

```
70 IF bullets < 1 THEN EXIT bandit
```

Υπάρχει ένα λάθος στον προγραμματισμό το οποίο δεν επιτρέπει για τη πιθανή τιμή μηδέν. Αυτό μπορεί να διορθωθεί βάζοντας την υπό συνθήκη EXIT πριν τις εντολές PRINT.

**ΠΡΟΓΡΑΜΜΑ 3**

```
100 REMark Western REPeat Zero Case
110 LET bullets = 0
120 REPeat Bandit
130    IF bullets = 0 THEN EXIT Bandit
140    PRINT "Take aim"
150    PRINT "Fire shot"
160    LET bullets = bullets -1
170 END REPeat Bandit
```

Το πρόγραμμα αυτό εργάζεται τώρα σωστά οποιαδήποτε είναι η αρχική τιμή του bullets υπό την προυπόθεηση να έχει τιμή μεγαλύτερη ή ίση με μηδέν. Η μέθοδος 2 αντιστοιχεί με το βρόχο REPEAT...UNTIL μερικών γλωσσών. Η μέθοδος 3 αντιστοιχεί με το βρόχο WHILE...ENDWHILE μερικών γλωσσών. Πάντως το REPeat...END REPeat με EXIT είναι περισσότερο εύχρηστο παρά συνδυασμός και των δύο. Αν έχετε χρησιμοποιήσει άλλες BASIC θα ερωτιέστε τι συμβαίνει με τη NEXT εντολή. θα το ξαναεισαγάγουμε αλλά θα δείτε ότι και οι δύο βρόχοι έχουν παρόμοια δομή και είναι ονομασμένοι.

| | | | |
|---|---|---|---|
| **FOR** | name = εντολές | **REPeat** | name = εντολές |
| **END FOR** | name | **END REPeat** | name |

Επί πλέον ο βρόχος REPeat πρέπει φυσιολογικά να έχει στις εντολές του την EXIT αλλιώς δε θα τερματιστεί ποτέ.

Σημειώστε ότι η εντολή EXIT κάνει τον έλεγχο να μεταφερθεί αμέσως μετά την εντολή END του βρόχου.

Μια εντολή NEXT μπορεί να τοποθετηθεί σ' ένα βρόχο. Μεταφέρει τον έλεγχο ακριβώς αμέσως μετά τη δεσμευμένη λέξη ανοίγματος FOR ή REPeat. Μπορεί να θεωρηθεί σαν ένα είδος αντιθέτου της εντολής EXIT. Κατά σύμπτωση και οι δύο λέξεις περιέχουν το EXT. Σκεφθείτε για μια EXTension των βρόχων και:

N σημαίνει "Τώρα αρχίστε πάλι"
I σημαίνει "Τελείωσε"

Η κατάσταση είναι η ίδια όπως στο παράδειγμα 1. Ο σερίφης έχει ένα όπλο με 6 σφαίρες και πρέπει να πυροβολήσει το ληστή αλλά ισχύουν δύο συνθήκες.

**ΠΑΡΑΔΕΙΓΜΑ 3**

[1] Αν κτυπήσει το ληστή σταματά να πυροβολεί και γυρίζει στο Dodge City.

[2] Αν τελειώσουν οι σφαίρες πριν κτυπηθεί ο ληστής λέει στο βοηθό του να προσέχει το ληστή ενώ αυτός (ο σερίφης) επιστρέφει στο Dodge City.

**ΠΡΟΓΡΑΜΜΑ 1**

```
100 REMark Western FOR with Epilogue
110 FOR bullets = 1 TO 6
120    PRINT "Take aim"
130    PRINT "FIRE A SHOT"
140    LET hit = RND(9)
150    IF hit = 7 THEN EXIT bullets
160 NEXT bullets
170    PRINT "Watch Bandit"
180 END FOR bullets
190 PRINT "Return to Dodge City"
```

Σ' αυτήν την περίπτωση το περιεχόμενο μεταξύ του FOR και του END FOR είναι ένα είδος επίλογου που εκτελείται μόνο αν ο βρόχος FOR κάνει όλη την πορεία του. Αν υπάρχει πρόωρο EXIT τότε ο επίλογος δεν εκτελείται.

Το ίδιο αποτέλεσμα μπορούμε να πετύχουμε και με ένα βρόχο REPeat αν και αυτός δεν είναι απαραίτητα και ο καλύτερος τρόπος. 'Ομως αξίζει να το κοιτάξετε (ίσως σε μια δεύτερη ανάγνωση) αν θέλετε να κατανοήσετε δομές που είναι απλές για να χρησιμοποιούνται με απλό τρόπο και ισχυρές για να τα βγάζουν πέρα με άτεχνες καταστάσεις όταν χρειαστεί.

**ΠΡΟΓΡΑΜΜΑ 2**

```
100 REMark Western REPeat with Epilogue
110 LET bullets = 6
120 REPeat Bandit
130    PRINT "Take aim"
140    PRINT "Fire shot"
150    LET hit = RND(9)
160    IF hit = 7 THEN EXIT Bandit
170    LET bullets = bullets -1
180    IF bullets <> 0 THEN NEXT Bandit
190    PRINT "Watch Bandit"
200 END REPeat Bandit
210 PRINT "Return to Dodge City"
```

Το πρόγραμμα δουλεύει σωστά μόνο όταν ο σερίφης έχει τουλάχιστον μία σφαίρα στην αρχή. Αποτυχαίνει αν η γραμμή 110 είναι:

```
110 LET bullets = 0
```

θα σκεφτείτε ότι ο σερίφης θα ήταν πολύ βλάκας για να ξεκινήσει μια επιχείρηση τέτοιου είδους χωρίς να έχει σφαίρες και θα έχετε δίκιο. Τώρα συζητάμε πως να διατηρήσουμε τη καλή δόμηση στην πιο περίπλοκη κατάσταση. Τουλάχιστον κρατήσαμε την ιδέα του προβλήματος απλή, γνωρίζουμε τι προσπαθούμε να κάνουμε. Τα πολύπλοκα δομικά προβλήματα συνήθως συναντώνται σε ιδέες πιο δύσκολες από τις προσομοιώσεις της 'Αγριας Δύσης. Αλλά αν θέλετε πραγματικά μια λύση στο πρόβλημα που να φροντίζει για ένα πιθανό χτύπημα, το τελείωμα των σφαιρών και έναν επίλογο, και ακόμη για τη μηδενική περίπτωση, τότε προσθέστε την ακόλουθη γραμμή στο παραπάνω πρόγραμμα.

```
125 IF bullets = 0 THEN PRINT "Watch Bandit" : EXIT bandit
```

Δεν μπορούμε να επινοήσουμε πιο πολύπλοκο τύπο προβλήματος απ' αυτό με ένα βρόχο. Η SuperBASIC μπορεί εύκολα να το χειριστεί, αν θέλετε.

## ΦΩΛΙΑΣΜΕΝΟΙ ΒΡΟΧΟΙ

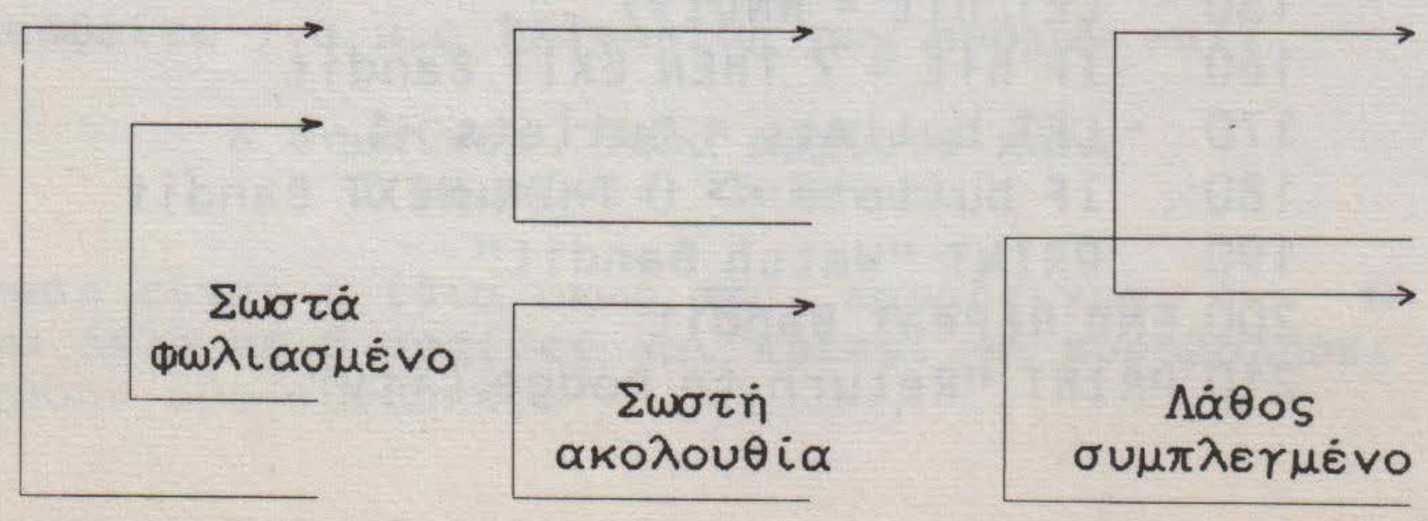
Δέστε τον ακόλουθο βρόχο FOR που σχεδιάζει μια σειρά σημείων με διάφορα τυχαία επιλεγμένα χρώματα (όχι μαύρο).

```
100 REMark Row of pixels
110 PAPER 0 : CLS
120 LET up = 50
130 FOR across = 20 TO 60
140   INK RND(2 TO 7)
150   POINT across, up
160 END FOR across
```

Το πρόγραμμα σχεδιάζει μια σειρά από σημεία έτσι:

.....

Αν θέλετε να πάρετε ας πούμε 51 σειρές από σημεία πρέπει να σχεδιάσετε μια σειρά για κάθε τιμή του up από 30 ως 80. Πρέπει όμως πάντα να λαμβάνετε υπ' όψη τον κανόνα ότι μια δομή μπορεί να είναι εντελώς μέσα σε μια άλλη ή μπορεί κατάλληλα να την περιβάλλει. Μπορεί επίσης να την ακολουθεί αλλά δεν μπορεί να "μπερδεύεται" με μια άλλη δομή. Τα βιβλία προγραμματισμού συνήθως δείχνουν πως οι βρόχοι FOR μπορούν να συνδυαστούν με ένα διάγραμμα όπως:



Στη SuperBASIC ο κανόνας ισχύει για όλες τις δομές. Μπορείτε να λύσετε όλα τα προβλήματα χρησιμοποιώντας τις σωστά. Γι' αυτό μεταχειριζόμαστε το βρόχο FOR σαν μια οντότητα και σχεδιάζουμε ένα νέο πρόγραμμα:

```
FOR up = 30 TO 80
  FOR across = 20 TO 60
    INK RND(2 To 7)
      POINT across, up
  END FOR across
END FOR up
```

'Οταν το μεταφράσουμε αυτό σε πρόγραμμα τότε δικαιούμαστε όχι μόνο να περιμένουμε να δουλέψει αλλά να ξέρουμε και τι θα κάνει. θα σχεδιάσει ένα παραλληλόγραμμο φτιαγμένο από σειρές σημείων.

```
100 REMark Rows of pixels
110 PAPER 0 : CLS
120 FOR up = 30 TO 80
140   FOR accross = 20 TO 60
150     INK RND(2 TO 7)
160     POINT across,up
170   END FOR across
180 END FOR up
```

Διαφορετικές δομές μπορούν να φωλιαστούν. Υποθέτουμε πως αντικαθιστούμε τον εσωτερικό βρόχο FOR του παραπάνω προγράμματος μ' ένα βρόχο REPeat. θα τελειώσουμε το βρόχο REPeat όταν ο μηδενικός χρωματικός κώδικας εμφανιστεί από μια επιλογή στην κλίμακα 0 ως 7.

```
100 REMark REPeat in FOR
110 PAPER 0 : CLS
120 FOR up = 30 TO 80
130   LET across = 19
140   REPeat dots
150     LET colour = RND(7)
160     INK colour
170     LET across = across + 1
180     POINT across, up
190     IF colour = 0 then EXIT dots
200   END REPeat dots
210 END FOR up
```

Η περισσότερη σοφία σχετικά με τον έλεγχο και τη δομή των προγραμμάτων μπορεί να εκφραστεί σε δύο κανόνες.

[1] Κατασκευάστε το πρόγραμμά σας χρησιμοποιώντας μόνο τις νόμιμες δομές για βρόχους και λήψεις αποφάσεων.

[2] Κάθε δομή πρέπει να είναι σωστά σε σειρά ή ολόκληρη μέσα σε μια άλλη.

**ΔΥΑΔΙΚΕΣ ΑΠΟΦΑΣΕΙΣ**

Οι τρεις τύποι δυαδικών αποφάσεων μπορούν να επιδειχθούν εύκολα λέγοντας τι να κάνουμε όταν βρέχει.

```
 i.  100 REMark Short form IF
     110 LET rain = RND(0 TO 1)
     120 IF rain THEN PRINT "Open brolly"

ii.  100 REMark Long form IF...END IF
     110 LET rain = RND(0 TO 1)
     120 IF rain THEN
     130   PRINT "Wear coat"
     140   PRINT "Open brolly"
     150   PRINT "Walk fast"
     160 END IF

iii. 100 REMark Long form IF ...ELSE...END IF
     110 LET rain = RND(0 TO 1)
     120 IF rain THEN
     130   PRINT "Take a bus"
     140 ELSE
     150   PRINT "Walk"
     160 END IF
```

'Ολες αυτές είναι δυαδικές αποφάσεις. Τα πρώτα δύο παραδείγματα είναι απλά, είτε κάτι συμβαίνει είτε δε συμβαίνει. Το τρίτο είναι μια γενική δυαδική απόφαση με δύο διακριτές πιθανές πορείες δράσης που και οι δύο πρέπει να οριστούν.

Μπορείτε να παραλείψετε το THEN στις μεγάλες μορφές αν θέλετε. Στις μικρές μορφές μπορείτε να αντικαταστήσετε το THEN με:

**ΠΑΡΑΔΕΙΓΜΑ**

Δέστε ένα πιο πολύπλοκο παράδειγμα στο οποίο φαίνεται φυσικό το φώλιασμα δυαδικών αποφάσεων. Αυτή η μορφή φωλιάσματος μπορεί να σας μπερδέψει και πρέπει να την κάνετε μόνο όταν φαίνεται απόλυτα φυσικό. Η προσοχή στην παρουσίαση και ιδίως στην ταυτοποίηση είναι πολύ σημαντική.

Αναλύστε ένα τμήμα κειμένου για να μετρήσετε τον αριθμό των φωνηέντων, συμφώνων και άλλων χαρακτήρων. Αγνοήστε τα διαστήματα. Για χάρη απλότητας το κείμενο είναι ολόκληρο σε κεφάλαια.

"COMPUTER HISTORY WAS MADE IN 1984"

READ δεδομένα

FOR κάθε χαρακτήρα

```
IF γράμμα THEN

IF φωνήεν

αύξησε το μετρητή φωνηέντων

     ELSE
        αύξησε το μετρητή συμφώνων
     END IF
   ELSE
      IF όχι διάστημα THEN αύξησε άλλο μετρητή
    ENDIF
  ENDFOR
PRINT αποτελέσματα.
```

```
100 REMark Character Counts
110 RESTORE 290
120 READ text$
130 LET vowels = 0 : cons = 0 : others = 0
140 FOR num = 1 TO LEN(text[)
150    LET ch$ = text$(num)
160    IF ch$ >= "A" AND ch$ <= "Z"
170       IF ch$ INSTR "AEIOU"
180          LET vowels = vowel + 1
190       ELSE
200          LET cons = cons + 1
210       END IF
220    ELSE
230       IF ch$ <> " " THEN others = others + 1
240    END IF
250 END FOR num
260 PRINT "Vowel count is" ! vowels
270 PRINT "Consonent count is" ! cons
280 PRINT "Other count is" ! others
290 DATA "COMPUTER HISTORY WAS MADE IN 1984"

Vowel count is 9
Consonant count is 15
Other count is 4
```

**ΠΟΛΛΑΠΛΕΣ ΑΠΟΦΑΣΕΙΣ - SELect**

'Οταν υπάρχουν τρεις ή περισσότερες πιθανές πορείες δράσης και καμιά δεν εξαρτάται από προηγούμενη επιλογή τότε η δομή που πρέπει φυσιολογικά να χρησιμοποιήσετε είναι η SELect που επιτρέπει επιλογή από οποιοδήποτε αριθμό πιθανοτήτων.

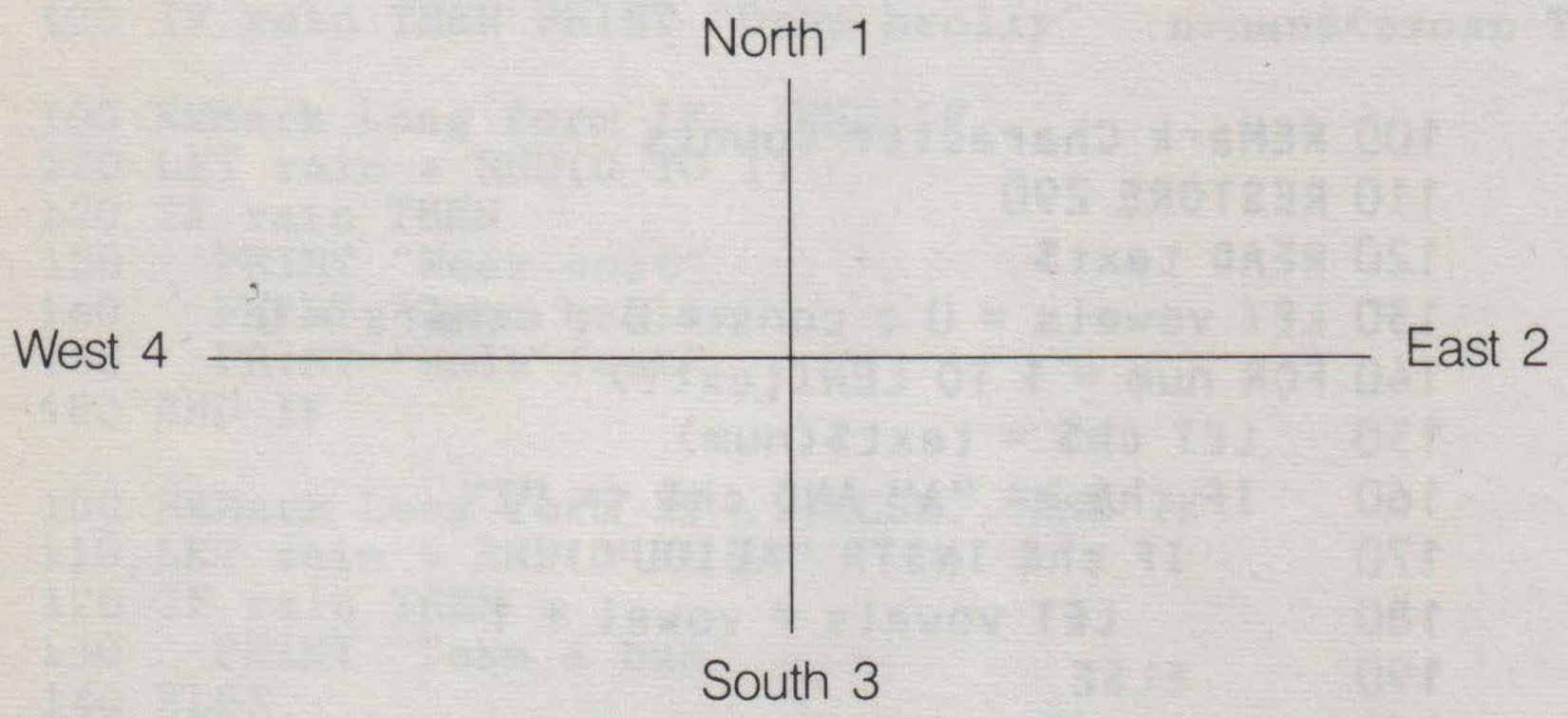
**ΠΑΡΑΔΕΙΓΜΑ**

'Ενα μαγικό φίδι μεγαλώνει απεριόριστα προσθέτοντας ένα τμήμα στο μπρος του μέρος. Κάθε τμήμα μπορεί να είναι μέχρι είκοσι

μονάδες μακρύ και μπορεί να είναι με νέο χρώμα ή με το ίδιο. Κάθε νέο τμήμα πρέπει να μεγαλώσει προς μία από τις διευθύνσεις Βορράς, Νότος, Ανατολή και Δύση. Το φίδι αρχίζει από το κέντρο του παραθύρου.

**ΜΕΘΟΔΟΣ**

Σε οποιαδήποτε στιγμή, όσο το φίδι είναι ακόμη στην οθόνη, μπορείτε εύκολα να διαλέξετε ένα τυχαίο μήκος και χρώμα μελάνης. Η κατεύθυνση μπορεί να επιλεγεί μ' έναν αριθμό 1, 2, 3 ή 4 όπως φαίνεται:




```
Επιλογή PAPER
Τοποθέτηση του φιδιού στο κέντρο του παραθύρου REPeat
   Επιλογή κατεύθυνσης, χρώματος και μήκους αύξησης
   FOR μονάδα = 1 TO αύξηση
     Μεγάλωμα του φιδιού βόρεια, νότια ανατολικά ή δυτικά
     IF το φίδι από το παράθυρο THEN EXIT
   END FOR
END REPeat
PRINT μήνυμα τέλους
```

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Magic Snake
110 PAPER 0 : CLS
120 LET across = 50 : up = 50
130 REPeat snake
140   LET direction = RND(1 TO 4) : colour = RND(2 TO 7)
150   LET growth = RND(2 TO 20)
160   INK colour
170   FOR unit = 1 TO growth
180     SELect ON direction
190       ON direction = 1
200         LET up = up + 1
210       ON direction = 2
220         LET across = across + 1
```

```
230         ON direction = 3
240            LET up = up - 1
250         ON direction = 4
260            LET across = across - 1
270      END SELect
280      IF across<1 OR across>99 OR up<1 OR up>99 THEN EXIT snake
290      POINT across,up
300   END FOR unit
310 END REPeat snake
320 PRINT "Snake off edge"
```

Η σύνταξη της SELect ON δομής επίσης επιτρέπει τη δυνατότητα επιλογής από μια λίστα τιμών όπως:

```
5,6,8,10 TO 13
```

Είναι επίσης δυνατό να εκτελεστεί μια πράξη αν καμμία από τις δηλωμένες τιμές δε βρεθεί. Η πλήρης δομή είναι της μορφής που φαίνεται παρακάτω:

**ΜΙΚΡΗ ΜΟΡΦΗ**

```
SELect ON num

ON    num = λίστα τιμών
      εντολές

ON    num = λίστα τιμών
      εντολές
      -
      -
      -
      -

ON    num = REMAINDER
      εντολές

END   SELect
```

όπου το num είναι οποιαδήποτε αριθμητική μεταβλητή και ο όρος REMAINDER είναι προαιρετικός.

Υπάρχει μια μικρή μορφή της δομής SELect. Για παράδειγμα:

```
100 INPUT num
110 SELect ON num = 0 TO 9 : PRINT "digit"
```

θα κάνει ότι περιμένατε.

**ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 14**

1. Αποθηκεύστε 10 αριθμούς σ' έναν πίνακα και κάντε μια "ταξινόμηση φυσαλίδας". Αυτό γίνεται συγκρίνοντας το πρώτο ζευγάρι και ανταλλάσσοντας τις θέσεις τους αν χρειάζεται, συγκρίνοντας το δεύτερο ζευγάρι (δεύτερος και

τρίτος αριθμός) μέχρι και το ένατο ζευγάρι (ένατος και δέκατος αριθμός). Το πρώτο τρέξιμο των εννιά συγκρίσεων και πιθανών ανταλλαγών εγγυάται ότι ο μεγαλύτερος αριθμός θα φτάσει στη σωστή του θέση. Ακόμη οκτώ τρεξίματα θα έχουν αποτέλεσμα οκτώ ακόμη σωστές θέσεις αφήνοντας μόνο το μικρότερο αριθμό ο οποίος πρέπει να είναι στη μοναδική (σωστή) θέση που έχει απομείνει. Η απλούστερη μορφή "ταξινόμησης φυσαλίδας" δέκα αριθμών απαιτεί εννιά τρεξίματα εννιά συγκρίσεων το καθένα.

2. Βρείτε τρόπους να επιταχύνετε την "ταξινόμηση φυσαλίδας" αλλά μην περιμένετε ότι θα είναι ποτέ ιδιαίτερα αποτελεσματική.

3. 'Ενας υπάλληλος σ' ένα πλειστηριασμό θέλει να πουλήσει ένα παλιό ρολόι και έχει εντολές να κάνει την πρώτη προσφορά στις 50 λίρες. Αν κανείς δεν προσφέρει, τότε μπορεί να κατέβει στις 40, 30, 20 λίρες αλλά όχι πιο κάτω, σε μια προσπάθεια να αποσυρθεί από την πώληση. 'Οταν αρχίσει η προσφορά, δέχεται μόνο αυξήσεις 5 λίρες μέχρι να γίνει η τελική προσφορά. Αν η τελική προσφορά είναι 35 λίρες (η "τιμή ασφάλειας") ή περισσότερα, το ρολόι πουλιέται. Αλλιώς αποσύρεται.

   Προσομοιώστε τον πλειστηριασμό χρησιμοποιώντας το ανάλογο μιας ζαριάς για να ξεκινήσει τις προσφορές. Θα ξεκινήσουν αν τύχει "έξι" σε οποιαδήποτε από τις τιμές έναρξης.

   'Οταν αρχίσουν οι προσφορές πρέπει να υπάρχουν τρεις πιθανότητες στις τέσσερις να υπάρξει ψηλότερη προσφορά σε κάθε κάλεσμα.

4. Σε μια σκηνή από την 'Αγρια Δύση, ο σερίφης δεν έχει πυρομαχικά και θέλει να συλλάβει έναν πιστολά που κατασκήνωσε στο δάσος. Τρέχει με το άλογό του ανάμεσα στα δένδρα προκαλώντας τον πιστολά να πυροβολήσει. Ελπίζει ότι όταν ο πιστολάς πυροβολήσει έιη φορές, θα μπορέσει να τρέξει και να νικήσει τον πιστολά καθώς αυτός θα προσπαθεί να ξαναγεμίσει. Προσομοιώστε τη συμπλοκή, δίνοντας στον πιστολά μια πιθανότητα στις είκοσι να χτυπήσει το σερίφη με κάθε πυροβολισμό. Αν ο σερίφης δε χτυπηθεί μετά από έξι πυροβολισμούς τότε θα συλλάβει τον πιστολά.

5. Οι οδηγίες του σερίφη στο βοηθό του είναι:

   "Αν το όπλο είναι άδειο ξαναγέμισε το και αν δεν είναι τότε συνέχισε να πυροβολάς μέχρι να χτυπήσεις το ληστή ή να παραδοθεί. Αν εμφανιστεί ο Mexico Pete φύγε γρήγορα".

Γράψτε το πρόγραμμα που να φροντίζει κατάλληλα για όλες αυτές τις καταστάσεις:

- 'Οτι και να συμβεί επέστρεψε στο Dodge City.
- Αν εμφανιστεί ο Mexico Pete τότε επέστρεψε αμέσως.
- Αν το όπλο είναι άδειο ξαναγέμισέ το.

- Αν το όπλο δεν είναι άδειο ζήτα από το ληστή να παραδοθεί.

- Αν ο ληστής παραδοθεί συνέλαβέ τον.

- Αν δεν παραδοθεί τότε πυροβόλησε.

- Αν χτυπήσεις το ληστή, συνέλαβέ τον και φρόντισε το τραύμα του.

Υποθέστε ότι έχετε απεριόριστα πολεμοφόδια. Χρησιμοποιήστε ένα προσομοιωμένο "ζάρι είκοσι πλευρών" ώστε το επτά να σημαίνει "παραδίνομαι" και το δεκατρία να σημαίνει ότι ο ληστής χτυπήθηκε.

ΚΕΦΑΛΑΙΟ 15

# ΔΙΑΔΙΚΑΣΙΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Στο πρώτο τμήμα αυτού του κεφαλαίου εξηγούμε τα πιο άμεσα χαρακτηριστικά των διαδικασιών και των συναρτήσεων της SuperBASIC. Το κάνουμε αυτό με πολύ απλά παραδείγματα ώστε να καταλαβαίνετε τη λειτουργία κάθε χαρακτηριστικού καθώς περιγράφεται. Παρόλο που τα παραδείγματα είναι απλά και έχουν ξαναειπωθεί, θα εκτιμήσετε το ότι οι ιδέες, αφού κατανοηθούν, μπορούν να εφαρμοστούν σε πιο πολύπλοκες καταστάσεις όπου πραγματικά έχει σημασία.

Μετά το πρώτο τμήμα υπάρχει μια συζήτηση που προσπαθεί να εξηγήσει "Γιατί διαδικασίες". Αν καταλαβαίνετε, λίγα ή πολλά, μέχρι αυτό το σημείο, τότε πάτε καλά και θα μπορείτε να χρησιμοποιείτε διαδικασίες και συναρτήσεις με αυξανόμενη αποδοτικότητα.

Η SuperBASIC είναι η πρώτη που σας επιτρέπει να κάνετε τα απλά πράγματα με απλούς τρόπους και μετά σας προσφέρει περισσότερα αν θέλετε. Οι επιπλέον ευκολίες και μερικά τεχνικά θέματα εξηγούνται στο δεύτερο τμήμα αυτού του κεφαλαίου αλλά μπορείτε να τα παραλείψετε, σίγουρα στην πρώτη ανάγνωση, και πάλι να βρίσκεστε σε ισχυρότερη θέση από τους περισσότερους χρήστες άλλων BASIC.

## ΤΙΜΕΣ ΠΑΡΑΜΕΤΡΩΝ

'Εχετε δει σε προηγούμενα κεφάλαια πως μια τιμή μπορεί να περάσει σε μια διαδικασία. Εδώ είναι ένα άλλο παράδειγμα.

## ΠΑΡΑΔΕΙΓΜΑ

Στο "Chan's Chinese Take-Away" υπάρχουν μόνο έξι θέματα στο μενού.

| Πιάτα ρυζιού | Γλυκά |
|---|---|
| 1 γαρίδες | 4 παγωτό |
| 2 κοτόπουλο | 5 τηγανιτό |
| 3 σπέσιαλ | 6 κρέμα |

Ο Chan έχει έναν απλό τρόπο για να υπολογίζει τις τιμές. Δουλεύει με πέννες και οι τιμές είναι:

- για ένα πιάτο ρυζιού 300 + 10 φορές τον αριθμό του μενού

- για ένα γλυκό 12 φορές τον αριθμό του μενού

'Ετσι ένας πελάτης που έφαγε ρύζι σπέσιαλ και παγωτό θα πληρώσει:

300 + 10 * 3 + 12 * 4 = 378 pence

Μια διαδικασία με όνομα item, δέχεται έναν αριθμό μενού ως τιμή παραμέτρου και εξάγει το κόστος.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Cost of Dish
110 item 3
120 item 4
130 DEFine PROCedure item(num)
140   IF num <= 3 THEN LET price = 300 + 10 * num
150   IF num >= 4 THEN LET price = 12 * num
160   PRINT !price!
170 END DEFine
```

**'Εξοδος**

```
330 48
```

Στο κύριο πρόγραμμα χρησιμοποιούνται οι πραγματικές παράμετροι 3 και 4. Ο ορισμός της διαδικασίας έχει μια τυπική παράμετρο τη num, η οποία παίρνει την τιμή που της δίνεται από το κύριο πρόγραμμα. Σημειώστε ότι οι τυπικές παράμετροι πρέπει να είναι σε παρένθεση, ενώ οι πραγματικές παράμετροι δε χρειάζονται παρένθεση.

**ΠΑΡΑΔΕΙΓΜΑ**

Τώρα υποθέτουμε ότι η μεταβλητή που δουλεύεται, η price, χρησιμοποιείται επίσης στο κύριο πρόγραμμα, όπου σημαίνει κάτι άλλο, ας πούμε την τιμή ενός ποτηριού μπύρας, 70 πέννες. Το ακόλουθο πρόγραμμα αποτυχαίνει να δώσει το επιθυμητό αποτέλεσμα.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Global price
110 LET price = 70
120 item 3
130 item 4
140 PRINT !price!
150 DEFine PROCedure item(num)
160   IF num <= 3 THEN price = 300 + 10 * num
170   IF num >= 4 THEN price = 12 * num
180   PRINT !price!
190 END DEFine
```

**'Εξοδος:**

```
330 48 48
```

Η τιμή της μπύρας έχει αλλάξει από τη διαδικασία. Λέμε ότι η μεταβλητή price είναι γενική επειδή χρησιμοποιείται οπουδήποτε στο πρόγραμμα.

**ΠΑΡΑΔΕΙΓΜΑ**

Κάντε τη μεταβλητή διαδικασίας price, τοπική στη διαδικασία. Αυτό σημαίνει ότι η SuperBASIC θα τη μεταχειρίζεται σαν ειδική μεταβλητή, προσπελάσιμη μόνο μέσα στη διαδικασία. Η μεταβλητή price στο κύριο πρόγραμμα θα είναι διαφορετική παρόλο που έχει το ίδιο όνομα.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark LOCAL price
110 LET price = 70
120 item 3
130 item 4
140 PRINT !price!
150 DEFine PROCedure item(num)
160   LOCAL price
170   IF num <=3 THEN LET price = 300 + 10*num
180   IF num >=4 THEN LET price = 12*num
190   PRINT !price!
200 END DEFine
```

**'Εξοδος:**

```
330 48 70
```

Αυτή τη φορά όλα λειτουργούν σωστά. Η γραμμή 160 έχει αποτέλεσμα να σημειωθεί εσωτερικά η μεταβλητή της διαδικασίας price ότι "ανήκει" μόνο στη διαδικασία item. Η άλλη μεταβλητή price δεν επηρεάζεται. Μπορείτε να δείτε ότι οι τοπικές μεταβλητές είναι χρήσιμες.

**ΠΑΡΑΔΕΙΓΜΑ**

Οι τοπικές μεταβλητές είναι τόσο χρήσιμες ώστε κάνουμε αυτόματα τις τυπικές παραμέτρους της διαδικασίας τοπικές. Αν και δεν το έχουμε αναφέρει πριν, οι παράμετροι όπως το num στο παραπάνω πρόγραμμα δεν μπορούν να ανακατευτούν με τις μεταβλητές του προγράμματος. Για να το αποδείξουμε αυτό θα βγάλουμε τη εντολή LOCAL από το παραπάνω πρόγραμμα και θα χρησιμοποιήσουμε τη num για την τιμή της μπύρας. Επειδή η num είναι τοπική στη διαδικασία, όλα δουλεύουν.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark LOCAL parameter
110 LET num = 70
120 item 3
130 item 4
140 PRINT num
150 DEFine PROCedure item(num)
160   IF num <= 3 THEN LET price = 300 + 10*num
170   IF num >= 4 THEN LET price = 12*num
180   PRINT !price!
190 END DEFine
```

**'Εξοδος:**

```
330 48 70
```

## ΜΕΤΑΒΛΗΤΕΣ ΠΑΡΑΜΕΤΡΟΙ

Μέχρι τώρα έχουμε χρησιμοποιήσει μόνο παραμέτρους για να περάσουμε τιμές σε μια διαδικασία. Αλλά ας υποθέσουμε ότι το κύριο πρόγραμμα θέλει το κόστος ενός είδους να του περαστεί από τη διαδικασία ώστε να υπολογίσει το συνολικό λογαριασμό. Μπορούμε να το κάνουμε αυτό εύκολα βάζοντας άλλη μια παράμετρο στην κλήση της διαδικασίας. Αυτή πρέπει να είναι μεταβλητή γιατί θα λάβει μια τιμή από τη διαδικασία. 'Ετσι την καλούμε μεταβλητή παράμετρο και πρέπει να ανάγεται σε μια αντίστοιχη μεταβλητή παράμετρο στον ορισμό της διαδικασίας.

## ΠΑΡΑΔΕΙΓΜΑ

Χρησιμοποιήστε τις πραγματικές μεταβλητές παραμέτρους cost_1 και cost_2 για να πάρετε τις τιμές της μεταβλητής price από τη διαδικασία. Κάντε το κύριο πρόγραμμα να υπολογίζει και να τυπώνει το συνολικό λογαριασμό.

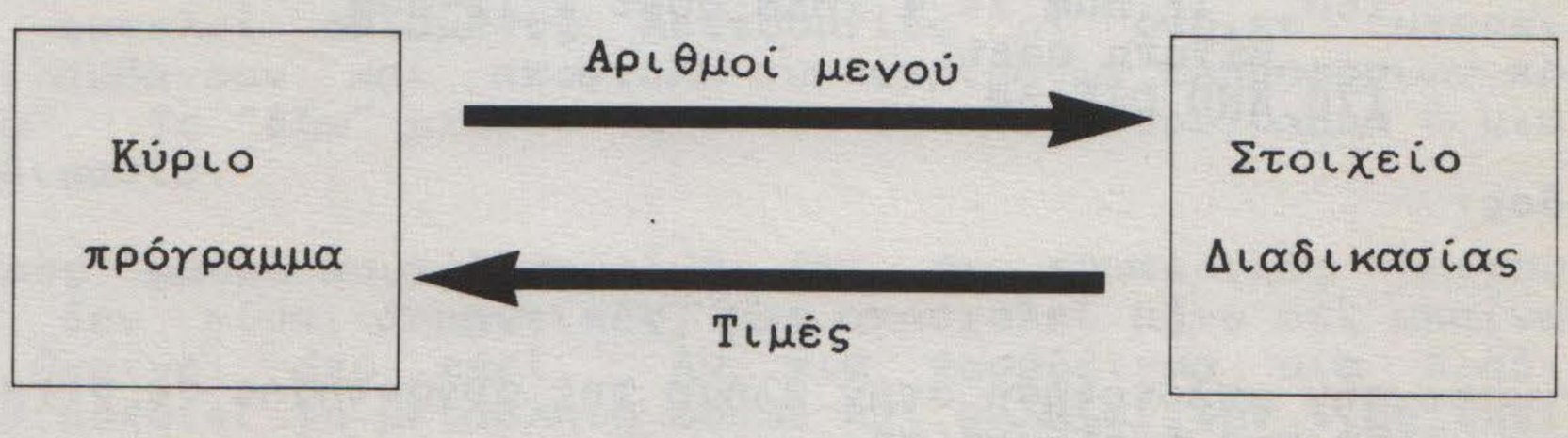
## ΠΡΟΓΡΑΜΜΑ

```
100 REMark Variable parameter
110 LET num = 70
120 item 3,cost_1
130 item 4,cost_2
140 LET bill = num + cost_1 + cost_2
150 PRINT bill
160 DEFine PROCedure item(num, price)
170   IF num < =3 THEN LET price = 300 + 10*num
180   IF num >= 4 THEN LET price = 12*num
190 END DEFine
```

**'Εξοδος:**

```
448
```

Οι παράμετροι num και price είναι αυτόματα τοπικές κι έτσι δεν υπάρχουν προβλήματα. Τα διαγράμματα δείχνουν πως οι πληροφορίες περνούν από το κύριο πρόγραμμα στη διαδικασία και το αντίθετο.




Αυτά είναι αρκετά για διαδικασίες και παραμέτρους προς το παρόν.

## ΣΥΝΑΡΤΗΣΕΙΣ

Γνωρίζετε ήδη πως λειτουργούν οι συναρτήσεις του συστήματος. Για παράδειγμα η συνάρτηση:

```
SQRT(9)
```

υπολογίζει την τιμή 3, που είναι η τετραγωνική ρίζα του 9. Λέμε ότι η συνάρτηση επιστρέφει την τιμή 3. Μια συνάρτηση, όπως μια διαδικασία, μπορεί να έχει μια ή περισσότερες παραμέτρους (ορίσματα) αλλά το διαφοροποιό χαρακτηριστικό της συνάρτησης είναι ότι επιστρέφει ακριβώς μια τιμή. Αυτό σημαίνει ότι μπορείτε να τη χρησιμοποιείτε σε παραστάσεις που ήδη έχετε. Μπορείτε να πληκτρολογήσετε:

```
PRINT 2*SQRT(9)
```

και να πάρετε την έξοδο 6. 'Ετσι η συνάρτηση συμπεριφέρεται όπως ακριβώς και μια διαδικασία με μια ή περισσότερες τιμές παραμέτρων και ακριβώς μια μεταβλητή παράμετρο που περιέχει την επιστρεφόμενη τιμή. Αυτή η μεταβλητή παράμετρος είναι η ίδια με το όνομα της συνάρτησης.

Οι παράμετροι δε χρειάζεται πάντα να είναι αριθμητικές:

```
LEN("string")
```

έχει αλφαριθμητικό όρισμα αλλά επιστρέφει την αριθμητική τιμή 6.

## ΠΑΡΑΔΕΙΓΜΑ

Ξαναγράψτε το πρόγραμμα του τελευταίου τμήματος που χρησιμοποιεί τη μεταβλητή παράμετρο price. Κάντε το price όνομα μιας συνάρτησης.

## ΠΡΟΓΡΑΜΜΑ

```
100 REMark FuNction
110 LET num = 70
120 LET bill = num + price(3) + price(4)
130 PRINT bill
140 DEFine FuNction price(num)
150   IF num <= 3 THEN cost = 300 + 10*num
160   IF num >= 4 THEN cost = 12*num
    RETurn cost
170 END DEFine
```

**'Εξοδος:**

```
448
```

Σημειώστε την απλοποίηση στην κλήση της συνάρτησης σε σύγκριση με την κλήση της διαδικασίας.

Υπάρχει ένας εναλλακτικός τρόπος ορισμού της τιμής που θα επιστραφεί. Χρησιμοποιεί τη λέξη RETURN όπως φαίνεται:

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark FuNction with RETurn
110 LET num = 70
120 LET bill = num + price(3) + price(4)
130 PRINT bill
140 DEFine FuNction price(num)
150   IF num <= 3 THEN RETurn 300 + 10* num
160   IF num >= 4 THEN RETurn 12* num
170 END DEFine
```

**'Εξοδος**

448

Για χάρη συμβατικότητας με άλλες BASIC, είναι επιτρεπτή και μια μικρή μορφή ορισμού συνάρτησης.

**ΠΑΡΑΔΕΙΓΜΑ**

Χρησιμοποιήστε τη μικρή μορφή ορισμού για τη συνάρτηση diff η οποία επιστρέφει τη διαφορά μήκους μεταξύ μιας μακριάς αλφαριθμητικής και μιας κοντής.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Short function
110 PRINT diff("Long string", "Short")
120 DEFine FuNction diff(lon,sho):RETurn LEN(lon)-LEN(sho)
```

**'Εξοδος:**

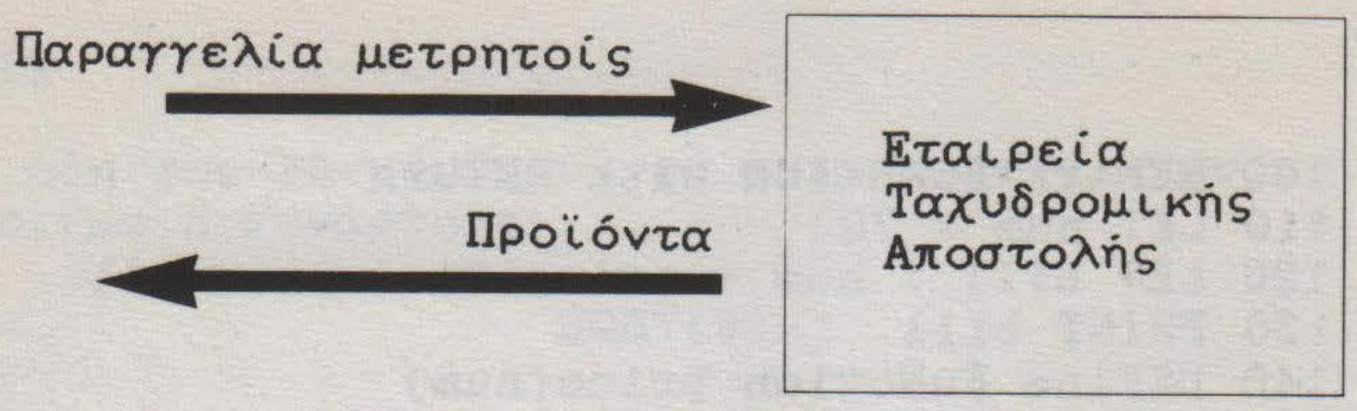
:

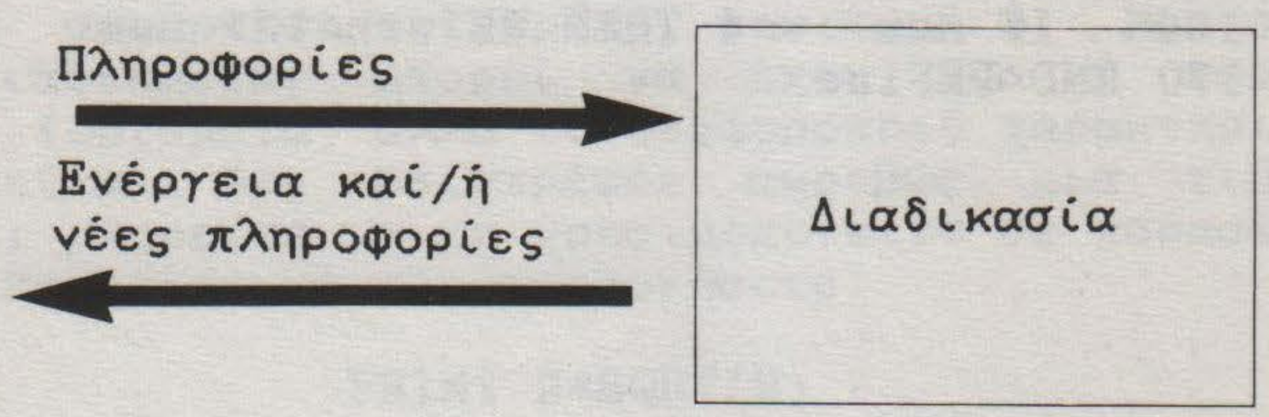
Η συγκοπτόμενη μορφή του FuNction είναι FN για συμβατικοτήτα με τις άλλες BASIC. Σ' αυτό οφείλεται η παράξενη μορφή της στα προγράμματα.

**ΓΙΑΤΙ ΔΙΑΔΙΚΑΣΙΕΣ;**

Η τελική ιδέα μιας διαδικασίας είναι η λειτουργία της σαν "μαύρο κουτί" που δέχεται συγκεκριμένες πληροφορίες από "έξω" και εκτελεί ορισμένες λειτουργίες οι οποίες μπορεί να περιλαμβάνουν και αποστολή συγκεκριμένων πληροφοριών προς τα "έξω". Το "έξω" μπορεί να είναι το κύριο πρόγραμμα ή μια άλλη διαδικασία.

Ο όρος "μαύρο κουτί" σημαίνει ότι οι εσωτερικές λειτουργίες του δεν είναι σημαντικές, μας απασχολεί μόνο ότι μπαίνει και ότι βγαίνει από εκεί. Αν για παράδειγμα μια διαδικασία χρησιμοποιεί τη μεταβλητή count και αλλάζει την τιμή της, αυτό μπορεί να επηρεάσει μια μεταβλητή με το ίδιο όνομα στο κύριο πρόγραμμα. Σκεφτείτε μια εταιρία που δουλεύει με ταχυδρομικές παραγγελίες. Τους στέλνετε μια παραγγελία και μετρητά, σας στέλνουν εμπορεύματα. Σε μια διαδικασία στέλνονται πληροφορίες κι αυτή στέλνει πίσω δράση και/ή νέες πληροφορίες.




θα θέλετε να μη χρησιμοποιήσει η εταιρία ταχυδρομικών παραγγελιών το όνομα και τη διεύθυνσή σας ή άλλε πληροφορίες για άλλους σκοπούς. Αυτό θα ήταν ένα ανεπιθύμητο μειονέκτημα. 'Ομοια δεν θα θέλατε μια διαδικασία να προκαλέσει απρόβλεπτες αλλαγές σε τιμές μεταβλητών που χρησιμοποιούνται στο κύριο πρόγραμμα.

Φυσικά μπορείτε να βεβαιωθείτε ότι δεν υπάρχουν διπλές χρήσεις ονομάτων μεταβλητών σ' ένα πρόγραμμα. Αυτό θα δουλέψει ως ένα σημείο αλλά σας έχουμε δείξει σ' αυτό το κεφάλαιο πως να αποφεύγετε τα προβλήματα ακόμη κι αν ξεχάσετε ποιές μεταβλητές έχουν χρησιμοποιηθεί σε μια συγκεκριμένη διαδικασία.

Ο δεύτερος στόχος της χρήσης διαδικασιών είναι να κάνουν ένα πρόγραμμα διαχωρίσημο σε τμήματα. Είναι καλύτερα αντί να έχετε ένα μακρύ κύριο πρόγραμμα, να χωρίσετε τη δουλειά σε "κομάτια μεγέθους μυαλού" όπως λέει ο Seymour Papert, ο εφευρέτης της LOGO. Αυτά είναι οι διαδικασίες, κάθε μια αρκετά μικρή για να την καταλαβαίνετε και να την ελέγχετε εύκολα. Αυτές συνδέονται με τις κλήσεις διαδικασιών σε μια ιεραρχική κλίμακα.

Ο τρίτος σκοπός είναι να αποφύγουμε να γράψουμε τον ίδιο κώδικα δύο φορές. Γράψτε το μια φορά σαν διαδικασία και καλέστε το δύο φορές αν χρειαστεί.

Δίνουμε παρακάτω άλλο ένα παράδειγμα που δείχνει πως οι διαδικασίες κάνουν ένα πρόγραμμα διαχωρίσιμο.

**ΠΑΡΑΔΕΙΓΜΑ**

Γίνεται μια παραγγελία για έξι πιάτα "Chan's Take Away" όπου το μενού είναι:

| Αριθμός είδους | Πιάτο | Τιμή |
|---|---|---|
| 1 | Γαρίδες | 3.50 |
| 2 | Κοτόπουλο | 2.80 |
| 3 | Σπέσιαλ | 3.30 |

Γράψτε διαδικασίες για τις ακόλουθες αποστολές:

[1] Δημιουργήστε δύο πίνακες τριών στοιχείων που να δείχνουν το μενού, τα πιάτα και τις τιμές. Χρησιμοποιήστε μια εντολή DATA.

[2] Προσομοιώστε μια παραγγελία έξι τυχαία επιλεγμένων πιάτων, χρησιμοποιώντας μια διαδικασία, την choose, και κρατήστε λογαριασμό για τον αριθμό των φορών που επιλέχθηκε κάθε πιάτο.

[3] Περάστε τους τρεις αριθμούς σε μια διαδικασία, τη waiter, η οποία επιστρέφει το κόστος της παραγγελίας στο κύριο πρόγραμμα, χρησιμοποιώντας την παράμετρο cost. Η διαδικασία waiter καλεί δύο άλλες διαδικασίες τις compute και cook, που υπολογίζουν το κόστος και προσομοιώνουν το "μαγείρεμα".

[4] Η διαδικασία cook τυπώνει τον απαιτούμενο αριθμό και το όνομα κάθε πιάτου.

Το κύριο πρόγραμμα πρέπει να καλεί τις διαδικασίες όπως χρειάζεται, να παίρνει το συνολικό κόστος από τη διαδικασία waiter, να προσθέτει 10% για φιλοδώρημα και να τυπώνει το ποσό του τελικού λογαριασμού.

**ΣΧΕΔΙΟ**

Αυτό το πρόγραμμα επιδεικνύει το πέρασμα παραμέτρων με πολύπλοκο τρόπο και γι' αυτό θα εξηγήσουμε το πρόγραμμα βήμα προς βήμα πριν το συναρμολογήσουμε.

```
100 REMark Procedures
110 DIM item$(3,7), price(3), dish(3)
120 LET tip = 0.9
130 set_up
-
-
190 DEFine PROCedure set_up
200   FOR k = 1 TO 3
210     READ item$(k)
220     READ price(k)
230   END FOR k
240 END DEFine
-
-
-
460 DATA "Prawns", 3.5, "Chicken", 2.8, "Special", 3.3
```

Τα ονόματα των ειδών του μενού και οι τιμές τους έχουν τοποθετηθεί στους πίνακες item$ και price.

Το επόμενο βήμα είναι η επιλογή ενός από τους αριθμούς του μενού για καθένα από τους έξι πελάτες. Ο λογαριασμός των φορών που απαιτείται κάθε πιάτο, θα κρατιέται στον πίνακα dish.

```
140 choose dish
-
-
-
250 DEFine PROCedure choose(dish)
260   FOR pick = 1 TO 6
270     LET number = RND(1 TO 3)
280     LET dish(number) = dish(number) + 1
290   END FOR pick
300 END DEFine
```

Σημειώστε ότι η τοπική παράμετρος dish είναι:

+ τοπική στη διαδικασία choose και

+ πίνακας στο κύριο πρόγραμμα.

Οι τρεις τιμές επιστρέφονται στο γενικό πίνακα που επίσης λέγεται dish. Αυτές οι τιμές περνούν μετά στη διαδικασία waiter.

```
150 waiter dish, bill
-
-
-
-
310 DEFine PROCedure waiter(dish, cost)
320   compute dish, cost
330   cook dish
340 END DEFine
```

Η waiter περνά τις πληροφορίες για τον αριθμό κάθε πιάτου που απαιτείται στη διαδικασία compute, που υπολογίζει το κόστος και το επιστρέφει.

```
350 DEFine PROCedure compute(dish, total)
360   LET total = 0
370   FOR k = 1 to 3
380     LET total = total + dish(k)*price(k)
390   END FOR k
400 END DEFine
```

Η waiter επίσης περνά πληροφορίες στην cook που απλώς τυπώνει τον απαιτούμενο αριθμό κάθε είδους του μενού.

```
410 DEFine PROCedure cook(dish)
420   FOR c = 1 TO 3
430     PRINT ! dish(c) ! item(c) !
440   END FOR c
450 END DEFine
```

Και πάλι ο πίνακας dish είναι τοπικός στη διαδικασία cook. Παίρνει τις πληροφορίες που χρησιμοποιεί η διαδικασία στις εντολές PRINT.

Ολόκληρο το πρόγραμμα φαίνεται παρακάτω:

**ПРОГРАММА**

```
100 REMark Procedures
110 RESTORE 490
120 DIM item$(3,7), price(3), dish(3)
130 REMark *** PROGRAM ***
140 LET tip = 0.1
150 set_up
160 choose dish
170 waiter dish, bill
180 LET bill = bill + tip*bill
190 PRINT "Total cost is £" ; bill
200 REMark *** PROCEDURE DEFINITIONS ***
210 DEFine PROCedure set_up
220    FOR k = 1 TO 3
230       READ item$(k)
240       READ price(k)
250    END FOR k
260 END DEFine
270 DEFine PROCedure choose(dish)
280    FOR pick = 1 TO 6
290       LET number = RND(1 TO 3)
300       LET dish(number) = dish(number) + 1
310    END FOR pick
320 END DEFine
330 DEFine PROCedure waiter(dish, cost)
340    compute dish,cost
350    cook dish
360 END DEFine
370 DEFine PROCedure compute(dish, total)
380    LET total = 0
390    FOR k = 1 TO 3
400       LET total = total + dish(k)*price(k)
410    END FOR k
420 END DEFine
430 DEFine PROCedure cook(dish)
440    FOR c = 1 TO 3
450       PRINT ! dish(c) ! item$(c)
460    END FOR c
470 END DEFine
480 REMark *** PROGRAM DATA ***
490 DATA "Prawns", 3.5, "Chicken",2.8,"Special",3.3
```

**'Εξοδος**

Η έξοδος εξαρτάται από την τυχαία επιλογή των πιάτων αλλά η ακόλουθη επιδεικνύει το σχήμα και δίνει ένα δείγμα εξόδου:

```
3 prawns
1 Chicken
2 Special

Total cost is 20.80 pound
```

**ΣΧΟΛΙΑ**

Προφανώς η χρήση διαδικασιών και παραμέτρων σ' ένα τόσο απλό πρόγραμμα δεν είναι απαραίτητη αλλά φανταστείτε ότι κάθε υπό-αποστολή μπορεί να είναι πολύ πιο περίπλοκη. Σε μια τέτοια περίπτωση, η χρήση διαδικασιών θα επέτρεπε μια τμηματική ανάπτυξη του προγράμματος με δοκιμή σε κάθε στάδιο. Το παραπάνω παράδειγμα απλά επιδεικνύει τα βασικά σημεία και τις σχέσεις των διαδικασιών.

'Ομοια το επόμενο παράδειγμα επιδεικνύει τη χρήση συναρτήσεων.

Σημειώστε ότι στο προηγούμενο παράδειγμα, οι διαδικασίες waiter και compute επιστρέφουν ακριβώς μια τιμή. Ξαναγράψτε τις διαδικασίες σαν συναρτήσεις και δείξτε ότι άλλες αλλαγές θα συμβούν εξ' αιτίας της μεταβολής αυτής.

```
DEFine FuNction waiter(dish)
   cook dish
   RETURN compute(dish)
END DEFine

DEFine FuNction compute(dish)
   LET total = 0
   FOR k = 1 TO 3
     LET total = total + dish(k) * price(k)
   END FOR k
   RETURN total
```

Η κλήση συνάρτησης στο waiter παίρνει μια διαφορετική μορφή:

```
LET bill = waiter(dish)
```

Το πρόγραμμα δουλεύει όπως πριν. Σημειώστε ότι υπάρχουν λιγότερες παράμετροι παρόλο που η δομή του προγράμματος είναι παρόμοια. Αυτό συμβαίνει επειδή τα ονόματα των συναρτήσεων λειτουργούν επίσης σαν παράμετροι επιστρέφοντας πληροφορίες στην πηγή απ' όπου καλέστηκαν.

**ΠΑΡΑΔΕΙΓΜΑ**

'Ολες οι μεταβλητές που χρησιμοποιούνται σαν τυπικές παράμετροι σε διαδικασίες ή συναρτήσεις είναι "ασφαλείς" επειδή είναι αυτόματα τοπικές. Ποιες μεταβλητές που χρησιμοποιούνται σε διαδικασίες ή συναρτήσεις δεν είναι τοπικές; Ποιες επιπλέον εντολές θα χρειαστείτε για να τις κάνετε τοπικές;

Αλλαγές Προγράμματος

Οι μεταβλητές k, pick και num δεν είναι τοπικές. Οι απαραίτητες αλλαγές για να γίνουν τοπικές είναι:

```
LOCAL  k

LOCAL  pick, num
```

**ΑΤΥΠΕΣ ΠΑΡΑΜΕΤΡΟΙ**

Οι τυπικές παράμετροι δεν είναι κανενός τύπου. Αυτό δεν το αναφέραμε πριν επειδή μπορείτε να δουλέψετε πολύ καλά χωρίς να το γνωρίζετε. Μπορεί να φαίνονται σαν να είναι κάποιου τύπου και μπορεί να προτιμάτε μια μεταβλητή που να φαίνεται ότι χειρίζεται αριθμούς και μια μεταβλητή που να φαίνεται ότι χειρίζεται αλφαριθμητικές. Αλλά όπως και να γράψετε τις παραμέτρους, δεν είναι κανενός τύπου. Για να το αποδείξουμε αυτό δοκιμάστε το ακόλουθο πρόγραμμα:

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Number or word
110 waiter 2
120 waiter "Chicken"
130 DEFine PROCedure waiter(item)
140   PRINT!item!
150 END DEFine
```

**'Εξοδος**

```
2 Chicken
```

Ο τύπος των παραμέτρων καθορίζεται μόνο όταν η διαδικασία κληθεί και "φτάσει" μια πραγματική παράμετρος.

ΣΚΟΠΟΣ ΜΕΤΑΒΛΗΤΩΝ

Κοιτάξτε το ακόλουθο πρόγραμμα και προσπαθήστε να βρείτε τους δύο αριθμούς που θα εξαχθούν.

```
110 REMark scope
120 LET number = 1
130 test
140 DEFine PROCedure test
150   LOCAL number
160   LET number = 2
170   PRINT number
      try
180 END DEFine
190 DEFine PROCedure try
200   PRINT number
210 END DEFine
```

Προφανώς ο πρώτος αριθμός που θα τυπωθεί είναι το 2, αλλά είναι η μεταβλητή number στη γραμμή 120, γενική;

Η απάντηση είναι ότι η τιμή του number στη γραμμή 150 θα μεταφερθεί στη διαδικασία try. Μια μεταβλητή που είναι τοπική

σε μια διαδικασία θα είναι η ίδια μεταβλητή σε μια δεύτερη διαδικασία που κλήθηκε από την πρώτη.

Με τον ίδιο τρόπο, αν η διαδικασία try κληθεί από το κύριο πρόγραμμα, τότε η μεταβλητή number θα είναι ο ίδιος αριθμός και στο κύριο πρόγραμμα και στη διαδικασία try. Τα συμπεράσματα μπορεί αρχικά να φαίνονται παράξενα αλλά είναι λογικά.

[1] Η μεταβλητή number στη γραμμή 120 είναι γενική.

[2] Η μεταβλητή number στη διαδικασία test είναι τοπική στη διαδικασία.

[3] Η μεταβλητή number στη διαδικασία try, "ανήκει" στο μέρος του προγράμματος από το οποίο καλέστηκε η διαδικασία.

Καλύψαμε πολλές ιδέες σ' αυτό το κεφάλαιο επειδή οι συναρτήσεις και οι διαδικασίες της SuperBASIC είναι πολύ ισχυρές. 'Ομως μην περιμένετε να χρησιμοποιήσετε όλα αυτά τα χαρακτηριστικά αμέσως. Χρησιμοποιήστε διαδικασίες και συναρτήσεις με απλούς τρόπους αρχικά. Αυτό είναι πολύ αποδοτικό, και η ισχύς υπάρχει όποτε τη χρειαστείτε.

## ΠΡΟΒΛΗΜΑΤΑ ΠΑΝΩ ΣΤΟ ΚΕΦΑΛΑΙΟ 15

1. 'Εξι υπάλληλοι αναγνωρίζονται μόνο από τα επώνυμά τους. Κάθε υπάλληλος πληρώνει ένα συγκεκριμένο ποσό για σύνταξη, εκφραζόμενο σε ποσοστό. Τα ακόλουθα δεδομένα αντιπροσωπεύουν τους συνολικούς μισθούς και το ποσοστό για τη σύνταξη των έξι υπαλλήλων.

| | | |
|---|---|---|
| Benson | 13,800 | 6.25 |
| Hanson | 8,700 | 6.00 |
| Johnson | 10,300 | 6.25 |
| Robson | 15,000 | 7.00 |
| Thomson | 6,200 | 6.00 |
| Watson | 5,100 | 5.75 |

Γράψτε διαδικασίες για να:

- εισάγετε τα δεδομένα σε πίνακες
- να υπολογίσετε τις πραγματικές συνεισφορές για σύνταξη
- να εξάγετε τη λίστα των ονομάτων και των υπολογισμένων συνεισφορών.

Συνδέστε τις διαδικασίες μ' ένα κύριο πρόγραμμα καλλώντας τις με σειρά.

2. Γράψτε μια συνάρτηση, τη choose με δύο ορίσματα, τα range και miss. Η συνάρτηση πρέπει να επιστρέφει έναν τυχαίο ακέραιο στη δοσμένη κλίμακα range ο οποίος δεν πρέπει να είναι η τιμή της miss.

Χρησιμοποιήστε τη συνάρτηση σ' ένα πρόγραμμα που επιλέγει

ένα τυχαίο PAPER και μετά σχεδιάζει τυχαίους κύκλους με τυχαίο INK έτσι ώστε να μην είναι στο ίδιο χρώμα με το PAPER.

3. Ξαναγράψτε τη λύση της άσκησης 1 έτσι ώστε μια συνάρτηση, η pension, να παίρνει το μισθό και το ποσοστό συνεισφοράς σύνταξης σαν ορίσματα και να επιστρέφει την υπολογισμένη συνεισφορά σύνταξης. Χρησιμοποιήστε δύο διαδικασίες, μια για να εισάγετε τα δεδομένα, και μια για να εξάγετε τις απαιτούμενες πληροφορίες χρησιμοποιώντας τη συνάρτηση pension.

4. Γράψτε τα ακόλουθα:

   - μια διαδικασία που δημιουργεί μια "τράπουλα".
   - μια διαδικασία που να ανακατεύει την τράπουλα.
   - μια διαδικασία που να παίρνει έναν αριθμό σαν όρισμα και να επιστρέφει μια αλφαριθμητική τιμή περιγράφοντας το χαρτί.
   - μια διαδικασία που να "μοιράζει" και να απεικονίζει τέσσερις μοιρασιές για πόκερ από πέντε χαρτιά η κάθε μια.
   - ένα κύριο πρόγραμμα που να καλεί τις παραπάνω διαδικασίες.
   - (δες κεφάλαιο 16 για περιγραφή ενός παρόμοιου προβλήματος)

# ΚΕΦΑΛΑΙΟ 16
# ΜΕΡΙΚΕΣ ΤΕΧΝΙΚΕΣ

Σ' αυτό το τελικό κεφάλαιο παρουσιάζουμε μερικές εφαρμογές ιδεών και ευκολιών που έχουν ήδη συζητηθεί και δείχνουμε πως μπορούν να εφαρμοστούν μερικές ακόμη ιδέες.

## ΠΡΟΣΟΜΟΙΩΣΗ ΠΑΙΧΝΙΔΙΟΥ ΤΡΑΠΟΥΛΑΣ

Είναι εύκολο να αποθηκεύσετε και να χειριστήτε "τραπουλόχαρτα" αναπαριστώντας τα με τους αριθμούς 1 ως 52. Αυτός είναι ο τρόπος μετατροπής ενός τέτοιου αριθμού στο ανάλογο χαρτί. Υποθέτουμε για παράδειγμα ότι εμφανίζεται ο αριθμός 29. Θα έχετε αποφασίσει ότι:

* τα χαρτιά 1-13 είναι κούπες

* τα χαρτιά 14-26 είναι μπαστούνια

* τα χαρτιά 27-39 είναι καρρά

* τα χαρτιά 40-52 είναι σπαθιά

και θα ξέρετε ότι 29 σημαίνει πως έχετε ένα "καρρώ". Μπορείτε να προγραμματίσετε το QL γι' αυτό με:

```
LET suit = (card -1) DIV 13
```

Αυτό θα παράγει μια τιμή από 0 ως 3 που μπορείτε να τη χρησιμοποιήσετε για να τυπωθεί το κατάλληλο "χρώμα". Η τιμή μπορεί να μειωθεί στην κλίμακα 1 ως 13 γράφοντας:

```
LET value = card MOD 13
IF value = 0 THEN LET value = 13
```

## ΠΡΟΓΡΑΜΜΑ

Μπορείτε να κάνετε τους αριθμούς 1 ως 13 να τυπώνουν 'Ασσος, 2, 3... Βαλές, Ντάμα, Ρήγας ή αν το προτιμάτε μπορούν να τυπώνονται φράσεις όπως "δύο κούπα". Το ακόλουθο πρόγραμμα θα τυπώσει το όνομα του χαρτιού ανάλογα με τον αριθμό που εισάγετε.

```
100 REMark Cards
110 DIM suitname$(4,8),cardval$(13,5),
120 LET f$ = " of"
130 set_up
140 REPeat cards
150    INPUT "Enter a card number 1-52:" ! card
160    IF card <1 OR card> 52 THEN EXIT cards
170    LET suit = (card-1) DIV 13
180    LET value = card MOD 13
190    IF value = 0 THEN LET value = 13
200    PRINT cardval$(value) ! f$ ! suitname$(suit)
210 END REPeat cards
220 DEFine PROCedure set_up
230    FOR s = 1 TO 4 : READ suitname$(s)
240    FOR v = 1 TO 13 : READ cardval$(v)
250 END DEFine
260 DATA "hearts","clubs","diamonds","spades"
270 DATA "Ace","Two","Three","Four","Five","Six","Seven"
280 DATA "Eight","Nine","Ten","Jack","Queen","King"
```

**ΕΙΣΟΔΟΣ ΚΑΙ ΕΞΟΔΟΣ**

```
13
king of hearts
49
Ten of spades
27
Ace of diamonds
0
```

**ΣΧΟΛΙΟ**

Σημειώστε τη χρήση των εντολών DATA που κρατούν ένα μόνιμο αρχείο δεδομένων που χρησιμοποιεί πάντα το πρόγραμμα. Τα άλλα δεδομένα που αλλάζουν κάθε φορά που τρέχει το πρόγραμμα, εισάγονται με εντολή INPUT. Αν τα δεδομένα εισόδου ήταν γνωστά πριν από το τρέξιμο του προγράμματος τότε θα ήταν εξίσου σωστό να χρησιμοποιήσετε άλλο ένα READ και περισσότερες εντολές DATA. Αυτό θα έδινε καλύτερο έλεγχο.

**ΣΕΙΡΙΑΚΑ ΑΡΧΕΙΑ ΔΕΔΟΜΕΝΩΝ**

Το ακόλουθο πρόγραμμα θα δημιουργήσει ένα αρχείο εκατό αριθμών.

**Αριθμητικό αρχείο**

```
100 REMark Number file
110 OPEN_NEW #6,MDV1_numbers
120 FOR num = 1 TO 100
130   PRINT #6,num
140 END FOR num
150 CLOSE #6
```

Αφού τρέξετε το πρόγραμμα ελέγξτε ότι το όνομα αρχείου "numbers" υπάρχει στον κατάλογο, πληκτρολογώντας:

```
DIR MDV1_numbers
```

Μπορείτε να πάρετε μια εικόνα του αρχείου χωρίς καμμία ιδιαίτερη μορφοποίηση αντιγράφοντας από το microdrive στην οθόνη.

```
COPY MDV1_numbers to SCR_
```

Μπορείτε επίσης να χρησιμοποιήσετε το ακόλουθο πρόγραμμα για να διαβάσετε το αρχείο και να απεικονίσετε τις καταχωρήσεις του στην οθόνη.

```
100 REMark Read File
110 OPEN_IN #6,MDV1_numbers
120 FOR num = 1 TO 100
130   INPUT #6,item
140   PRINT ! item !
150 END FOR num
160 CLOSE #6
```

Αν θέλετε μπορείτε να αλλάξετε το πρόγραμμα για να πάρετε μια έξοδο διαφορετικής μορφής.

**Αρχείο χαρακτήρων**

Με όμοιο τρόπο, τα ακόλουθα προγράμματα θα δημιουργήσουν ένα αρχείο εκατό τυχαία επιλεγμένων γραμμάτων και θα τον ξαναδιαβάσουν.

```
100 REMark Letter File
110 OPEN_NEW #6,MDV1_chfile
120 FOR num = 1 TO 100
130   LET ch$ = CHR$(RND(65 TO 90))
140   PRINT #6,ch$
150 END FOR num
160 CLOSE #6
```

```
100 REMark Get Letters
110 OPEN_IN#6,MDV1_chfile
120 FOR num = 1 TO 100
130   INPUT #6,item$
140   PRINT ! item$ !
150 END FOR num
160 CLOSE #6
```

**ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΕΝΑ ΑΡΧΕΙΟ ΔΕΔΟΜΕΝΩΝ**

Υποθέτουμε ότι θέλετε να δημιουργήσετε ένα απλό αρχείο ονομάτων και τηλεφωνικών αριθμών.

| | |
|---|---|
| RON | 678462 |
| GEOFF | 896487 |
| ZOE | 249386 |
| BEN | 584621 |
| MEG | 482349 |
| CATH | 438975 |
| WENDY | 982387 |

θα το κάνει το ακόλουθο πρόγραμμα:

```
100 REMark Phone numbers
110 OPEN_NEW #6,MDV1_phone
120 FOR record = 1 TO 7
130   INPUT name$,num$
140   PRINT #6,name$,num$
150 END FOR record
160 CLOSE #6
```

Πληκτρολογήστε RUN και εισάγετε ένα όνομα ακολουθούμενο από το πλήκτρο ENTER και έναν αριθμό ακολουθούμενο από το πλήκτρο ENTER Επαναλάβετε το επτά φορές.

Σημειώστε ότι τα δεδομένα περνούν από μνήμη (buffer). Αποθηκεύονται εσωτερικά μέχρι να είναι έτοιμο το σύστημα να μεταφέρει μια ποσότητα στο microdrive. Τότε μόνο γίνεται προσπέλαση στο microdrive όπως μπορείτε να δείτε και να ακούσετε.

**ΑΝΤΙΓΡΑΦΗ ΑΡΧΕΙΟΥ**

Αφού δημιουργήσετε ένα αρχείο θα πρέπει αμέσως να τον αντιγράψετε για λόγους ασφαλείας. Για να το κάνετε αυτό, πληκτρολογήστε:

```
COPY MDV1_phone to MDV2_phone
```

**ΔΙΑΒΑΣΜΑ ΑΡΧΕΙΟΥ**

Πρέπει να είστε σίγουροι ότι το αρχείο υπάρχει σε σωστή μορφή γι αυτό διαβάστε το από το microdrive και απεικονίστε το στην οθόνη. Μπορείτε να το κάνετε εύκολα χρησιμοποιώντας:

```
COPY MDV2_phone to SCR_
```

Η έξοδος στην οθόνη δεν θα δώσει διαστήματα ανάμεσα στο όνομα και στον αριθμό αλλά θα δώσει μια "νέα γραμμή" στο τέλος κάθε καταχώρησης. Η έξοδος θα είναι:

```
RON678462
GEOFF896487
ZOE249386
BEN584621
MEG482349
CATH438975
WENDY982387
```

Μπορείτε να έχετε μια πιο ελεγχόμενη παρουσίαση των δεδομένων με το ακόλουθο πρόγραμμα:

```
100 REMark Read Phone NUmbers
110 OPEN_IN  5,MDV1_phone
120 FOR record = 1 TO 7
130   INPUT  5,rec$
140   PRINTM,rec$
150 END FOR record
160 CLOSE  5
```

Τα δεδομένα τυπώνονται όπως πριν, αλλά αυτή τη φορά κάθε ζεύγος πεδίων κρατιέται στη μεταβλητή rec$ πριν τυπωθεί στην οθόνη. 'Ετσι έχετε τη δυνατότητα να το φέρετε στην επιθυμητή μορφή.

Σημειώστε ότι περισσότερες από μια αλφαριθμητικές μεταβλητές μπορούν να χρησιμοποιηθούν στο στάδιο δημιουργίας του αρχείου με INPUT και PRINT αλλά ολόκληρη η καταχώρηση που δημιουργήθηκε έτσι μπορεί να επανέλθει από το microdrive σαν μια μοναδική αλφαριθμητική μεταβλητή (η rec$ στο παραπάνω παράδειγμα).

## ΜΙΑ ΤΑΞΙΝΟΜΗΣΗ ΕΙΣΑΓΩΓΗΣ

Τα ακόλουθα χρώματα είναι διαθέσιμα στη μορφή χαμηλής διακριτικότητας (με κωδική σειρά 0-7)

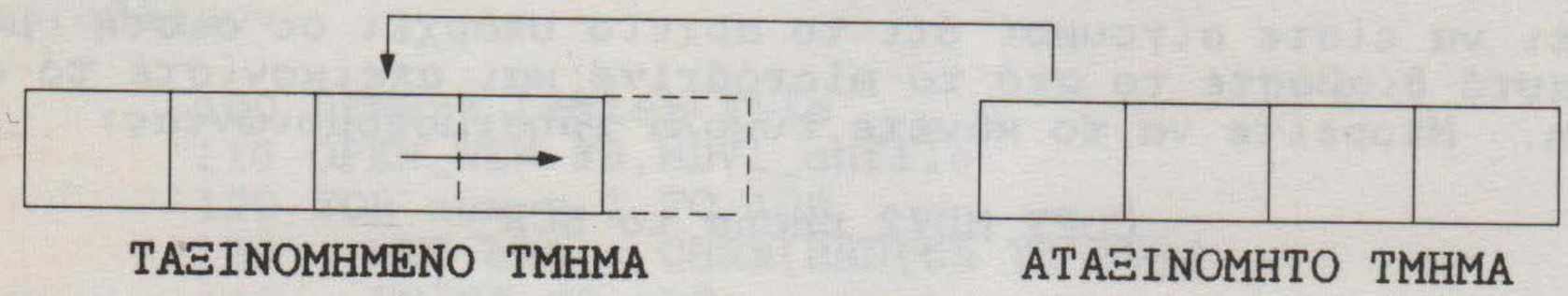
black blue red magenta green cyan yellow white

## ΠΑΡΑΔΕΙΓΜΑ

Γράψτε ένα πρόγραμμα για να ταξινομήσετε τα χρώματα με αλφαβητική σειρά χρησιμοποιώντας ταξινόμηση εισαγωγής.

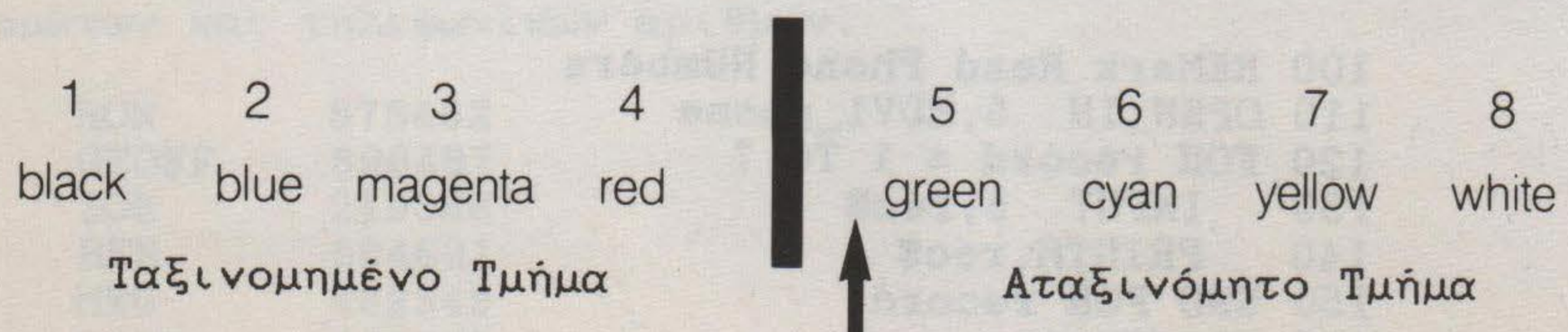
## ΜΕΘΟΔΟΣ

Τοποθετούμε τα οκτώ χρώματα σ' έναν πίνακα, τον colour$ τον οποίο διαιρούμε σε δύο μέρη:




Παίρνουμε το αριστερότερο στοιχείο του μη - ταξινομημένου τμήματος και το συγκρίνουμε με κάθε στοιχείο από δεξιά προς τα αριστερά στο ταξινομημένο τμήμα, μέχρι να βρούμε τη σωστή του θέση. Καθώς συγκρίνουμε, ανακατεύουμε το ταξινομημένο τμήμα έτσι ώστε όταν βρούμε τη σωστή θέση εισαγωγής να μπορούμε να το κάνουμε αμέσως, χωρίς άλλο ανακάτεμα.

Υποθέτουμε ότι φτάσαμε στο σημείο όπου τέσσερα στοιχεία έχουν ταξινομηθεί και τώρα δείχνουμε στο green, το αριστερώτερο στοιχείο του ταξινομημένου τμήματος.

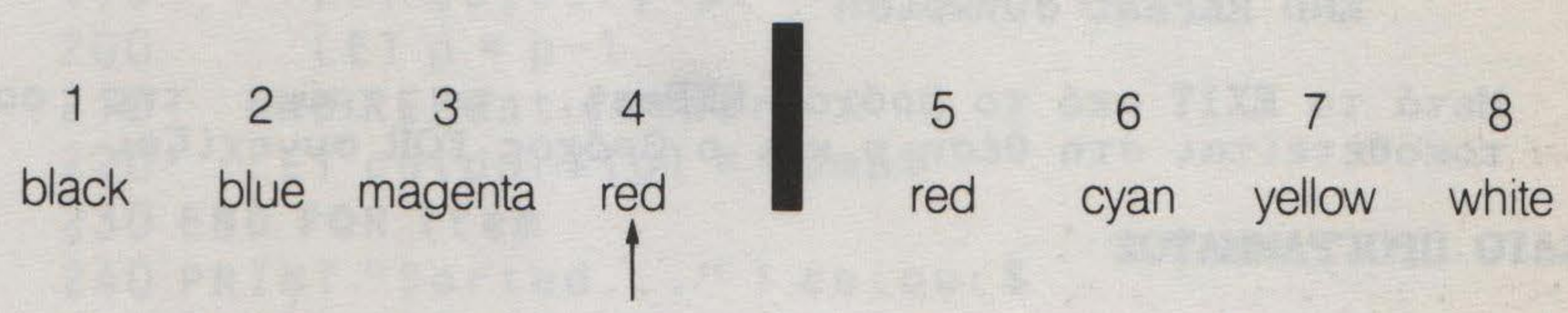


1. Τοποθετούμε το green στη μεταβλητή comp$ και δημιουργούμε μια μεταβλητή, την p με 5.

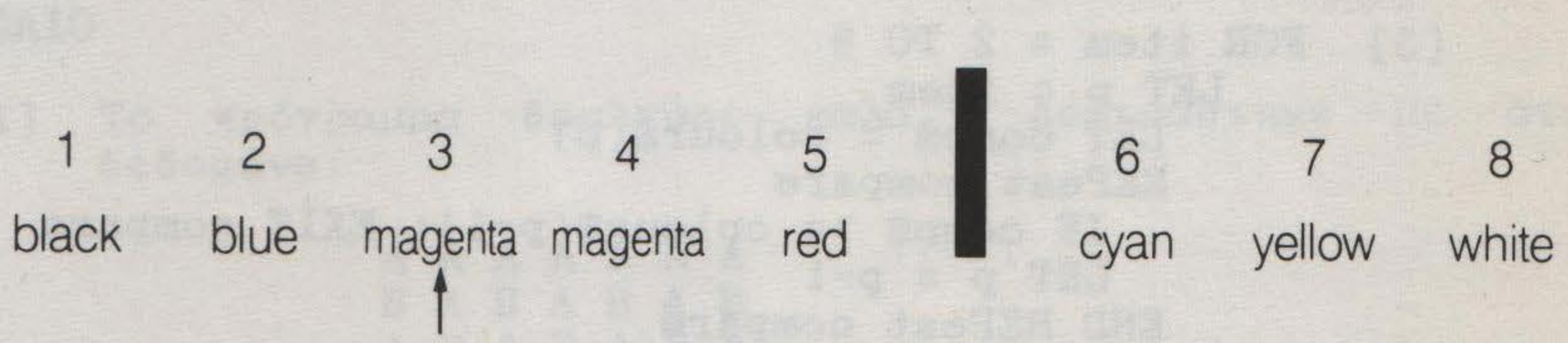
2. Η μεταβλητή p θα δείχνει τη θέση στην οποία νομίζουμε κάθε φορά πως πρέπει να πάει το green. Ξέρουμε ότι το green πρέπει να κινηθεί αριστερά γι' αυτό μειώνουμε την τιμή της p.

3. Συγκρίνουμε το green με το red. Αν το green είναι μεγαλύτερο (πιο κοντά στο Z) ή ίσο με το red τότε σταματάμε και το green μένει εκεί που είναι.

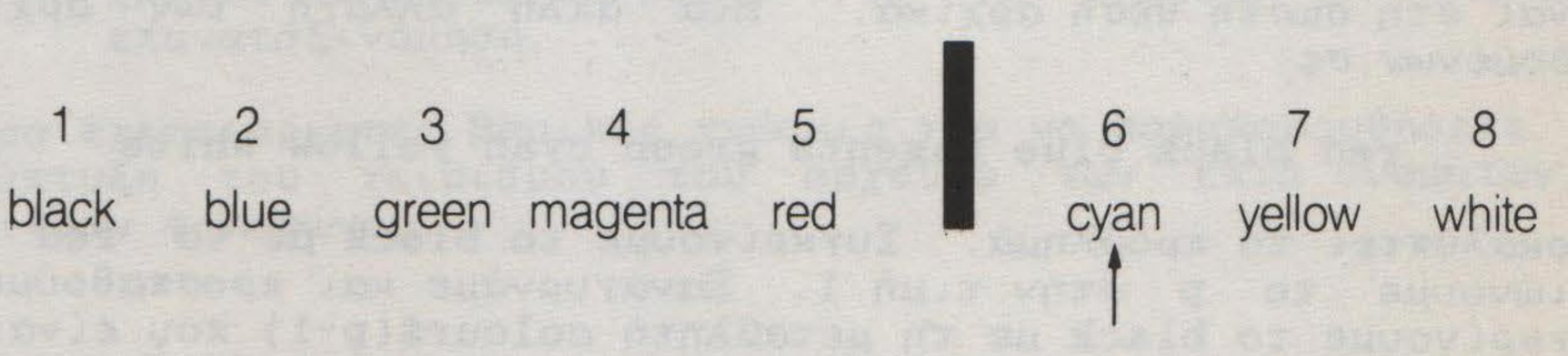
   Αλλιώς αντιγράφουμε το red στη θέση 5 και μειώνουμε την τιμή της p ώστε:




4. Τώρα επαναλαμβάνουμε τη διαδικασία αλλά αυτή τη φορά συγκρίνουμε το green με το magenta και παίρνουμε:




5. Τελικά κινούμαστε πάλι αριστερά συγκρίνοντας το green με το blue. Αυτή τη φορά δε χρειάζεται να κινηθούμε ή να αλλάξουμε τίποτα. Βγαίνουμε από το βρόχο και τοποθετούμε το green στη θέση 3. Τώρα είμαστε έτοιμοι να δείξουμε στο έκτο στοιχείο, το cyan.




**ΑΝΑΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ**

1. Πρώτα θα αποθηκεύσουμε τα χρώματα σ' έναν πίνακα, τον colour$(8) και θα χρησιμοποιήσουμε:

   comp$ το τρέχον χρώμα που συγκρίνεται

p ο δείκτης της θέσης που νομίζουμε ότι πρέπει να πάει το χρώμα στη comp$

2. 'Ενας βρόχος FOR θα εστιάσει την προσοχή μας στις θέσεις 2 ως 8 με σειρά (ένα μόνο στοιχείο είναι ήδη ταξινομημένο).

3. 'Ενας βρόχος REPeat θα κάνει τις συγκρίσεις μέχρι να βρούμε που πραγματικά πηγαίνει η τιμή της comp$.

```
REPeat compare
      IF comp$ δε χρειάζεται να πάει άλλο αριστερά
τότε EXIT
      αντέγραψε ένα χρώμα στη δεξιά διπλανή του
θέση
      και μείωσε την p
END REPeat σύγκριση
```

4. Μετά το EXIT από το βρόχο REPeat, το χρώμα της comp$ τοποθετείται στη θέση p και ο βρόχος FOR συνεχίζει.

**ΣΧΕΔΙΟ ΠΡΟΓΡΑΜΜΑΤΟΣ**

[1] Ορισμός του πίνακα colour$

[2] Ανάγνωση των χρωμάτων στον πίνακα

```
[3]  FOR item = 2 TO 8
       LET p = item
         LET comp$ = colour$(p)
         REPeat compare
           IF comp$ >= colour$(p-1): EXIT compare
           LET p = p-1
         END REPeat compare
         LET colour$(p) = comp$
     END FOR item
```

[4] PRINT sorted array colour$

[5] DATA

Το παραπέρα ψάξιμο άποκαλύπτει ένα λάθος. Συμβαίνει πολύ εύκολα αν έχουμε δεδομένα στα οποία το πρώτο στοιχείο δεν είναι στη σωστή θέση αρχικά. Μια απλή αλλαγή των αρχικών δεδομένων σε:

```
red black blue magenta green cyan yellow white
```

αποκαλύπτει το πρόβλημα. Συγκρίνουμε το black με το red και μειώνουμε το p στην τιμή 1. Ξαναγυρνάμε και προσπαθούμε να συγκρίνουμε το black με τη μεταβλητή colour$(p-1) που είναι η colour$(0) η οποία δεν υπάρχει.

Αυτό είναι ένα γνωστό πρόβλημα στους υπολογισμούς και η λύση είναι να "στείλουμε ένα φρουρό" στο τέλος του πίνακα. Αμέσως πριν μπούμε στο βρόχο REPeat χρειαζόμαστε:

```
LET colour$(0) = comp$
```

Ευτυχώς η SuperBASIC επιτρέπει μηδενικούς δείκτες, αλλιώς το πρόβλημα έπρεπει να λυθεί σε βάρος της ευαναγνωστικότητας.

**ΤΡΟΠΟΠΟΙΗΜΕΝΟ ΠΡΟΓΡΑΜΜΑ**

```
100 REM Insertion Sort
110 DIM colour$(8,7)
120 FOR item = 1 TO 8 : READ colour$(item)
130 FOR item = 2 TO 8
140   LET p = item
150   LET comp$ = colour$(p)
160   LET colour$(0) = comp$
170   REPeat compare
180     IF comp$ >= colour$(p-1) : EXIT compare
190     LET colour$(p) = colour$(p-1)
200     LET p = p-1
210   END REPeat compare
220   LET colour$(p) = comp$
230 END FOR item
240 PRINT "Sorted..." ! colour$
250 DATA "black", "blue", "magenta", "red"
260 DATA "green", "cyan", "yellow", "white"
```

**ΣΧΟΛΙΟ**

[1] Το πρόγραμμα δουλεύει καλά. Δοκιμάστηκε με άτεχνα δεδομένα:

```
A A A A A A A
B A B A B A B
A B A B A B A
B C D E F G H
G F E D C B A
```

[2] Η ταξινόμηση εισαγωγής δεν είναι ιδιαίτερα γρήγορη, αλλά μπορεί να είναι χρήσιμη για πρόσθεση μερικών στοιχείων σε μια ήδη ταξινομημένη λίστα. Είναι μερικές φορές βολικό να δίνουμε μικρά ποσά χρόνου συχνά για να έχουμε τα στοιχεία σε σειρά παρά να δίνουμε σημαντικά ποσά χρόνου λιγότερο συχνά για να κάνουμε ολοκληρωμένη επαναταξινόμηση.

Τώρα έχετε αρκετές βασικές γνώσεις για να παρακολουθήσετε την ανάπτυξη του χειρισμού του αρχείου των επτά ονομάτων και αριθμών τηλεφώνου.

## ΤΑΞΙΝΟΜΩΝΤΑΣ ΕΝΑ ΑΡΧΕΙΟ ΣΕ MICRODRIVE

Για να ταξινομήσουμε το αρχείο "phone" με αλφαβητική σειρά ονομάτων πρέπει να το διαβάσουμε σ' έναν εσωτερικό πίνακα που θα τον ταξινομήσουμε και μετά να δημιουργήσουμε ένα νέο αρχείο που θα είναι με αλφαβητική σειρά ονομάτων.

Δεν είναι καλή πρακτική να σβήνετε ένα αρχείο πριν ο αντικαταστάτης του δημιουργηθεί και αποδειχθεί σωστό. 'Ετσι πρέπει να αντιγράψετε πρώτα το αρχείο για ασφάλεια, με διαφορετικό όνομα. Η απαιτούμενη διαδικασία είναι η ακόλουθη:

[1] Αντιγραφή του αρχείου "phone" στον "phone_temp"

[2] Διάβασμα του αρχείου "phone" σ' έναν πίνακα

[3] Ταξινόμηση του πίνακα

[4] Παύση για έλεγχο ότι όλα είναι σε τάξη

[5] Σβήσιμο του αρχείου "phone"

[6] Δημιουργία νέου αρχείου "phone"

Αυτά είναι όλα όσα χρειάζεται να κάνει το πρόγραμμα αλλά το νέο αρχείο θα πρέπει να ελεγχθεί αμέσως χρησιμοποιώντας:

```
COPY MDV1_phone to SCR_
```

'Ολοι οι απαραίτητοι έλεγχοι μπορούν να γίνουν τότε.

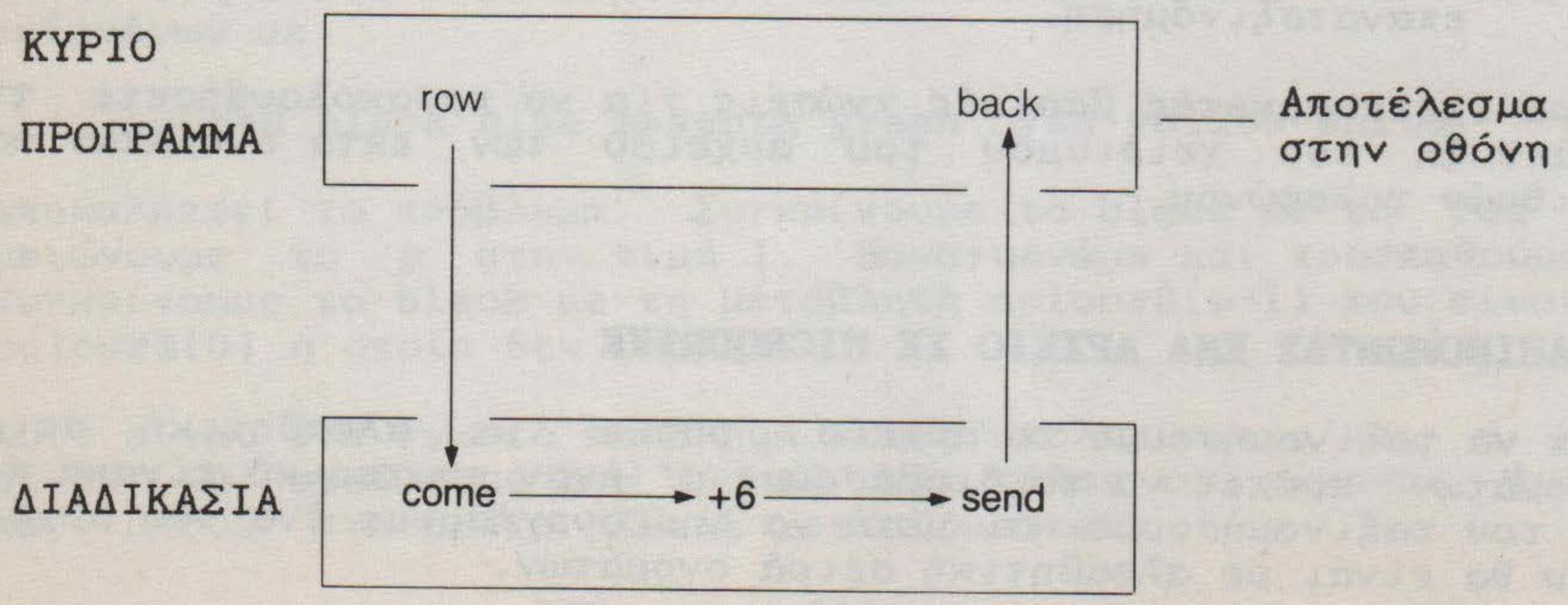
```
DELETE MDV2_phone
COPY MDV1_phone to MDV2_phone
COPY MDV1_phone to SCR_
DELETE MDV1_phone_temp
```

Οι παραπάνω λειτουργίες συμπληρώνουν τη διαδικασία της αντικατάστασης του μη - ταξινομημένου αρχείου από τον ταξινομημένο, στην κύρια έκδοση και στο αντίγραφο.

**ΠΑΡΑΜΕΤΡΟΙ ΠΙΝΑΚΩΝ**

Στο ακόλουθο πρόγραμμα επιδεικνύουμε τη μεταφορά ολόκληρων πινάκων από το κύριο πρόγραμμα στη διαδικασία και αντίστροφα. Τα δεδομένα περνούν και στις δύο κατευθύνσεις..

Στη γραμμή 130 ο πίνακας row που κρατά τους αριθμούς 1,2,3 περνά στη διαδικασία addsix. Η παράμετρος come, δέχεται τα εισερχόμενα δεδομένα και η διαδικασία προσθέτει έξι σε κάθε στοιχείο. Η παράμετρος πίνακας send, κρατά σ' αυτό το σημείο τους αριθμούς 7, 8, 9.



Αυτοί οι αριθμοί επιστρέφουν στο κύριο πρόγραμμα για να γίνουν οι τιμές του πίνακα back. Οι τιμές τυπώνονται για να δείξουν ότι τα δεδομένα μετακινήθηκαν όπως χρειαζόταν.

**ΠΡΟΓΡΑΜΜΑ**

```
100 REMark Pass Arrays
110 DIM row(3),back(3)
120 FOR k = 1 TO 3 : LET row(k) = k
130 addsix row, back
140 FOR k = 1 TO 3 : PRINT ! back(k) !
150 DEFine PROCedure addsix(come,send)
160    FOR k = 1 TO 3 : LET send(k)=come(k)+6
170 END DEFine
```

```
7 8 9
```

**'Εξοδος:**

```
7 8 9
```

Η ακόλουθη διαδικασία δέχεται έναν πίνακα που περιέχει δεδομένα για ταξινόμηση. Το μηδενικό στοιχείο θα περιέχει τον αριθμό των θεμάτων. Σημειώστε ότι δεν έχει σημασία αν ο πίνακας είναι αριθμητικός ή αλφαριθμητικός. Η αρχή του εξαναγκασμού θα μετατρέψει τα αλφαριθμητικά δεδομένα σε αριθμητικά αν χρειαστεί.

'Ενα δεύτερο ενδιαφέρον σημείο είναι ότι το στοιχείο του πίνακα, come(0) χρησιμοποιείται για δύο σκοπούς:

- μεταφέρει τον αριθμό των θεμάτων που θα ταξινομηθούν
- χρησιμοποιείται για να κρατάει το τρέχον στοιχείο που τοποθετείται.

```
100 DEFine PROCedure sort(come,send)
110    LET num = come(0)
120    FOR item = 2 TO num
130       LET p = item
140       LET come(0) = come(p)
150       REPeat compare
160          IF come(0) >= come(p-1) : EXIT compare
170          LET come(p) = come(p-1)
180          LET p = p-1
190       END REPeat compare
200       LET come(p) = come(0)
210    END FOR item
220    FOR k=1 TO 7 : send(k) = come(k)
230 END DEFine
```

```
10 REMark Test Sort
20 DIM row$(7,3),back$(7,3)
30 LET row$(0) = 7
40 FOR k = 1 TO 7 : READ row$(k)
50 sort row$,back$
60 PRINT ! back$ !
70 DATA "EEL", "DOG", "ANT", "GNU", "CAT", "BUG", "FOX"
```

```
ANT BUG CAT DOG EEL FOX GNU
```

**'Εξοδος:**

ANT BUG CAT DOG EEL FOX GNU

**ΣΧΟΛΙΟ**

Αυτό το πρόγραμμα επιδεικνύει την ευκολία χειρισμού πινάκων στη SuperBASIC. Το μόνο που χρειάζεται να κάνετε είναι να χρησιμοποιήσετε το όνομα του πίνακα για να τον περάσετε σαν παράμετρο ή για να τυπώσετε ολόκληρο τον πίνακα. 'Οταν η διαδικασία αποθηκευτεί (SAVE) τότε μπορείτε να χρησιμοποιήσετε MERGE MDV1_sort για να την προσθέσετε σ' ένα πρόγραμμα που βρίσκεται στην κύρια μνήμη.

Τώρα πια έχετε κατανοήσει αρκετά από την τεχνική και τη σύνταξη ώστε να μπορείτε να χειριστήτε ένα πιο πολύπλοκο σχέδιο οθόνης. Υποθέτουμε ότι θέλετε να παραστήσετε τις μοιρασιές τεσσάρων χαρτοπαικτών. Μια μοιρασιά αντιπροσωπεύεται από κάτι τέτοιο:

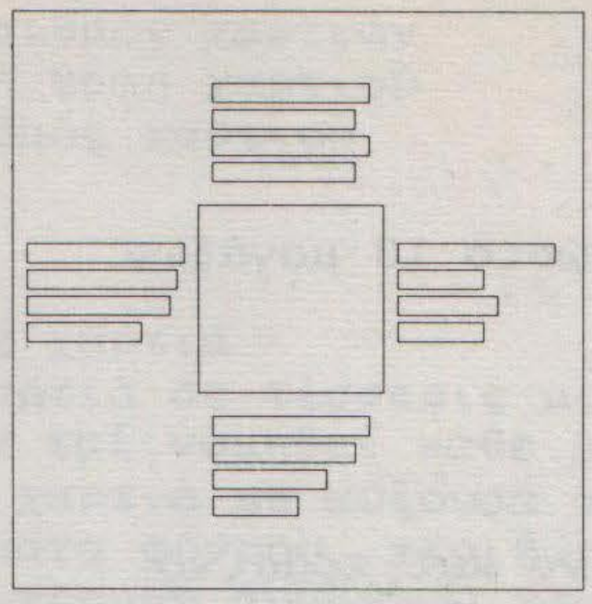
```
H: A  3 7 Q
C: 5  9 J
D: 6 10 K
S: 2  4 Q
```

Για να βοηθήσουμε την παρουσίαση, θα τυπώνουμε τις κούπες και τα καρρά σε κόκκινο, και τα μπαστούνια και τα Σπαθιά σε μαύρο. 'Ενα κατάλληλο χρώμα ταινίας (STRIP) μπορεί να είναι το άσπρο. Το γενικό φόντο μπορεί να είναι πράσινο και το τραπέζι μπορεί να είναι από χρώμα που γίνεται με ανάμιξη δύο άλλων χρωμάτων.

**ΜΕΘΟΔΟΣ**

Αφού θα χρειαστεί ένα σημαντικό ποσό τυπώματος χαρακτήρων, είναι καλύτερα να αρχίσετε να σχεδιάζετε με pixels. Μπορείτε να δείτε ότι θα χρειαστούν δώδεκα γραμμές χαρακτήρων με κάποιο διάστημα μεταξύ των γραμμών και ένα συνολικό ύψος οθόνης από 256 pixels.

Είναι χρήσιμο να ξαναθυμηθούμε ότι τα δυνατά ύψη χαρακτήρων είναι 10 pixels ή 20 pixels. Είναι προφανές ότι πρέπει να χρησιμοποιηθεί το ύψος των 10 pixels για να υπάρχει χώρος για ένα σωστό σχέδιο.



Πρέπει να προσδιοριστεί ο αριθμός των θέσεων χαρακτήρων κατά μήκος της οθόνης. Αν δεχτούμε τη συνθήκη ότι "Τ" σημαίνει "10" τότε όλες οι τιμές των χαρτιών μπορούν να παρασταθούν από απλούς χαρακτήρες. Υποθέτουμε επίσης ότι επιτρέπουμε ένα μέγιστο οκτώ χαρτιών ίδιου "χρώματος" σε πρώτη φάση. Μπορούμε να ξανασκεφτούμε το πρόβλημα ξανά αν χρειαστεί. 'Ετσι απαιτούνται συνολικά 10 χαρακτήρες για κάθε μοιρασιά. Η κατά μήκος απαίτηση είναι:

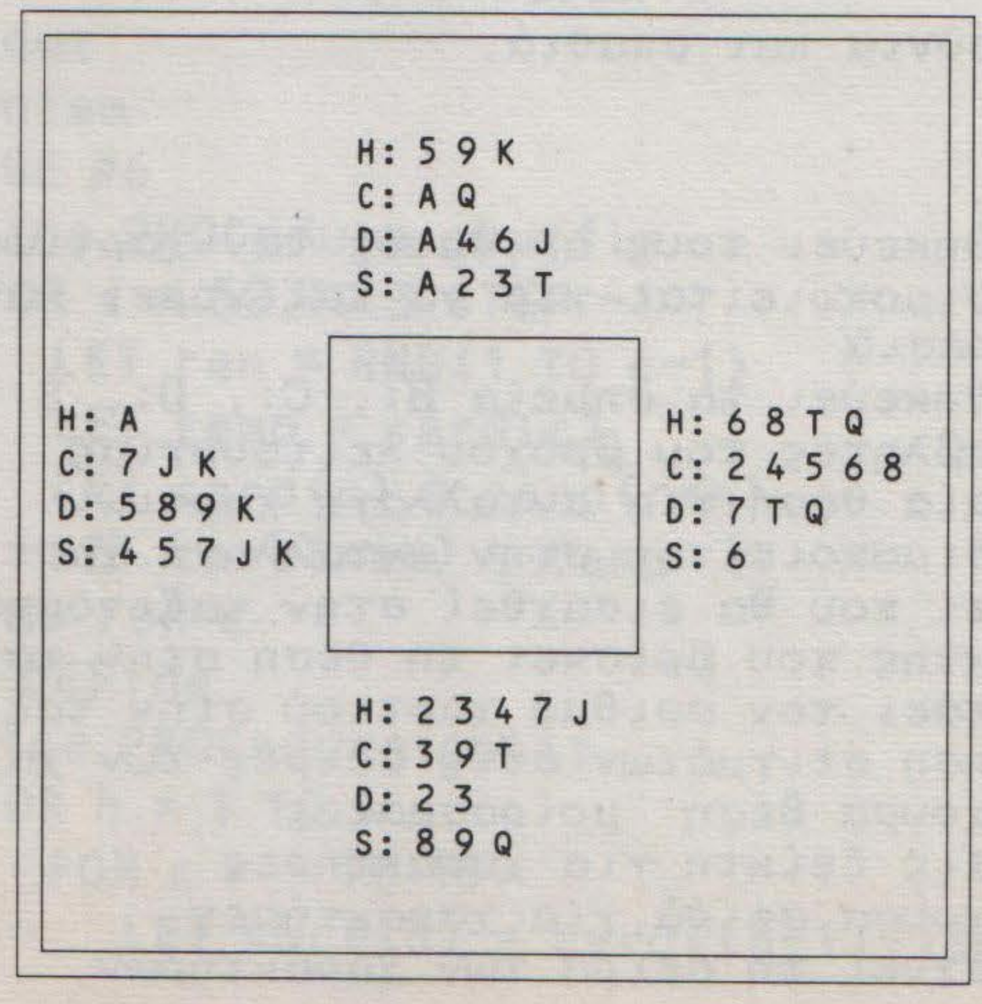
ΔΥΤΙΚΗ ΜΟΙΡΑΣΙΑ + ΠΛΑΤΟΣ ΤΡΑΠΕΖΙΟΥ + ΑΝΑΤΟΛΙΚΗ ΜΟΙΡΑΣΙΑ

Βάζοντας ένα διάστημα ανάμεσα στους χαρακτήρες αυτό θα γίνει:

20 + ΠΛΑΤΟΣ ΤΡΑΠΕΖΙΟΥ + 20

Η απόφαση τώρα εξαρτάται από το ποια μορφή οθόνης θα διαλέξετε. Η μορφή 256 θα τα βγάλει πέρα όπως θα δείτε αργότερα, αλλά πρώτα θα δουλέψουμε στη μορφή 512 pixels. Το ελάχιστο πλάτος χαρακτήρα είναι έξι pixels που δίνουν σύνολο 240 pixels + ΠΛΑΤΟΣ ΤΡΑΠΕΖΙΟΥ. Το διάγραμμα θα είναι κάπως ισορροπημένο αν έχουμε ένα πλάτος τραπεζιού περίπου στο μισό του 240.

'Ετσι πρέπει να πειραματιστούμε με πλάτος τραπεζιού περίπου 120 pixels το οποίο μπορεί να ρυθμιστεί. 'Ενας μικρός έλεγχος παρήγαγε το σχήμα που φαίνεται:



**ΠΑΡΑΘΥΡΟ**

440 χ 220 στο 35,15

Πράσινο με μαύρο περιθώριο 10 μονάδων

**ΤΡΑΠΕΖΙ**

100 χ 60 στο 150,60

σχέδιο σκακιέρας κόκκινο και πράσινο

**ΜΟΙΡΑΣΙΕΣ**

Οι αρχικές θέσεις του δείκτη (CURSOR) είναι:

| | |
|---|---|
| ΒΟΡΡΑΣ | 150,10 |
| ΑΝΑΤΟΛΗ | 260,60 |
| ΝΟΤΟΣ | 150,130 |
| ΔΥΣΗ | 30,60 |

**ΜΕΓΕΘΟΣ ΧΑΡΑΚΤΗΡΩΝ**

Το κανονικό της MODE 512

**ΑΡΙΘΜΟΣ PIXELS**

ανάμεσα στις γραμμές είναι 12

**ΧΡΩΜΑ ΧΑΡΑΚΤΗΡΩΝ**

άσπρο

**ΤΑΙΝΙΑ ΧΑΡΑΚΤΗΡΩΝ**

Κόκκινη για κούπες και καρρά

Μαύρη για μπαστούνια και σπαθιά.

**ΜΕΤΑΒΛΗΤΕΣ**

| | |
|---|---|
| **card(52)** | αποθηκεύει τους αριθμούς των χαρτιών |
| **sort(13)** | χρησιμοποιείται για να ταξινομεί κάθε μοιρασιά |
| **tok$(4,2)** | αποθηκεύει τα σημεία H:, C:, D:, S: |
| **kmcmh** | μεταβλητές του βρόχου λειτουργίας |
| **ran** | τυχαία θέση για ανταλλαγή χαρτιών |
| **temp** | χρησιμοποιείται στην ανταλλαγή χαρτιών |
| **item** | χαρτί που θα εισαχθεί στην ταξινόμηση |
| **dart** | δείκτης που βρίσκει τη θέση στην ταξινόμηση |
| **comp** | κρατάει τον αριθμό χαρτιού στην ταξινόμηση |
| **inc** | αύξηση στιγμάτων στις σειρές των χαρτιών |
| **seat** | τρέχουσα θέση "μοιράσματος" |
| **ac,dn,** | θέσεις δείκτη για χαρακτήρες |
| **row** | τρέχουσα σειρά για χαρακτήρες |
| **lin$** | φτιάχνει τη σειρά των χαρακτήρων |

| | |
|---|---|
| **max** | μέγιστος αριθμός χαρτιών |
| **p** | δείχνει στη θέση χαρτιού |
| **n** | τρέχων αριθμός χαρτιού |

**ΔΙΑΔΙΚΑΣΙΕΣ**

| | |
|---|---|
| **shuffle** | ανακατεύει 52 χαρτιά |
| **split** | χωρίζει τα χαρτιά σε τέσσερις μοιρασιές και καλεί sortem για να ταξινομηθεί κάθε μοιρασιά |
| **sortem** | ταξινομεί 13 χαρτιά με αύξουσα σειρά |
| **layout** | δίνει τα χρώματα φόντου, περιθώριου και τραπεζιού |
| **printem** | τυπώνει κάθε γραμμή συμβόλων χαρτιών |
| **getline** | παίρνει μια σειρά χαρτιών και μετατρέπει τους αριθμούς στα σύμβολα A,2,3,4,5,66,7,8,9,T,J,Q,K |

**ΠΕΡΙΛΗΨΗ ΣΧΕΔΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ**

[1] Δήλωση πινάκων, λήψη συμβόλων και τοποθέτηση 52 αριθμών στον πίνακα card.

[2] Ανακάτεμα χαρτιών.

[3] Χωρισμός σε 4 μοιρασιές και ταξινόμηση κάθε μιας.

[4] OPEN το παράθυρο της οθόνης.

[5] Κατασκευή του σχέδιου της οθόνης.

[6] Τύπωμα των τεσσάρων μοιρασιών.

[7] CLOSE το παράθυρο της οθόνης.

```
100 DIM card(52),sort(13),tok$(4,2)
110 FOR k = 1 TO 4 : READ tok$(k)
120 FOR k = 1 TO 52 : LET card(k) = k
130 shuffle
140 split
150 OPEN #6,scr_440x220a35x15
160 layout
170 printem
180 CLOSE #6
190 DEFine PROCedure shuffle
200    FOR c = 52 TO 3 STEP -1
210       LET ran = RND(1 TO c-1)
220       LET temp = card(c)
230       LET card(c) = card(ran)
240       LET card(ran) = temp
250    END FOR c
260 END DEFine
270 DEFine PROCedure split
280    FOR h = 1 TO 4
290       FOR c = 1 TO 13
300          LET sort(c) = card((h-1)*13+c)
```

```
310       END FOR c
320       sortem
330       FOR c = 1 TO 13
340          LET card((h-1)*13+c) = sort(c)
350       END FOR c
360    END FOR h
370 END DEFine
380 DEFine PROCedure sortem
390    FOR item = 2 TO 13
400       LET dart = item
410       LET comp = sort(dart)
420       LET sort(0) = comp
430       REPeat compare
440          IF comp >= sort(dart-1) : EXIT compare
450          LET sort(dart) = sort(dart-1)
460          LET dart = dart -1
470       END REPeat compare
480       LET sort(dart) = comp
490    END FOR item
500 END DEFine
510 DEFine PROCedure layout
520    PAPER #6,4 : CLS #6
530    BORDER #6,10,0
540    BLOCK #6,100,60,150,60,2,4
550 END DEFine
560 DEFine PROCedure printem
570    LET inc = 12 : INK #6,7
580    LET p = 0
590    FOR seat = 1 TO 4
600       READ ac,dn
610       FOR row = 1 TO 4
620          getline
630          CURSOR #6,ac,dn
640          PRINT #6,lin$
650          LET dn = dn + inc
660       END FOR row
670    END FOR seat
680 END DEFine
690 DEFine PROCedure getline
700    IF row MOD 2 = 0 THEN STRIP #6,0
710    IF row MOD 2 = 1 THEN STRIP #6,2
720    LET lin$ = tok$(row)
730    LET max = row*13
740    REPeat one_suit
750       LET p = p + 1
760       LET n = card(p)
```

```
770       IF n >max THEN p = p-1 : EXIT one_suit
780       LET n = n MOD 13
790       IF n = 0 THEN n = 13
800       IF n = 1 : LET ch$ = "A"
810       IF n >= 2 AND n <= 9 : LET ch$ = n
820       IF n = 10 : LET ch$ = "T"
830       IF n = 11 : LET ch$ = "J"
840       IF n = 12 : LET ch$ = "Q"
850       IF n = 13 : LET ch$ = "K"
860       LET lin$ = lin$ & " " & ch$
870       IF p = 52 : EXIT one=suit
880       IF card(p)>card(p+1) : EXIT one_suit
890    END REPeat one_suit
900 END DEFine
910 DATA "H:","C:","D:","S:"
920 DATA 150,10,260,60,150,130,30,60
```

**ΣΧΟΛΙΟ**

Το πρόγραμμα δουλεύει στη MODE 256 αλλά οι διάφορες γραμμές των συμβόλων χαρτιών μπορεί να επικαλύπτουν το "τραπέζι" ή να υπερχειλίζουν στην άκρη του παραθύρου. Μια απλή αλλαγή στη διαδικασία getline από:

```
1270 LET lin$ = lin$ & " " ch$
```

σε:

```
1270 LET lin$ = lin$ & ch$
```

θα το διωρθώσει αυτό. Τα διαστήματα μεταξύ των χαρακτήρων εξαφανίζονται αλλά το μεγαλύτερο μέγεθος χαρακτήρων κάνει τις σειρές πιο ευανάγνωστες. Επομένως το πρόγραμμα δουλεύει καλά και στις δύο μορφές γραφικών.

**ΕΠΙΛΟΓΟΣ**

Προσπαθήσαμε να σας δείξουμε πως μπορείτε να χρησιμοποιήσετε τη SuperBASIC για επίλυση προβλημάτων. Δείξαμε πως οι απλές αποστολές μπορούν να εκτελεστούν με απλούς τρόπους. 'Οταν η αποστολή είναι εκ του φυσικού της πολύπλοκη όπως ο χειρισμός πινάκων ή η σχεδίαση γραφικών στην οθόνη, η SuperBASIC σας επιτρέπει να τις χειριστήτε αποδοτικά με τη μέγιστη δυνατή σαφήνεια.

Αν ήσαστε αρχάριος και μελετήσατε ένα μεγάλο μέρος αυτού του οδηγού τότε αρχίσατε καλά στην πορεία σας προς το σωστό προγραμματισμό. Αν είσαστε ήδη έμπειρος τότε ελπίζουμε ότι θα εκτιμήσετε και θα εκμεταλευτείτε τα επιπλέον χαρακτηριστικά που προσφέρει η SuperBASIC.

Η κλίμακα των αποστολών που μπορούν να πραγματοποιηθούν με τη SuperBASIC είναι τόσο τεράστια ώστε μπορέσαμε να αγγίξουμε μόνο ένα κλάσμα τους σ' αυτόν τον οδηγό. Δεν μπορούμε να μαντέψουμε ποιες από τις χιλιάδες δυνατότητες θα επιχειρήσετε αλλά ευχόμαστε να τις βρείτε όλες καρποφόρες, προκλητικές και διασκεδαστικές.

# QL

## Έννοιες

## 'Εννοιες

Ο Οδηγός Αναφοράς Εννοιών περιγράφει τις έννοιες που σχετίζονται με τη SuperBASIC καιτ τα μηχανήματα του QL. Θα πρέπει να σκέφτεστε τον Οδηγό Αναφοράς Εννοιών σαν μια πηγή πληροφοριών. Αν υπάρχουν κάποιες ερωτήσεις για τη SuperBASIC ή το ίδιο το QL που δημιουργούνται από τη χρήση του υπολογιστή, ή άλλων τμημάτων του εγχειριδίου, τότε ο Οδηγός Αναφοράς Εννοιών, μπορεί να δώσει μια απάντηση. Οι έννοιες καταγράφονται με αλφαβητική σειρά χρησιμοποιώντας την πιο πιθανή ονομασία για κάθε έννοια. Αν δεν μπορεί να βρεθεί το αντικείμενο, τότε συμβουλευτείτε το ευρετήριο το οποίο θα σας σε ποια σελίδα θα πρέπει να δείτε.

'Οταν ένα παράδειγμα καταγράφεται με αριθμούς σειρών, τότε είναι ένα ολόκληρο πρόγραμμα και μπορεί να εισαχθεί και να τρέξει. Τα παραδείγματα που καταγράφονται χωρίς αριθμούς είναι συνήθως απλές εντολές και ίσως δε θα πρέπει να τις εισάγετε στον υπολογιστή ξεχωριστά. Τα παραδείγματα που περιέχουν κουκίδες δεν θα δουλέψουν σωστά σε μία τηλεόραση.

## Μεταβλητές με δείκτες (arrays)

Στις μεταβλητές με δείκτες θα πρέπει να δίνονται διαστάσεις προτού χρησιμοποιηθούν. 'Οταν σε μία μεταβλητή με δείκτες έχουν δοθεί διαστάσεις η τιμή καθενός από τα στοιχεία της τοποθετείται στο μηδέν ή σε μια αλφαριθμητική σταθερά μήκους μηδέν αν είναι μια αλφαριθμητική μεταβλητή με δείκτες. Μια διάσταση μεταβλητής με δείκτη τρέχει από το μηδέν ως τη συγκεκριμένη τιμή. Δεν υπάρχει όριο στον αριθμό των διαστάσεων που μπορεί να καθοριστεί εκτός από τη συνολική χωρητικότητα της μνήμης του υπολογιστή. Μια μεταβλητή με δείκτες φυλάσσεται έτσι ώστε ο τελευταίος δείκτης που καθορίστηκε να μεταβάλλεται όσο γίνεται πιο γρήγορα:

**ΠΑΡΑΔΕΙΓΜΑ**

Η μεταβλητή με δείκτες που καθορίστηκε σαν:

```
DIM array(2,4)
```

θα φυλαχτεί σαν

```
0,0       low address
0,1
0,2
0,3
0,4
1,0
1,1
1,2
1,3
1,4
2,0
2,1
2,2
2,3
2,4       high address
```

Το στοιχείο που αναφέρεται από το array(a,b,c) είναι ίσο με το στοιχείο που αναφέρεται από το array(a)(b)(c).

| Εντολή | Λειτουργία |
|---|---|
| DIM | δίνονται διαστάσεις σε μία μεταβλητή με δείκτες |
| DIMN | βρίσκονται οι διαστάσεις μιας μεταβλητής με δείκτες |

## BASIC

Η SuperBASIC περιέχει τις περισσότερες από τις συναρτήσεις, διαδικασίες που βρίσκονται σε άλλες διαλέκτους της BASIC. Πολλές από αυτές τις συναρτήσεις είναι περιττές στη SuperBASIC αλλά περιλαμβάνονται για λόγους συμβατότητας.

| | |
|---|---|
| GOTO | Χρησιμοποιήστε IF, REPeat, κ.λ.π.tc |
| GOSUB | Χρησιμοποιήστε DEFine PROCedure |
| ON GOTO | Χρησιμοποιήστε SELect |
| ON GOSUB | Χρησιμοποιήστε SELect |

Μερικές διαταγές δεν υπάρχουν. Μπορούν πάντα να πραγματοποιούνται χρησιμοποιώντας μια πιο γενική συνάρτηση. Για παράδειγμα, δεν υπάρχουν οι διαταγές LPRINT και LLIST στη SuperBASIC αλλά το εξαγόμενο μπορεί να κατευθυνθεί σε έναν εκτυπωτή ανοίγοντας το σχετικό κανάλι και χρησιμοποιώντας το PRINT ή το LIST.

| | |
|---|---|
| LPRINT | Χρησιμοποιήστε το PRINT# |
| LLIST | Χρησιμοποιήστε το LIST# |
| VAL | Δεν απαιτείται στη SuperBASIC |
| STR$ | Δεν απαιτείται στη SuperBASIC |

| | |
|---|---|
| IN | Δεν εφαρμόζεται στον 68008 |
| OUT | Δεν εφαρμόζεται στον 68008 |

**ΣΧΟΛΙΟ**

Σχεδόν όλα τα είδη της BASIC χρειάζονται τις εντολές VAL(x$) και STR$(x) για να μπορούν να μετατρέψουν την εσωτερική αντιπροσώπευση της αξίας μιας αλφαριθμητικής μεταβλητής σε μια αριθμητική έκφραση.

Αυτές οι συναρτήσεις δε χρειάζονται στη SuperBASIC γιατί υπάρχουν άλλες δυνατότητες που αναφέρονται σαν "coercion".

## break

Αν σε κάποια στιγμή, ο υπολογιστής αποτύχει να απαντήσει ή επιθυμείτε να διακόψετε ένα SuperBASIC πρόγραμμα τότε:

πιέστε το

CTRL

και μετά το

SPACE

'Ενα πρόγραμμα που έχει διακοπεί κατ' αυτόν τον τρόπο μπορεί να ξαναρχίσει χρησιμοποιώντας την εντολή CONTINUE.

## Κανάλια (Channels)

'Ενα κανάλι είναι ένας τρόπος με τον οποίο δεδομένα μπορούν να εξαχθούν ή να εισαχθούν από μια συσκευή του QL. 'Ενα κανάλι μπορεί να χρησιμοποιηθεί αφού πρώτα ενεργοποιηθεί με την εντολή OPEN. Ορισμένα κανάλια θα πρέπει να μένουν πάντα ανοικτά. Αυτά είναι τα default κανάλια και είναι η απλή επικοινωνία με το QL διαμέσου του πληκτρολογίου και της οθόνης. 'Οταν ένα κανάλι δεν είναι πια σε χρήση μπορεί να αποενεργοποιηθεί με την εντολή CLOSE.

'Ενα κανάλι αναγνωρίζεται από έναν αριθμό. 'Ενας αριθμός καναλιού είναι μια αριθμητική έκφραση στην οποία προηγείται ένα σύμβολο #. 'Οταν το κανάλι ανοίγεται, μια συσκευή συνδέεται σε έναν αριθμό καναλιού και το κανάλι αποκτά έναν αριθμό. Απ 'δω και πέρα, το κανάλι αναγνωρίζεται μόνο από το δικό του αριθμό καναλιού. Για παράδειγμα η εντολή:

```
OPEN #5,SER1
```

θα συνδέσει τη σειριακή πόρτα 1 με τον αριθμό καναλιού 5. 'Οταν το κανάλι κλείνεται χρειάζεται μόνο να συγκεκριμενοποιηθεί ο αριθμός καναλιού. Για παράδειγμα:

```
CLOSE #5
```

Ανοίγοντας ένα κανάλι απαιτείται η ενεργοποίηση του οδηγού συσκευής για εκείνο το κανάλι. Συνήθως υπάρχουν περισσότεροι

από ένας τρόποι για την ενεργοποίηση του οδηγού συσκευής. Για παράδειγμα, το δίκτυο απαιτεί έναν αριθμό σταθμού. Αυτή η επί πλέον πληροφορία προστίθεται στο όνομα της συσκευής και περνιέται στην εντολή OPEN σαν μια παράμετρος. (Δείτε στην έννοια συσκευή και περιφερειακή επέκταση).

Τα δεδομένα μπορούν να εξαχθούν σε ένα κανάλι πληκτρολογώντας τα σ' αυτό. Αυτός είναι ο ίδιος μηχανισμός με τον οποίο εμφανίζονται τα εξαγόμενα στην οθόνη του QL, το PRINT χωρίς καμμία παράμετρο εξάγει στο default κανάλι #1. Για παράδειγμα οι εντολές:

```
10 OPEN #5, mdv1_test_file
20 PRINT #5, "this text is in file test_file"
30 CLOSE #5
```

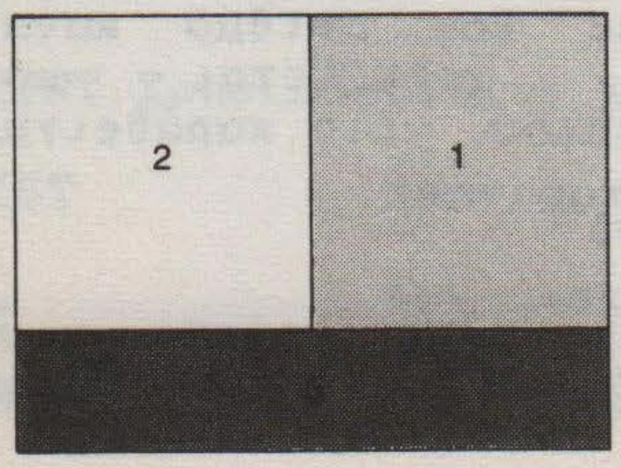
θα εξάγουν το κείμενο "this text is in file test_file" στο αρχείο test_file. Είναι σπουδαίο να κλείσετε το αρχείο αφού έχουν ολοκληρωθεί όλες οι είσοδοι για να είστε σίγουροι ότι έχουν γραφτεί όλα τα δεδομένα.

Δεδομένα μπορούν να εισαχθούν από ένα αρχείο με ανάλογο τρόπο χρησιμοποιώντας το INPUT. Επίσης μπορούν να εισαχθούν από ένα κανάλι, βγάζοντας ένα χαρακτήρα τη φορά χρησιμοποιώντας το INKEY$.

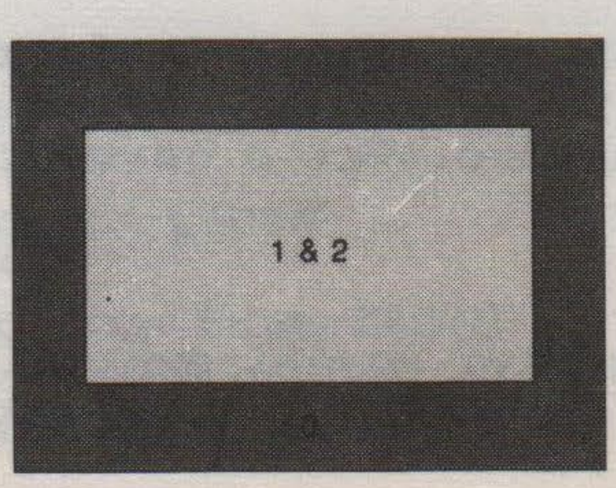
'Ενα κανάλι μπορεί να ανοιχθεί σαν ένα κανάλι κονσόλας. Τα εξαγόμενα κατευθύνονται σε ένα συγκεκριμένο παράθυρο στην οθόνη του QL και τα εισαγόμενα παίρνονται από το πληκτρολόγιο του QL. 'Οταν ανοίγεται ένα κανάλι κονσόλας συγκεκριμενοποιείται το μέγεθος και το σχήμα του αρχικού παράθυρου. Αν είναι ενεργά περισσότερα από ένα κανάλια κονσόλας, τότε είναι πιθανό, περισσότερα από ένα κανάλια να απαιτούν εισαγωγή, την ίδια στιγμή. Σε αυτή την περίπτωση, το απαιτούμενο κανάλι μπορεί να διαλεχτεί πατώντας το CTRL και το C για να ανακυκλώσετε τα κανάλια που περιμένουν. Ο δρομέας στο παράθυρο του επιλεγμένου καναλιού θα αρχίσει να αναβοσβήνει.

Το QL έχει τρία default κανάλια που ανοίγονται αυτόματα. Καθένα από αυτά τα κανάλια συνδέεται με ένα παράθυρο στην οθόνη του QL:

- κανάλι 0 - εντολή και λάθους κανάλι
- κανάλι 1 - εξαγόμενο και κανάλι γραφικών
- κανάλι 2 - κανάλι λίστας προγράμματος




Μόνιτορ




Τηλεόραση

| Εντολή | Λειτουργία |
|---|---|
| OPEN | ανοίγει ένα κανάλι για I/O |
| CLOSE | κλείνει ένα προηγούμενα ανοικτό κανάλι |
| PRINT | εξάγει σε ένα κανάλι |
| INPUT | εισάγει από ένα κανάλι |
| INKEY$ | εισάγει ένα χαρακτήρα από ένα κανάλι |

## Σετ χαρακτήρων και πλήκτρων

Οι έλεγχοι του δρομέα δεν έχουν ενσωματωθεί μέσα στο λειτουργικό σύστημα: παρ' όλα αυτά, αυτές οι λειτουργίες πρόκειται να χρησιμοποιηθούν σε εφαρμογές από το software, θα πρέπει να χρησιμοποιηθούν τα καθορισμένα πλήκτρα: Επίσης τα πλήκτρα αυτά κανονικά δε θα πρέπει να χρησιμοποιηθούν για οποιοδήποτε άλλο σκοπό.

| Δεκαεξαδικό | Hex | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 0 | 00 | CTRL £ | NULL |
| 1 | 01 | CTRL A | |
| 2 | 02 | CTRL B | |
| 3 | 03 | CTRL C | change input channel (see note) |
| 4 | 04 | CTRL D | |
| 5 | 05 | CTRL E | |
| 6 | 06 | CTRL F | |
| 7 | 07 | CTRL G | |
| 8 | 08 | CTRL H | |
| 9 | 09 | TAB (CTRL I) | Next field |
| 10 | 0A | ENTER (CTRL J) | New line/Command entry |
| 11 | 0B | CTRL K | |
| 12 | 0C | CTRL L | |
| 13 | 0D | CTRL M | Enter |
| 14 | 0E | CTRL N | |
| 15 | 0F | CTRL O | |
| 16 | 10 | CTRL P | |
| 17 | 11 | CTRL Q | |
| 18 | 12 | CTRL R | |
| 19 | 13 | CTRL S | |
| 20 | 14 | CTRL T | |
| 21 | 15 | CTRL U | |
| 22 | 16 | CTRL V | |
| 23 | 17 | CTRL W | |
| 24 | 18 | CTRL X | |
| 25 | 19 | CTRL Y | |
| 26 | 1A | CTRL Z | |
| 27 | 1B | ESC (CTRL SHIFT \|) | Abort current level of command |
| 28 | 1C | CTRL SHIFT \ | |
| 29 | 1D | CTRL SHIFT ] | |
| 30 | 1E | CTRL SHIFT £ | |

| Δεκαεξαδικό Hex | | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 31 | 1F | CTRL SHIFT ESC | |
| 32 | 20 | Space | Space |
| 33 | 21 | SHIFT 1 | ! |
| 34 | 22 | SHIFT ' | " |
| 35 | 23 | SHIFT 3 | # |
| 36 | 24 | SHIFT 4 | $ |
| 37 | 25 | SHIFT 5 | % |
| 38 | 26 | SHIFT 7 | & |
| 39 | 27 | ' | ' |
| 40 | 28 | SHIFT 9 | ( |
| 41 | 29 | SHIFT 0 | ) |
| 42 | 2A | SHIFT 8 | * |
| 43 | 2B | SHIFT = | + |
| 44 | 2C | , | , |
| 45 | 2D | - | - |
| 46 | 2E | . | . |
| 47 | 2F | / | / |
| 48 | 30 | 0 | 0 |
| 49 | 31 | 1 | 1 |
| 50 | 32 | 2 | 2 |
| 51 | 33 | 3 | 3 |
| 52 | 34 | 4 | 4 |
| 53 | 35 | 5 | 5 |
| 54 | 36 | 6 | 6 |
| 55 | 37 | 7 | 7 |
| 56 | 38 | 8 | 8 |
| 57 | 39 | 9 | 9 |
| 58 | 3A | SHIFT ; | : |
| 59 | 3B | ; | ; |
| 60 | 3C | SHIFT , | < |
| 61 | 3D | = | = |
| 62 | 3E | SHIFT . | > |
| 63 | 3F | SHIFT / | ? |
| 64 | 40 | SHIFT 2 | @ |
| 65 | 41 | SHIFT A | A |
| 66 | 42 | SHIFT B | B |
| 67 | 43 | SHIFT C | C |
| 68 | 44 | SHIFT D | D |
| 69 | 45 | SHIFT E | E |
| 70 | 46 | SHIFT F | F |
| 71 | 47 | SHIFT G | G |
| 72 | 48 | SHIFT H | H |
| 73 | 49 | SHIFT I | I |
| 74 | 4A | SHIFT J | J |
| 75 | 4B | SHIFT K | K |
| 76 | 4C | SHIFT L | L |

| Δεκαεξαδικό | Hex | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 77 | 4D | SHIFT M | M |
| 78 | 4E | SHIFT N | N |
| 79 | 4F | SHIFT O | O |
| 80 | 50 | SHIFT P | P |
| 81 | 51 | SHIFT Q | Q |
| 82 | 52 | SHIFT R | R |
| 83 | 53 | SHIFT S | S |
| 84 | 54 | SHIFT T | T |
| 85 | 55 | SHIFT U | U |
| 86 | 56 | SHIFT V | V |
| 87 | 57 | SHIFT W | W |
| 88 | 58 | SHIFT X | X |
| 89 | 59 | SHIFT Y | Y |
| 90 | 5A | SHIFT Z | Z |
| 91 | 5B | \| | \| |
| 92 | 5C | \ | \ |
| 93 | 5D | \| | \| |
| 94 | 5E | SHIFT 6 | ^ |
| 95 | 5F | SHIFT - | — |
| 96 | 60 | £ | £ |
| 97 | 61 | A | a |
| 98 | 62 | B | b |
| 99 | 63 | C | c |
| 100 | 64 | D | d |
| 101 | 65 | E | e |
| 102 | 66 | F | f |
| 103 | 67 | G | g |
| 104 | 68 | H | h |
| 105 | 69 | I | i |
| 106 | 6A | J | j |
| 107 | 6B | K | k |
| 108 | 6C | L | l |
| 109 | 6D | M | m |
| 110 | 6E | N | n |
| 111 | 6F | O | o |
| 112 | 70 | P | p |
| 113 | 71 | Q | q |
| 114 | 72 | R | r |
| 115 | 73 | S | s |
| 116 | 74 | T | t |
| 117 | 75 | U | u |
| 118 | 76 | V | v |
| 119 | 77 | W | w |
| 120 | 78 | X | x |
| 121 | 79 | Y | y |
| 122 | 7A | Z | z |

| Δεκαεξαδικό | Hex | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 123 | 7B | SHIFT \| | { |
| 124 | 7C | SHIFT \ | \| |
| 125 | 7D | SHIFT ] | } |
| 126 | 7E | SHIFT £ | ~ |
| 127 | 7F | SHIFT ESC | © |
| 128 | 80 | CTRL ESC | ä |
| 129 | 81 | CTRL SHIFT 1 | ã |
| 130 | 82 | CTRL SHIFT ' | å |
| 131 | 83 | CTRL SHIFT 3 | é |
| 132 | 84 | CTRL SHIFT 4 | ö |
| 133 | 85 | CTRL SHIFT 5 | õ |
| 134 | 86 | CTRL SHIFT 7 | ç |
| 135 | 87 | CTRL ' | ü |
| 136 | 88 | CTRL SHIFT 9 | å |
| 137 | 89 | CTRL SHIFT 0 | ñ |
| 138 | 8A | CTRL SHIFT 8 | œ |
| 139 | 8B | CTRL SHIFT = | ¿ |
| 140 | 8C | CTRL , | á |
| 141 | 8D | CTRL - | à |
| 142 | 8E | CTRL . | â |
| 143 | 8F | CTRL / | ë |
| 144 | 90 | CTRL 0 | è |
| 145 | 91 | CTRL 1 | ê |
| 146 | 92 | CTRL 2 | ï |
| 147 | 93 | CTRL 3 | í |
| 148 | 94 | CTRL 4 | ì |
| 149 | 95 | CTRL 5 | î |
| 150 | 96 | CTRL 6 | ó |
| 151 | 97 | CTRL 7 | ò |
| 152 | 98 | CTRL 8 | ô |
| 153 | 99 | CTRL 9 | ú |
| 154 | 9A | CTRL SHIFT ; | ù |
| 155 | 9B | CTRL ; | û |
| 156 | 9C | CTRL SHIFT , | ß |
| 157 | 9D | CTRL = | ¢ |
| 158 | 9E | CTRL SHIFT . | ¥ |
| 159 | 9F | CTRL SHIFT / | ` |
| 160 | A0 | CTRL SHIFT 2 | Ä |
| 161 | A1 | CTRL SHIFT A | Ã |
| 162 | A2 | CTRL SHIFT B | Å |
| 163 | A3 | CTRL SHIFT C | É |
| 164 | A4 | CTRL SHIFT D | Ö |
| 165 | A5 | CTRL SHIFT E | Õ |
| 166 | A6 | CTRL SHIFT F | Ç |
| 167 | A7 | CTRL SHIFT G | Ü |
| 168 | A8 | CTRL SHIFT H | Å |

| Δεκαεξαδικό | Hex | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 169 | A9 | CTRL SHIFT I | Ñ |
| 170 | AA | CTRL SHIFT J | Œ |
| 171 | AB | CTRL SHIFT K | ¡ |
| 172 | AC | CTRL SHIFT L ^ | alpha |
| 173 | AD | CTRL SHIFT M | delta |
| 174 | AE | CTRL SHIFT N | theta |
| 175 | AF | CTRL SHIFT O | lambda |
| 176 | B0 | CTRL SHIFT P | mu |
| 177 | B1 | CTRL SHIFT Q | pi |
| 178 | B2 | CTRL SHIFT R | phi |
| 179 | B3 | CTRL SHIFT S | ¡ |
| 180 | B4 | CTRL SHIFT T | ¿ |
| 181 | B5 | CTRL SHIFT U | |
| 182 | B6 | CTRL SHIFT V | § |
| 183 | B7 | CTRL SHIFT W | ¤ |
| 184 | B8 | CTRL SHIFT X | << |
| 185 | B9 | CTRL SHIFT Y | >> |
| 186 | BA | CTRL SHIFT Z | ° |
| 187 | BB | CTRL \| | ÷ |
| 188 | BC | CTRL \ | ← |
| 189 | BD | CTRL \| | → |
| 190 | BE | CTRL SHIFT 6 | ↑ |
| 191 | BF | CTRL SHIFT - | ↓ |
| 192 | C0 | Left | Ο δείκτης αριστερά ένα χαρακτήρα |
| 193 | C1 | ALT Left | Ο δείκτης στην αρχή της γραμμής |
| 194 | C2 | CTRL Left | Διαγραφή προς τα αριστερά ένα χαρακτήρα |
| 195 | C3 | CTRL ALT Left | Διαγραφή γραμμής |
| 196 | C4 | SHIFT Left | Ο δείκτης αριστερά μία λέξη |
| 197 | C5 | SHIFT ALT Left | Ρολάρισμα προς τα αριστερά |
| 198 | C6 | SHIFT CTRL Left | Διαγραφή αριστερά μία λέξη |
| 199 | C7 | SHIFT CTRL ALT Left | Ο δείκτης δεξιά ένα χαρακτήρα |
| 200 | C8 | Right | Ο δείκτης στο τέλος της γραμμής |
| 201 | C9 | ALT Right | Διαγραφή χαρακτήρα κάτω από το δείκτη |
| 202 | CA | CTRL Right | Διαγραφή μέχρι το τέλος της γραμμής |
| 203 | CB | CTRL ALT Right | Δείκτης δεξιά μία λέξη |
| 204 | CC | SHIFT Right | Ρολάρισμα στα δεξιά |
| 205 | CD | SHIFT ALT Right | Διαγραφή λέξης κάτω ή προς τα δεξιά του δείκτη |
| 206 | CE | SHIFT CTRL Right | Δείκτης προς τα πάνω |
| 207 | CF | SHIFT CTRL ALT Right | Ρολάρισμα προς τα πάνω |
| 208 | D0 | Up | Ψάξιμο προς τα πίσω |
| 209 | D1 | ALT Up | Αρχή της οθόνης |
| 210 | D2 | CTRL Up | |
| 211 | D3 | ALT CTRL UP | |
| 212 | D4 | SHIFT Up | |
| 213 | D5 | SHIFT ALT Up | |
| 214 | D6 | SHIFT CTRL Up | |
| 215 | D7 | SHIFT CTRL ALT Up | |

| Δεκαεξαδικό | Hex | Πληκτρολόγηση | Εμφάνιση/Λειτουργία |
|---|---|---|---|
| 216 | D8 | Down | Δείκτης κάτω |
| 217 | D9 | ALT Down | |
| 218 | DA | CTRL Down | Ρολάρισμα προς τα κάτω |
| 219 | DB | ALT CTRL Down | |
| 220 | DC | SHIFT Down | Ψάξιμο προς τα μπρος |
| 221 | DD | SHIFT ALT Down | |
| 222 | DE | SHIFT CTRL Down | Τέλος οθόνης |
| 223 | DF | SHIFT CTRL ALT Down | |
| 224 | E0 | CAPSLOCK | Πλήκτρο ΚΕΦΑΛΑΙΩΝ |
| 225 | E1 | ALT CAPSLOCK | |
| 226 | E2 | CTRL CAPSLOCK | |
| 227 | E3 | ALT CTRL CAPSLOCK | |
| 228 | E4 | SHIFT CAPSLOCK | |
| 229 | E5 | SHIFT ALT CAPSLOCK | |
| 230 | E6 | SHIFT CTRL CAPSLOCK | |
| 231 | E7 | SHIFT CTRL ALT CAPSLOCK | |
| 232 | E8 | F1 | |
| 233 | E9 | CTRL F1 | |
| 234 | EA | SHIFT F1 | |
| 235 | EB | CTRL SHIFT F1 | |
| 236 | EC | F2 | |
| 237 | ED | CTRL F2 | |
| 238 | EE | SHIFT F2 | |
| 239 | EF | CTRL SHIFT F2 | |
| 240 | F0 | F3 | |
| 241 | F1 | CTRL F3 | |
| 242 | F2 | SHIFT F3 | |
| 243 | F3 | CTRL SHIFT F3 | |
| 244 | F4 | F4 | |
| 245 | F5 | CTRL F4 | |
| 246 | F6 | SHIFT F4 | |
| 247 | F7 | CTRL SHIFT F4 | |
| 248 | F8 | F5 | |
| 249 | F9 | CTRL F5 | |
| 250 | FA | SHIFT F5 | |
| 251 | FB | CTRL SHIFT F5 | "Ειδικό" διάστημα |
| 252 | FC | SHIFT space | θέση TAB προς τα πίσω (ακυρώνεται το πλήκτρο CTRL) |
| 253 | FD | SHIFT TAB | |
| 254 | FE | SHIFT ENTER | "Ειδική" νέα γραμμή (ακυρώνεται το πλήκτρο CTRL) |
| 255 | FF | See below | |

Οι κωδικοί μέχρι το δεκαεξαδικό 20 είναι είτε χαρακτήρες ελέγχου ή μη εκτυπωτικοί χαρακτήρες. Η εναλλακτική πληκτρολόγηση εμφανίζεται σε παρενθέσεις μετά τη κύρια πληκτρολόγηση.

Σημειώστε ότι το CTRL-C είναι παγιδευμένο από το Qdos και δεν μπορεί να ανιχνεύσει χωρίς αλλαγές στις μεταβλητές του συστήματος.

Σημειώστε ότι οι κωδικοί C0-DF είναι διαταγές ελέγχου του δρομέα.
Το πλήκτρο ALT όταν πατηθεί με οποιαδήποτε άλλο συνδυασμό πλήκτρων εκτός από τα πλήκτρα του δρομέα ή το CAPSLOCK ενεργοποιεί τον κωδικό FF, που ακολουθείται από ένα byte που δείχνει τι θα ήταν ο κωδικός πλήκτρου αν δεν είχε πατηθεί το ALT.

Σημειώστε ακόμη ότι το CAPSLOCK και CTRL-F5 είναι παγιδευμένα από το Qdos και δεν μπορούν να ανιχνευτούν χωρίς ειδικό προγραμματισμό.

## Ρολόι (clock)

Το QL περιέχει ένα ρολόι με την πραγματική ώρα που τρέχει όταν είναι ανοικτός ο υπολογιστής.

Η μορφή που χρησιμοποιείται για την ημερομηνία και το χρόνο είναι η γνωστή ISO μορφή:

1983 JAN 01 12:09:10

Παρατήρηση: Μπορεί να ληφθεί ξεχωριστά ο χρόνος, ο μήνας, η μέρα και η ώρα μεταβιβάζοντας την τιμή τους σε μια αλφαριθμητική μεταβλητή με το όνομα DATE και τεμαχίζοντας τη μετά.

| Εντολή | Λειτουργία |
|---|---|
| SDATE | Τοποθετεί το ρολόι |
| ADATE | Διορθώνει το ρολόι |
| DATE | Δίνει την ημερομηνία σαν έναν αριθμό |
| DATE$ | Δίνει την ημερομηνία σαν μια αλφαριθμητική |
| DAY$ | Δίνει τη μέρα της εβδομάδας |

## Εξαναγκασμός (Coercion)

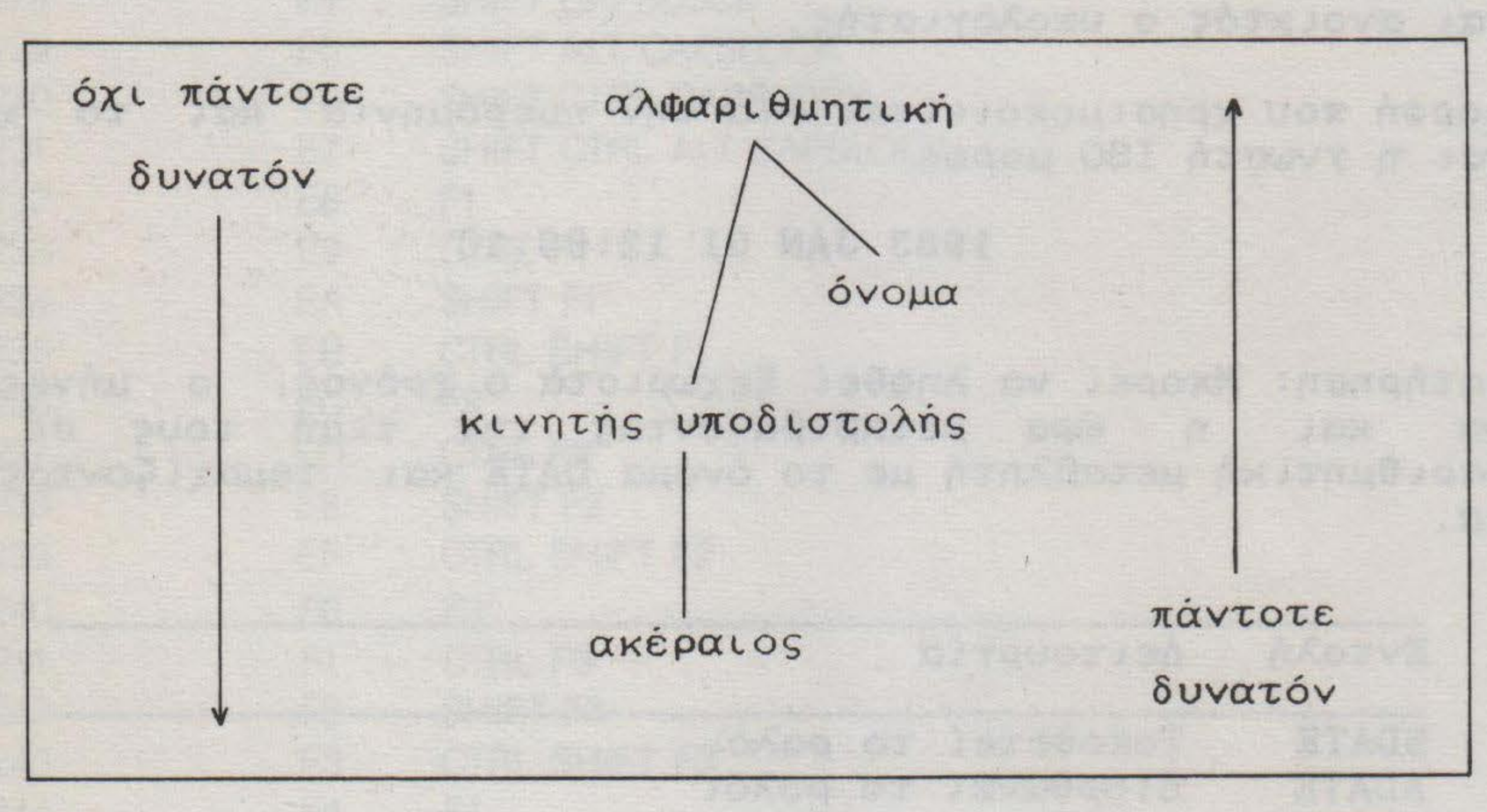
Αν είναι απαραίτητο, η SuperBASIC θα μετατρέψει τη μορφή των ακατάλληλων δεδομένων σε μια μορφή που θα επιτρέψει στη συγκεκριμένη πράξη να συνεχιστεί.

Οι συντελεστές που χρησιμοποιούνται καθορίζουν την απαιτούμενη μετατροπή. Για παράδειγμα, αν μια πράξη απαιτεί μια αλφαριθμητική παράμετρο και τροφοδοτείται μια αριθμητική παράμετρος, τότε η SuperBASIC θα μετατρέψει πρώτα την παράμετρο στην αλφαριθμητική μορφή. Δεν είναι πάντα δυνατό να μετατρέψετε τα δεδομένα στην απαιτούμενη μορφή και αν δεν μπορούν να μετατραπούν παρουσιάζεται ένα λάθος.

Η μορφή μιας παραμέτρου συνάρτησης ή διαδικασίας (procedure) μπορεί επίσης να μετατραπεί στη σωστή μορφή. Για παράδειγμα η εντολή της SuperBASIC, LOAD απαιτεί μια παράμετρο της μορφής

όνομα αλλά μπορεί να δεχτεί μια παράμετρο αλφαριθμητικής μορφής και η οποία θα μετατραπεί στη σωστή μορφή από την ίδια διαδικασία. Ο εξαναγκασμός αυτής της μορφής εξαρτάται πάντα από τον τρόπο που η συνάρτηση ή η διαδικασία εφαρμόστηκαν.

Υπάρχει μια φυσική διάταξη των μορφών δεδομένων στο QL, η διάταξη αυτή απεικονίζεται στο παρακάτω σχήμα. Η αλφαριθμητική μορφή είναι η πιο γενική μορφή μια και μπορεί να αντιπροσωπεύσει ονομασίες, πραγματικούς αριθμούς και ακέραιους. Οι πραγματικοί αριθμοί (floating point) δεν είναι τόσο γενικής μορφής όσο η αλφαριθμητική, αλλά είναι πιο γενική από τους ακέραιους μια και δεδομένα πραγματικών αριθμών μπορούν να αντιπροσωπεύουν και δεδομένα ακεραίων (σχεδόν ακριβώς). Το παρακάτω σχήμα δείχνει την κατάταξη διαγραμματικά. Τα δεδομένα μπορούν πάντα να μετατρέπονται κινώντας προς τα πάνω το διάγραμμα αλλά δεν είναι πάντα δυνατό να κινηθούν προς τα κάτω.




**Παράδειγμα**

| | |
|---|---|
| a = b + c | (δεν είναι απαραίτητη η μετατροπή πριν εκτελεστεί η πρόσθεση. Δεν είναι απαραίτητη η μετατροπή προτού προσδιοριστεί το αποτέλεσμα του α). |
| a% = b + c | (δεν είναι απαραίτητη η μετατροπή προτού εκτελεστεί η πρόσθεση, αλλά το αποτέλεσμα μετατρέπεται σε ακέραιο πριν εμφανιστεί). |
| a$ = b$ + c$ | (το b$ και το c$ μετατρέπονται σε πραγματικούς αριθμούς, αν είναι δυνατό, προτού προστεθούν μαζί. Το αποτέλεσμα μετατρέπεται στην αλφαριθμητική μορφή πριν εμφανιστεί. |
| LOAD ''mdv1_data'' | (το string "mdv1_data" μετατρέπεται στη μορφή όνομα από τη διαδιακασία φορτώματος προτού χρησιμοποιηθεί). |

Παρατήρηση: Στη SuperBASIC προτάσεις μπορούν να γραφτούν που θα δημιουργούσαν λάθη στις περισσότερες άλλες γλώσσες υπολογιστή. Γενικά, είναι πιθανό να ανακατεύετε μορφές δεδομένων με ένα πολύ εναλλακτικό τρόπο:

```
i.  PRINT "1" + 2 + "3"

ii. LET a$ = 1 + 2 + a$ + "4"
```

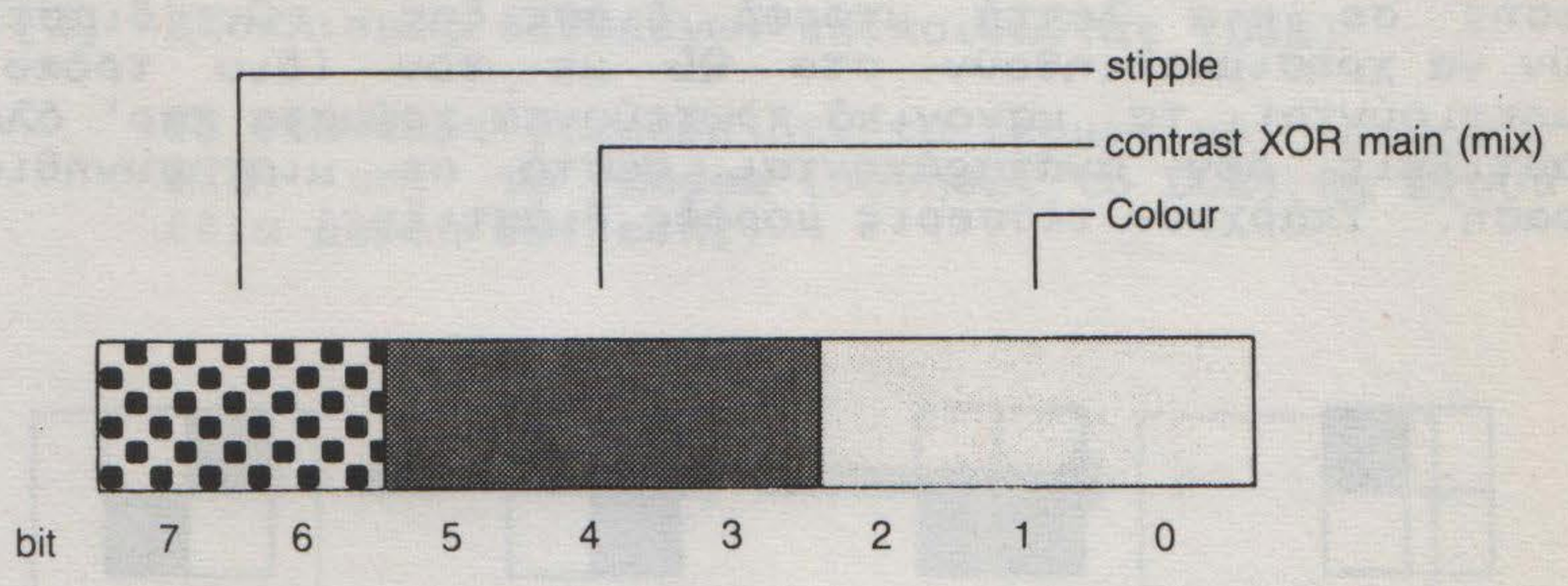
**Χρώμα (colour)**

Τα χρώματα στο QL μπορούν να είναι είτε ένα συνεχές χρώμα ή διάστικτο χρώμα, ένας συνδυασμός δύο χρωμάτων με κάποια προκαθορισμένη μορφή. Ο καθορισμός χρώματος στο QL μπορεί να έχει ως τρία διαφορετικά στοιχεία: ένα χρώμα, ένα χρώμα αντίθεσης και μια διάστικτη μορφή.

**Μονό όρισμα (single)**

```
colour:= composite_colour
```

Το μονό όρισμα προσδιορίζει τα τρία μέρη του χρώματος. Το κύριο χρώμα περιέχει τελευταία τρία bits του byte του χρώματος. Τα επόμενα τρία bits περιέχουν το αποκλειστικό ή (XOR) του κύριου χρώματος και το χρώμα της αντίθεσης του χρώματος. Τα πάνω δύο bits περιέχουν τη διάστικτη μορφή.




Με το να προσδιορίσετε μόνο τα τελευταία τρία bits (π.χ. το απαιτούμενο χρώμα) δεν θα απαιτηθεί διάστιξη και θα χρησιμοποιηθεί ένα μοναδικό χρώμα για την οθόνη.

**Διπλό Ορισμα (double)**

```
colour:= background, contrast
```

Το χρώμα είναι διάστικτο και αποτελείται από τα δύο καθορισμένα χρώματα. Υποτίθεται η default σταυρολέξου διάστιξη (διάστιξη 3).

**Τριπλό όρισμα (Triple)**

```
colour:= background, contrast, stipple
```

Τα χρώματα βάθους, αντίθεσης και της διάστιξης καθορίζονται το καθένα χωριστά.

**Χρώματα (Colours)**

Οι κώδικες για την επιλογή χρώματος εξαρτώνται από τη μορφή οθόνης που χρησιμοποιείται:

| Κωδικός | Μορφή bit | Σύνθεση | | | Χρώμα 8-χρωμη | Χρώμα 4-χρωμη |
|---|---|---|---|---|---|---|
| 0 | 0 0 0 | | | | μαύρο | μαύρο |
| 1 | 0 0 1 | | | μπλε | μπλε | μαύρο |
| 2 | 0 1 0 | | κόκκινο | | κόκκινο | κόκκινο |
| 3 | 0 1 1 | | κόκκινο + | μπλε | πορφυρό | κόκκινο |
| 4 | 1 0 0 | πράσινο | | | πράσινο | πράσινο |
| 5 | 1 0 1 | πράσινο + | | μπλε | γαλάζιο | πράσινο |
| 6 | 1 1 0 | πράσινο + | κόκκινο | | κίτρινο | άσπρο |
| 7 | 1 1 1 | πράσινο + | κόκκινο + | μπλε | άσπρο | άσπρο |

Σύνθεση χρώματος και κωδικοί.

**Διαστίξεις (stipples)**

Οι διαστίξεις αναμιγνύουν ένα χρώμα βάθους με ένα χρώμα αντίθεσης σε μια λεπτή μπορφή διάστιξης. Οι διαστίξεις μπορούν να χρησιμοποιηθούν στο QL με τον ίδιο τρόπο που χρησιμοποιούνται τα κανονικά πρωτεύοντα χρώματα παρ' όλο που οι διαστίξεις δεν αναπαράγονται σωστά σε μια συνηθισμένη τηλεόραση. Υπάρχουν τέσσερις μορφές διάστιξης:


Stipple 0


Stipple 1


Stipple 2


Stipple 3

Η διάστιξη 3 είναι η default.

**Παράδειγμα**

i. PAPER 255 : CLS

ii. PAPER 2,4 : CLS

iii. PAPER 0,2,0 : CLS

**Προειδοποίηση**

Οι διαστίξεις μπορεί να μην αναπαράγονται σωστά σε μία τηλεόραση που τροφοδοτείται διαμέσου της UHF υποδοχής.

## Επικοινωνίες (communications RS-232-C)

Το QL έχει δύο σειριακές πόρτες (που ονομάζονται SER1 και SER2) για να το συνδέουν σε συσκευή που χρησιμοποιεί σειριακή εποικοινωνία που υπακούει στο EIA στάνταρτ, στο RS-232-C ή σε ένα συμβατό στάνταρτ.

Το στάνταρτ RS-232-C είχε αρχικά σχεδιαστεί για να βοηθήσει τους υπολογιστές για να στέλνουν και να δέχονται δεδομένα διαμέσου των τηλεφονικών γραμμών χρησιμοποιώντας ένα modem. 'Ομως τώρα χρησιμοποιείται συχνά για να συνδέει τους υπολογιστές απευθείας τον ένα με τον άλλο και σε πολλά μέρη με περιφερειακές συσκευές, π.χ. εκτυπωτές, μονάδες σχεδίασης κ.τ.λ.

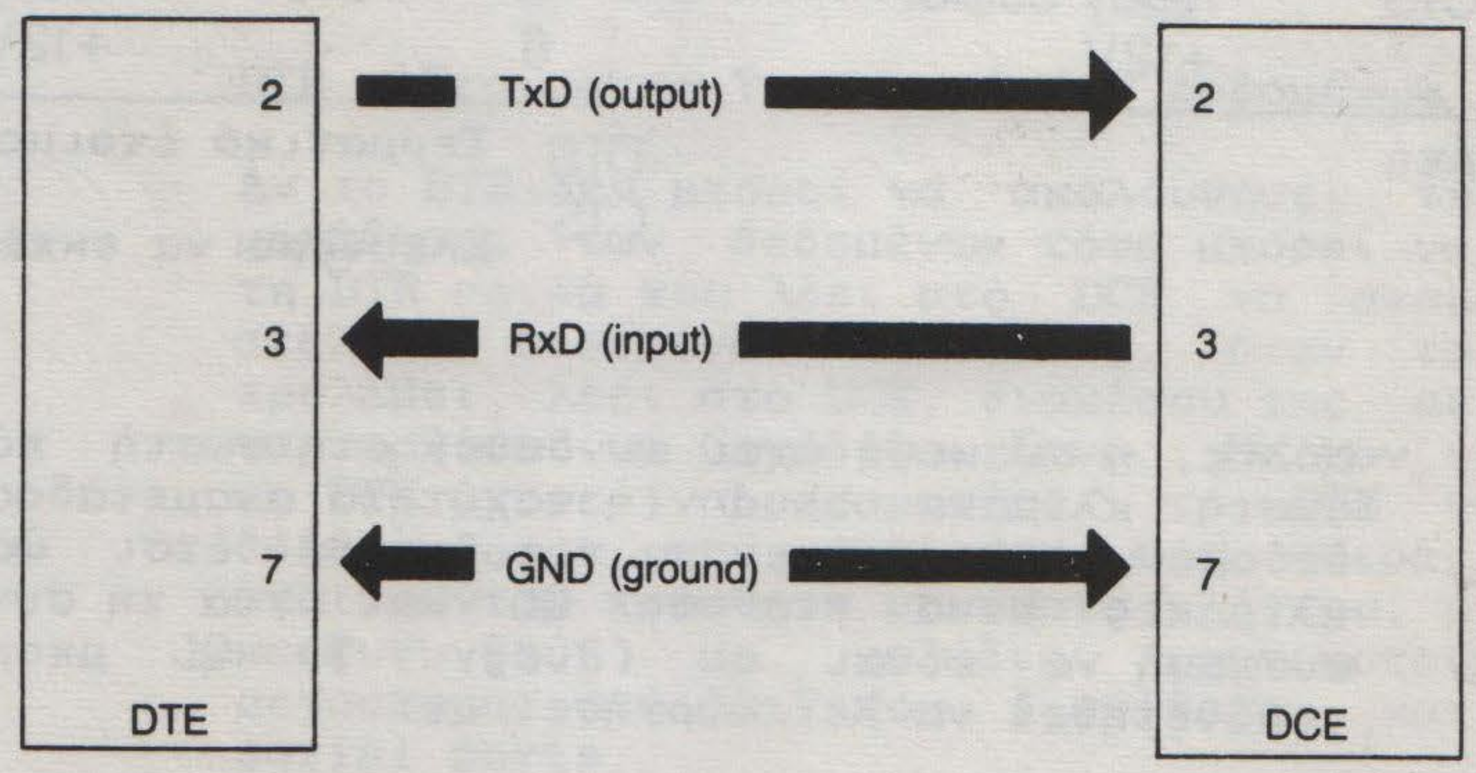
Καθώς το στάνταρτ RS-232-C μπορεί να δηλωθεί με διαφορετικές μορφές σε διαφορετικά κομμάτια εξοπλισμού μπορεί να είναι μια πολύ δύσκολη εργασία, ακόμα και για έναν ειδικό, για να συνδέσετε μαζί, για μια πρώτη φορά, δύο κομμάτια υποτιθέμενου στάνταρ RS-232-C εξοπλισμού. Αυτό το μέρος θα προσπαθήσει να καλύψει τα περισσότερα από τα βασικά προβλήματα που θα συναντήσετε.

Το RS-232-C αναφέρεται σε δύο μορφές εξοπλισμού:

[1] Εξοπλισμός δεδομένων τερματικού (DTE)

[2] Εξοπλισμός δεδομένων επικοινωνίας (DCE)

Το στάνταρτ θεώρησε ότι το τερματικό (συνήθως το DTE) και το modem (συνήθως το DCE) θα έχουν την ίδια μορφή σύνδεσης.




Το παραπάνω διάγραμμα απεικονίζει πως το DTE μεταδίδει δεδομένα στο pin 2 ενώ το DCE πρέπει να δεχθεί στο δικό του pin 2 (το οποίο ακόμα ονομάζεται μετάδοση δεδομένων!). Παρόμοια, το DTE δέχεται δεδομένα στο pin 3, ενώ το DTE πρέπει να μεταδίδει δεδομένα στο δικό του pin 3 (αυτό

ονομάζεται ακόμα μετάδοση δεδομένων). Ενώ είναι πολύ μπερδεμένο από μόνο του, μπορεί να δημιουργήσει ακόμα μεγαλύτερα προβλήματα όταν υπάρχει συμφωνία στο ότι μια συγκεκριμένη συσκευή θα πρέπει να καθοριστεί σαν DCE ή DTE.

Δυστυχώς, μερικοί άνθρωποι αποφασίζουν ότι οι υπολογιστές θα πρέπει να είναι DCE συσκευές ενώ άλλοι καθορίζουν αντίστοιχους υπολογιστές σαν DTE συσκευές. Είναι φανερό ότι αυτό οδηγεί σε δυσκολίες στον καθορισμό των σειριακών πορτών (serial ports) σε κάθε μέρος εξοπλισμού.

Το SER1 στο QL καθορίζεται σαν DCE ενώ το SER2 καθορίζεται σαν DTE. θα πρέπει να είναι δυνατό να συνδέσετε τουλάχιστον μια από τις σειριακές πόρτες σε μια δοσμένη συσκευή, απλά, χρησιμοποιώντας οποιαδήποτε πόρτα έχει το καλώδιο με το σωστό τρόπο. Το pin-out για τις σειριακές πόρτες δίνεται παρακάτω. Το καλώδιο για να συνδέσετε το QL με μια στάνταρ 25-way σύνδεση τύπου "D" μπορεί να βρεθεί.

| SER1 | | | SER2 | | |
|---|---|---|---|---|---|
| pin | name | function | pin | name | function |
| 1 | GND | signal ground | 1 | GND | signal ground |
| 2 | TxD | input | 2 | TxD | output |
| 3 | RxD | output | 3 | RxD | input |
| 4 | DTR | ready input | 4 | DTR | ready output |
| 5 | CTS | ready output | 5 | CTS | ready input |
| 6 | - | +12V | 6 | - | +12V |

TxD Εκπομπή
RxD Λήψη

DTR Τερματικό έτοιμο
CTS Ελεύθερο να εκπέμψει

Μόλις, η συσκευή έχει συνδεθεί στη σωστή πόρτα η σωστή κλίμακα baud (η ταχύτητα αναμετάδοσης των δεδομένων) θα πρέπει να τοποθετηθεί έτσι ώστε οι κλίμακες baud για το QL και για τη συνδεμένη συσκευή να είναι οι ίδιες. Το QL μπορεί να τοποθετηθεί να λειτουργήσει σε

75
300
600
1200
2400
4800
9600
19200 (transmit only) baud.

Η κλίμακα baud του QL καθορίζεται από την εντολή BAUD και είναι και για τα δύο κανάλια. Οι κλίμακες baud δεν μπορούν να καθοριστούν ανεξάρτητα.

Η ισοτιμία (parity) που πρόκειται να χρησιμοποιηθεί από το QL θα πρέπει επίσης να τοποθετηθεί για να ταιριάζει με την αναμενόμενη από τον περιφερειακό εξοπλισμό. Η ισοτιμία συνήθως χρησιμοποιείται για να παρατηρήσει απλά λάθη μετάδοσης και μπορεί να τοποθετηθεί να είναι περιτή (even), αρτία (odd), σημείωση (mark), διάστημα (space) ή καθόλου (no parity) π.χ. και τα 8 bits του byte να χρησιμοποιούνται για δεδομένα.

Το Stop bits σημειώνουν το τέλος της μετάδοσης ενός byte ή ενός χαρακτήρα. Το QL θα δεχθεί δεδομένα με ένα, ένα και μισό, ή δύο stop bits, και θα μεταδίδει πάντα δεδομένα τουλάχιστον με δύο stop bits. Σημειώστε ότι αν το QL είναι τοποθετημένο σε 9600 baud δε θα δεχθεί με ένα μόνο stop bit.

Μπορεί να είναι απαραίτητο να συνδέσετε τις (handshake) σειρές χειραψίας ανάμεσα στο QL και ένα συνδεμένο μέρος εξοπλισμού. Αυτό επιτρέπει στο QL και στο περιφερειακό του να επιμελούνται και να ελέγχουν την κλίμακα της επικοινωνίας τους. Μπορεί να χρειαστεί να γίνει αυτό αν ένα από τα δύο δεν μπορεί να ανταπεξέλθει στην ταχύτητα με την οποία τα δεδομένα μεταδίδονται. Το QL χρησιμοποεί δύο σειρές χειραψίας:

CTS Καθαρό για αποστολή

DTR 'Ετοιμο το Τερματικό των Δεδομένων

Αν το DTE δεν μπορεί να ακολουθήσει την κλίμακα μετάδοσης των δεδομένων τότε μπορεί να αναιρέσει τη DTR σειρά που λέει στο DCE να σταματήσει να στέλνει δεδομένα. Προφανώς, όταν το DTE έχει προλάβει, λέει στο DCE, διαμέσου της σειράς DTR, να αρχίσει να μεταδίδει ξανά. Με τον ίδιο τρόπο, το DCE μπορεί να σταματήσει το DTE να στέλνει δεδομένα με την αναίρεση της σειράς CTS. Αν απαιτούνται πρόσθετα μηνύματα ελέγχου, μπορούν να κατευθυνθούν με καλώδιο χρησιμοποιώντας το μετασχηματιστή 12 V που διατίθεται και στα δύο serial ports.

Παρ' όλο που η μετάδοση είναι πιθανή διαμέσου του QL χωρίς καθόλου handshaking, το QL δε θα δεχθεί σωστά κάτω από οποιεσδήποτε συνθήκες χωρίς τη χρήση του CTS στο SER1 και του DTR στο SER2.

Οι επικοινωνίες στο QL είναι πλήρως διπλές (full duplex), δηλαδή η μετάδοση και η λήψη γίνονται ταυτόχρονα.

Η ισοτιμία και η χειραψία επιλέγονται όταν το σειριακό κανάλι είναι ανοικτό.

| Εντολή | Λειτουργία |
|---|---|
| BAUD | ορίζει την ταχύτητα μετάδοσης |
| OPEN | ανοίγει τα σειριακά κανάλια* |
| CLOSE | κλείνει τα σειριακά κανάλια |

* δείτε στην έννοια συσκευή για έναν πλήρη ορισμό.

## Μορφές μεταβλητών δεδομένων

### Ακέραιοι (Integers)

Οι ακέραιοι είναι ολόκληροι αριθμοί στην κλίμακα από το -32768 ως το +32767. Οι μεταβλητές θεωρούνται ότι είναι ακέραιες αν ο αναγνωριστής μεταβλητής ακολουθείται από το σύμβολο %. Δεν υπάρχουν ακέραιες σταθερές στη SuperBASIC, έτσι όλες οι σταθερές αποθηκεύονται σαν αριθμοί κινητής υποδιαστολής.

syntax: *identifier*%

example:
i. counter%
ii. size_limit%
iii. this_is_an_integer_variable%

### Αριθμοί κινητής υποδιαστολής (Floating point)

Οι αριθμοί κινητής υποδιαστολής βρίσκονται στην κλίμακα +-(10 εις την 615 έως 10 εις την -615) με 8 σημαντικά ψηφία. Οι αριθμοί κινητής υποδιαστολής είναι η default μορφή δεδομένων στη SuperBASIC. 'Ολες οι σταθερές κρατιούνται σε μορφή κινητής υποδιαστολής και μπορούν να εισαχθούν χρησιμοποιώντας εκθετική μορφή.

syntax: *identifier* | *constant*

example:
i. current_accumulation
ii. 76.2356
iii. 354E25

### $tring

Μία αλφαριθμητική είναι μια ακολουθία από χαρακτήρες με μήκος ως 32766 χαρακτήρες. Οι μεταβλητές γίνονται δεκτές ότι έχουν μορφή αλφαριθμητικής αν το όνομα της μεταβλητής ακολουθείται από ένα $. Τα αλφαριθμητικά δεδομένα αντιπροσωπεύονται με το κλείσιμο των απαιτούμενων χαρακτήρων σε μονά ή διπλά εισαγωγικά.

syntax: *identifier*$ | "*text*"

example:
i. `string_variables$`
ii. `"this is string data"`
iii. `"this is another string"`

**'Ονομα (name)**

Η μορφή όνομα έχει την ίδια μορφή όπως ένα κανονικό SuperBASIC αναγνωριστή και χρησιμοποιείται από το σύστημα για να ονομάσει αρχεία κ.τ.λ.

syntax: *identifier*

example:
i. `mdv1_data_file`
ii. `ser1e`

**Συσκευές**

Μια συσκευή είναι ένα κομμάτι εξοπλισμού στο QL στο οποίο τα δεδομένα μπορούν να εισαχθούν (input) ή να ειξαχθούν (output).

Μια και το σύστημα δεν κάνει υποθέσεις για την συσκευή I/O, η οποία θα χρησιμοποιηθεί, η I/O συσκευή μπορεί εύκολα να αλλαχθεί και τα δεδομένα να εκτραπούν ανάμεσα στις συσκευές. Για παράδειγμα, ένα πρόγραμμα μπορεί να πρέπει να εξάγει προς έναν εκτυπωτή σε κάποιο σημείο στη διάρκεια του τρεξίματος. Αν δε διατίθεται ο εκτυπωτής τότε το εξαγόμενο μπορεί να εκτραπεί σε ένα αρχείο του Microdrive και να φυλαχθεί. Το αρχείο μπορεί τότε να εκτυπωθεί σε μια αργότερη ημερομηνία. Το I/O στο QL μπορεί να παρθεί σαν να γράφεται και να διαβάζεται από ένα λογικό αρχείο που είναι μια στανταρτ μορφή ανεξάρτητα από τη συσκευή.

'Ολες οι συγκεκριμένες λειτουργίες συσκευών εκτελούνται από ανεξάρτητες drivers συσκευής ειδικά γραμμένες για κάθε συσκευή στο QL. Το σύστημα αυτόματα θα βρει και θα περιλάβει drivers που είναι κατάλληλοι. Αυτοί θα πρέπει να είναι γραμμένοι στην ανάλογη φόρμα του driver της συσκευής του QL, δες την έννοια περιφερειακή επέκταση.

'Οταν ενεργοποιηθεί μια συσκευή, ανοίγεται ένα κανάλι και συνδέεται, με τη συσκευή. Για να ανοίξετε σωστά ένα κανάλι θα πρέπει να δοθούν βασικές πληροφορίες. Αυτές οι επιπλέον πληροφορίες μπορούν να προσαρτηθούν στο όνομα της συσκευής (π.χ. με την προσάρτηση του ονόματος ενός αρχείου δίσκου στο όνομα της μονάδας δίσκου).

Το όνομα του αρχείου θα πρέπει να συμβιβάζεται στους κανόνες για ένα τύπο όνομα SuperBASIC παρ' όλο που είναι επίσης πιθανό να δημιουργήσετε το όνομα αρχείου σαν μια SuperBASIC αλφαριθμητική έκφραση.

Συνολικά η γενική μορφή ενός ονόματος αρχείου είναι:

identifier [information]

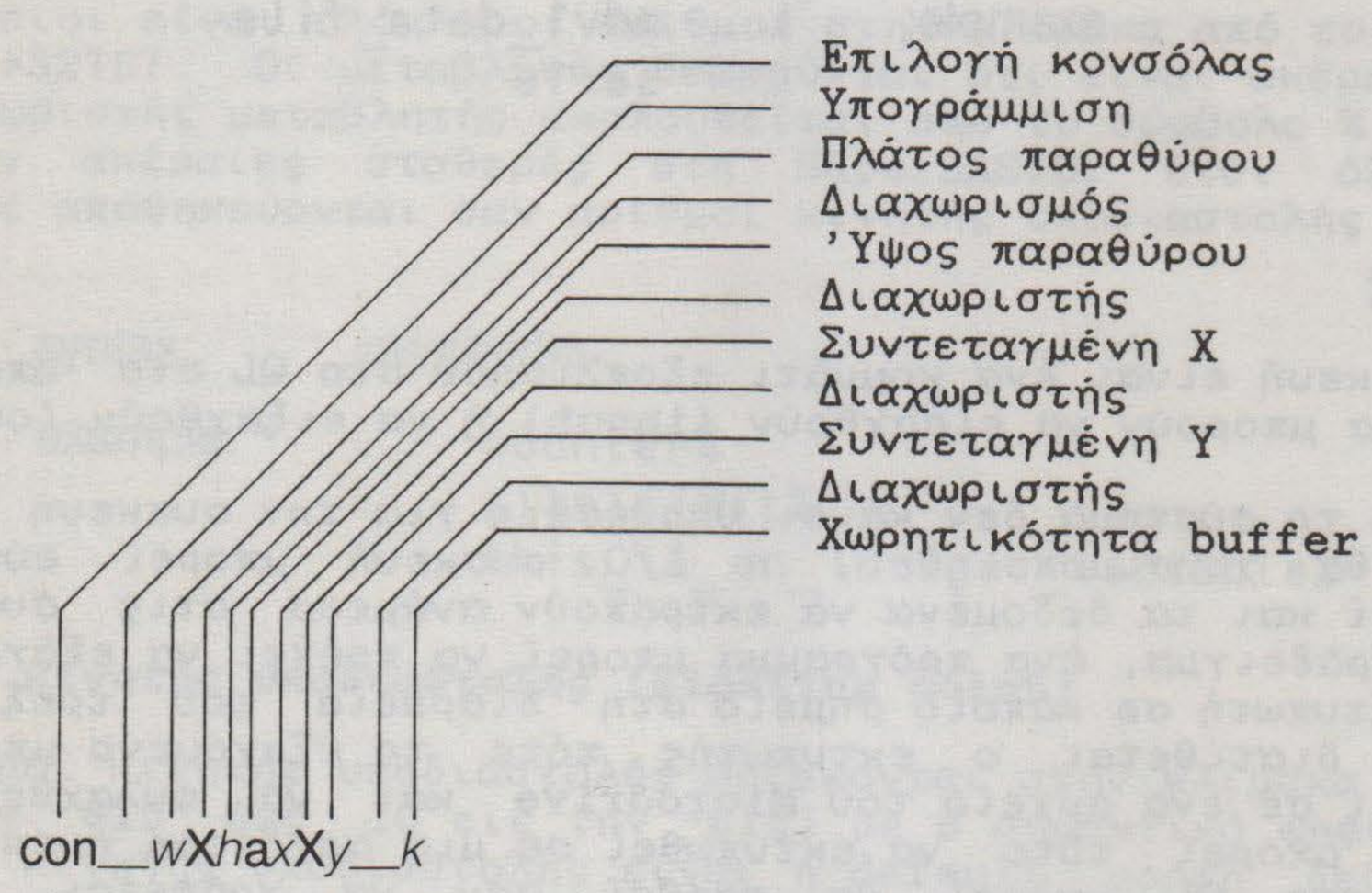
όπου ολόκληρο το όνομα του αρχείο (συμπεριλαμβανομένης της επιπλέον πληροφορίας) συμφωνεί με τους κανόνες του SuperBASIC αναγνωριστή.

Κάθε λογική συσκευή στο σύστημα απαιτεί τη δική της συγκεκριμένη "επιπλέον πληροφορία" παρ' όλο που οι default παράμετροι θα υποτεθούν στην κάθε περίπτωση αν είναι δυνατό:

ΟΡΙΣΜΟΣ device: = name

όπου η μορφή του ονόματος της συσκευής περιγράφεται παρακάτω.

Παράδειγμα:




**CON_*w*X*h*a*x*X*y*_*k*** Console I/O

| | |
|---|---|
| [*w*X*h*] | - window *width, height* |
| [A*x*X*y*] | - window *X,Y* coordinate of upper left-hand corner |
| [*k*] | - keyboard type ahead buffer length (bytes) |

default: con_448x180a32x16_128

παράδειγμα
```
OPEN #4,con_20x50a0x0_32
OPEN #8,con_20x50
OPEN #7,con_20x50a10x10
```

'Εξοδος σε οθόνη

**SCR_*w*X*h*a*x*X*y*** [*w*X*h*] - window *width, height*
[A*x*X*y*] - window *X,Y* coordinate

default: scr__448x180a32x16

παράδειγμα
```
OPEN #4, scr_10x10a20x50
OPEN #5, scr_10x10
```

SER*nphz*

Σειριακή πόρτα (RS 232 - C)

n = 1 ή 2

| ισοτιμία | χειραψία | πρωτόκολο |
|---|---|---|
| e – even | i – ignore | r – raw data no EOF |
| o – odd | h – handshake | z – control Z is EOF |
| m – mark | | c – as z but converts ASCII 10 (Qdos newline character) to ASCII 13 <CR>) |
| s – space | | |

default: **ser1rh** (8 bit no parity with handshake)

παράδειγμα
```
OPEN #3, ser1e
OPEN #4, serc
COPY mdv1_test_file TO ser1c
```

NET*d__s*

Σειριακό δίκτυο I/O

d δεικνύει κατεύθυνση

i - input
o - output

s = αριθμός σταθμού

0 - for broadcast
**own station** - for general listen (input only)

default: no default

example:
```
OPEN #7, neti_32
OPEN #4, neto_0
COPY ser1 TO neto_21
```

MDVn__*name*

Προσπέλαση αρχείου σε microdrive
n = αριθμός microdrive
name = όνομα αρχείου

default: no default

παράδειγμα
```
OPEN #9, mdv1_data_file
OPEN #9, mdv1_test_program
COPY mdv1_test_file TO scr_
```

| Εντολή | Λειτουργία |
|---|---|
| OPEN | αρχίζει μια συσκευή και την ενεργοποιεί για χρήση |
| CLOSE | αποενεργοποιεί μια συσκευή |
| COPY | αντιγράφει τα δεδομένα ανάμεσα στις συσκευές |
| COPY_N | ίδιο με COPY εκτός επικεφαλίδας αρχείου |
| EOF | εξετάζει για το τέλος του αρχείου |
| WIDTH | ορίζει το πλάτος εκτύπωσης |

## Άμεση διαταγή

Η SuperBASIC δημιουργεί διαφορά ανάμεσα σε μια εντολή που πληκτρολογείται μετά από έναν αριθμό σειράς και μια εντολή που πληκτρολογείται χωρίς έναν αριθμό σειράς. Χωρίς αριθμό σειράς η εντολή είναι μία άμεση διαταγή και εκτελείται αμέσως από τη SuperBASIC. Για παράδειγμα, η RUN πληκτρολογείται σαν άμεση διαταγή και εκτελείται, το αποτέλεσμα είναι ότι το πρόγραμμα

αρχίζει να τρέχει. Αν μία εντολή πληκτρολογείται με έναν αριθμό σειράς τότε η σύνταξη της σειράς εξετάζεται και αναφέρονται τα τυχόν παρατηρημένα λάθη σύνταξης. Μια σωστή σειρά εισάγεται στο πρόγραμμα SuperBASIC και αποθηκεύεται. Αυτές οι εντολές αποτελούν ένα SuperBASIC πρόγραμμα και θα εκτελεστούν μόνο όταν αρχίσει το πρόγραμμα με τις εντολές RUN ή GOTO.

Δεν έχουν όλες οι εντολές της SuperBASIC νόημα όταν εισάγονται σαν μια άμεση διαταγή. Για παράδειγμα, η END FOR, η END DEFine, κ.τ.λ.

| Διαταγή | Λειτουργία |
|---|---|
| RUN | τρέχει ένα πρόγραμμα |
| NEW | καθαρίζει ένα πρόγραμμα |
| CLEAR | καθαρίζει τις μεταβλητές |
| FORMAT | φορμάρει μία μικροκασέτα |
| SAVE | φυλάσσει ένα πρόγραμμα |
| LOAD | φορτώνει ένα πρόγραμμα |
| MERGE | συγχωνεύει ένα πρόγραμμα |
| LRUN | φορτώνέι και τρέχει ένα πρόγραμμα |
| LIST | καταγράφει ένα πρόγραμμα. |
| RENUM | επαναριθμεί ένα πρόγραμμα |
| DLINE | διαγράφει σειρές του προγράμματος |
| PRINT | εκτυπώνει δεδομένα |
| EXEC | εκτελεί ένα έργο |
| EDIT | διορθώνει ένα πρόγραμμα |
| AUTO | αριθμεί ένα πρόγραμμα |
| COPY | αντιγράφει τα δεδομένα ανάμεσα στις συσκευές |

## Χειρισμός των λαθών

Τα λάθη αναφέρονται από τη SuperBASIC σε μια συγκεκριμένη μορφή.

```
AT line line_number error_text
```

'Οπου ο αριθμός σειράς (line-number) είναι ο αριθμός της σειράς όπου παρατηρήθηκε το λάθος και το κείμενο λάθους καταγράφεται παρακάτω.

[1] Not complete

Μια προσπάθεια έχει τερματιστεί πρόωρα ή έχει πατηθεί το break.

[2] Invalid job

Μια επιστροφή λάθους από τα Qdos σχετικά με το σύστημα καλεί τον έλεγχο των πολλαπλών εργασιών ή I/O.

[3] Out of memory

Τα Qdos και/ή η SuperBASIC έχει ανεπαρκή ελεύθερη μνήμη.

[4] Out of range

Συνήθως προέρχεται από μια προσπάθεια για να γράψετε έξω από ένα παράθυρο ή μια μεταβλητή με δείκτη είναι λανθασμένη.

[5] Buffer full

Μία I/O εργασία για να φέρετε από ένα buffer γεμάτο με χαρακτήρες γέμισε το buffer προτού να βρεθεί το τέλος μιας εγγραφής.

[6] Channel not open

Προσπάθεια για να διαβάσετε, να γράψετε ή να κλείσετε ένα κανάλι που δεν έχει ανοιχτεί. Μπορεί επίσης να συμβεί αν μια προσπάθεια για να ανοίξετε το κανάλι αποτύχει.

[7] Not found

Δεν μπορούν να βρεθούν το σύστημα αρχείου η συσκευή, το μέσο ή το αρχείο.

Η SuperBASIC δεν μπορεί να βρει έναν αναγνωριστή. Αυτό μπορεί να είναι σαν αποτέλεσμα από λανθασμένες φωλιασμένες δομές.

[8] Already exists

Το σύστημα αρχείου έχει βρει ένα ήδη υπαρκτό αρχείο με το ίδιο όνομα σαν αυτό που πρόκειται να ανοιχθεί για να γράψουμε.

[9] In use

Το σύστημα αρχείου έχει βρει ότι ένα αρχείο ή συσκευή χρησιμοποιείται ήδη αποκλειστικά.

[10] End of file

Παρατηρήθηκε το τέλος του αρχείου κατά τη διάρκεια της εισαγωγής.

[11] Drive full

Μια συσκευή συνήθως Microdrive έχει γεμίσει.

[12] Bad name

Το σύστημα αρχείου έχει αναγνωρίσει το όνομα αλλά υπάρχει ένα λάθος στη σύνταξη ή στην τιμή της παραμέτρου.

Στη SuperBASIC σημαίνει ότι ένα όνομα έχει χρησιμοποιηθεί άσχετα από τον προορισμό του. Για παράδειγμα, μια μεταβλητή έχει χρησιμοποιηθεί σαν μία

διαδικασία.

[13] Xmit error

Λάθος ισοτιμίας RS-233-C.

[14] Format failed

'Εχει αποτύχει μια προσπάθεια φορμαρίσματος, το μέσο είναι πιθανά ελαττωματικό.

[15] Bad parameter

Υπάρχει ένα λάθος στον κατάλογο των παραμέτρων ενός συστήματος ή μιας διαδικασίας της SuperBASIC ή κλήσης μιας συνάρτησης.

'Εγινε μια προσπάθεια να διαβαστούν δεδομένα από μια συσκευή μόνο γραψίματος.

[16] Bad or changed medium

Το μέσο είναι πιθανά ελαττωματικό.

[17] Error in expression

Παρατηρήθηκε ένα λάθος καθώς υπολογιζόταν μια παράσταση.

[18] Overflow

Αριθμητική υπερχείλιση, διαίρεση με το μηδέν, τετραγωνική ρίζα ενός αρνητικού αριθμού κ.τ.λ.

[19] Not Implemented

[20] Read only

'Εγινε μια προσπάθεια να γραφτούν δεδομένα σε ένα συγχρόνως χρησιμοποιούμενο αρχείο.

[21] Bad line

'Εχει συμβεί ένα λάθος στη σύνταξη της SuperBASIC.

[22] PROC/FN cleared

Αυτό είναι ένα μήνυμα για πληροφόρηση μόνο και όχι αναφορά λάθους π.χ. ότι το πρόγραμμα σταμάτησε.

**Ανάκληση λάθους**

Αφού παρουσιάστηκε ένα λάθος, το πρόγραμμα μπορεί να ξαναρχίσει στην επόμενη εντολή, πληκτρολογώντας:

CONTINUE

Αν η κατάσταση του λάθους μπορεί να διορθωθεί, χωρίς να αλλάξει το πρόγραμμα, αυτό μπορεί να ξαναρχίσει στην εντολή που δημιούργησε το λάθος. Πληκτρολογήστε:

RETRY

## Εκφράσεις

Οι εκφράσεις στη SuperBASIC μπορούν να είναι string, αριθμητικές, λογικές ή ένα μίγμα οι ακατάλληλες μορφές δεδομένων μετατρέποντας αυτόματα σε μια κατάλληλη μορφή από το σύστημα όποτε αυτό είναι πιθανό.

**ΟΡΙΣΜΟΣ**

*monop*:= | +
| –
|NOT

***expression***:= | [*monop*] *expression operator expression*
| (*expression*)
| *atom*

*atom*:= | *variable*
| *constant*
| *function* [ ( *expression* *[ , *expression*] *) ]
| *array__element*

*variable*:= | *identifier*
| *identifier* **%**
| *identifier* **$**

*function*:= | *identifier*
| *identifier* **%**
| *identifier* **$**

*constant*:= | *digit* * [*digit*] *
| *[*digit*] * . *[*digit*] *
| *[*digit*] * [.] *[*digit*] * **E** *[*digit*] *

Η τελική τιμή που επιστρέφεται από τον υπολογισμό της έκφρασης μπορεί να είναι ένας ακέραιος που δίνει μία integer_expression, μία αλφαριθμητική που δίνει μια string_expression ή ένας πραγματικός αριθμός (κινητής υποδιαστολής) που δίνει μια floating_expression. Συχνά οι εκφράσεις κινητής υποδιαστολής και ακέραιες είναι ισοδύναμες και χρησιμοποιείται η έννοια numeric_expression.

Οι λογικοί τελεστές μπορούν να περιληφθούν μέσα σε μια έκφραση. Αν η συγκεκριμένη λειτουργία είναι αληθής τότε το ένα επιστρέφεται σαν η τιμή της λειτουργίας. Αν η λειτουργία είναι ψευδής τότε επιστρέφεται ένα μηδέν. Παρ' όλο που οι

λογικοί τελεστές μπορούν να χρησιμοποιηθούν σε οποιαδήποτε έκφραση, συνήθως όμως χρησιμοποιούνται σαν μέρος μιας IF εντολής.

i. test_data + 23.3 + 5
ii. "abcdefghijklmnopqrstuvwxyz"(2 TO 4)
iii. 32.1 * (colour=1)
iv. count = -limit

## Τύποι αρχείων

### Αρχεία

'Ολα τα I/O στο QL είναι προς ή από ένα λογικό αρχείο. Υπάρχουν πολλοί τύποι αρχείων.

| | |
|---|---|
| data | Είναι SuperBASIC προγράμματα, ή αρχεία κειμένου. Δημιουργούνται με τη χρήση του PRINT, SAVE, μπορούν να προσπελαθούν χρησιμοποιώντας το INPUT, INKEY$, LOAD κ.τ.λ. |
| exec | 'Ενα εκτελέσιμο παροδικό πρόγραμμα. Φυλάσσεται με τη χρήση SEXEC, φορτώνεται χρησιμοποιώντας το EXEC, το EXEC_W κ.τ.λ. |
| code | Ακατέργαστα δεδομένα μνήμης, εικόνες οθόνης κ.τ.λ. Φυλάσσονται χρησιμοποιώντας το SBYTES, φορτώνονται χρησιμοποιώντας το LBYTES. |

## Συναρτήσεις και διαδικασίες

Οι SuperBASIC συναρτήσεις και διαδικασίες ορίζονται με τις εντολές DEFine FuNction και DEFine PROCedure. Μια συνάρτηση ενεργοποιείται (ή καλείται) με την πληκτρολόγηση του ονόματός της στο κατάλληλο σημείο στη SuperBASIC έκφραση. Η συνάρτηση θα πρέπει να περιλαμβάνεται σε μια έκφραση γιατί επιστρέφει μια τιμή που πρέπει να χρησιμοποιηθεί. Μία διαδικασία μπαίνει σε λειτουργία (ή καλείται) με την πληκτρολόγηση του ονόματος του σαν το πρώτο μέρος σε μία εντολή της SuperBASIC.

Μπορούν να περάσουν δεδομένα μέσα σε μια συνάρτηση ή διαδικασία με την προσκόληση ενός καταλόγου πραγματικών παραμέτρων μετά τη συνάρτηση ή το όνομα της διαδικασίας. Ο κατάλογος συγκρίνεται με ένα παρόμοιο κατάλογο που είναι προσαρτημένος μετά το όνομα της συνάρτησης ή του διαβήματος όταν αυτό καθορίστηκε. Αυτός ο δεύτερος κατάλογος ονομάζεται τυπικές παράμετροι της συνάρτησης ή της διαδικασίας. Οι τυπικές παράμετροι θα πρέπει να είναι μεταβλητές της SuperBASIC. Οι πραγματικές παράμετροι θα πρέπει να είναι μια μεταβλητή με δείκτες ένα κομμάτι μεταβλητής με δείκτη ή μια SuperBASIC έκφραση της οποίας η απλούστερη μορφή είναι μια απλή μεταβλητή ή σταθερή.

Μια και οι πραγματικές παράμετροι είναι πραγματικές εκφράσεις θα πρέπει να έχουν μια πραγματική μορφή συνδεδεμένη μ' αυτές.

Οι τυπικές παράμετροι απλά χρησιμοποιούνται για να σημειώσουν πως πρέπει να επεξεργάζονται οι τωρινές παράμετροι και γι' αυτό δε συνδέεται κάποια μορφή με αυτές. Τα στοιχεία σε κάθε κατάλογο παραμέτρων ταιριάζονται κατά σειρά όταν η συνάρτηση ή η διαδικασία καλείται και οι τυπικές παράμετροι γίνονται ισοδύναμες με τις πραγματικές παραμέτρους. Υπάρχουν τρεις ξεχωριστοί τρόποι για τη χρήση παραμέτρων.

Αν η πραγματική παράμετρος είναι μια απλή μεταβλητή, και αν εκχωρούνται δεδομένα στην τυπική παράμετρο στη συνάρτηση ή στη διαδικασία, τότε τα δεδομένα εκχωρούνται επίσης στην αντίστοιχη πραγματική παράμετρο.

Αν η πραγματική παράμετρος είναι μια έκφραση τότε η εκχώρηση δεδομένων στην αντίστοιχη τυπική παράμετρο δε θα έχει κανένα αποτέλεσμα έξω από τη διαδικασία. Σημειώστε ότι μια μεταβλητή μπορεί να γίνει μία έκφραση εγκλείοντας την σε αγκύλες.

Αν η πραγματική παράμετρος είναι μία μεταβλητή αλλά δεν έχει οριστεί προηγουμένως τότε η εκχώρηση των δεδομένων στην αντίστοιχη τυπική παράμετρο θα τοποθετήσει τη συγκεκριμένη μεταβλητή σαν την πραγματική παράμετρο.

Οι μεταβλητές μπορούν να οριστούν να είναι τοπικές σε μια συνάρτηση ή διαδικασία με την πρόταση LOCAL. Οι τοπικές μεταβλητές δεν έχουν κανένα αποτέλεσμα σε παρόμοια ονομασμένες μεταβλητές έξω από τη συνάρτηση ή τη διαδικασία στα οποία είναι ορισμένες και έτσι επιτρέπουν μεγαλύτερη ελευθερία στην επιλογή λογικών ονομάτων μεταβλητών χωρίς να διακινδυνεύετε να καταστρέψετε εξωτερικές μεταβλητές. Οι τοπικές μεταβλητές διατίθενται σε οποιαδήποτε εσωτερική συνάρτηση ή διαδικασία εκτός αν επανακαθορίζονται να είναι τοπικές.

Οι συναρτήσεις και οι διαδικασίες στη SuperBASIC μπορούν να χρησιμοποιηθούν εναλλάξ. Αυτό σημαίνει ότι μία συνάρτηση ή μία διαδικασία μπορεί να καλέσει τον εαυτό της είτε απευθείας είτε έμμεσα.

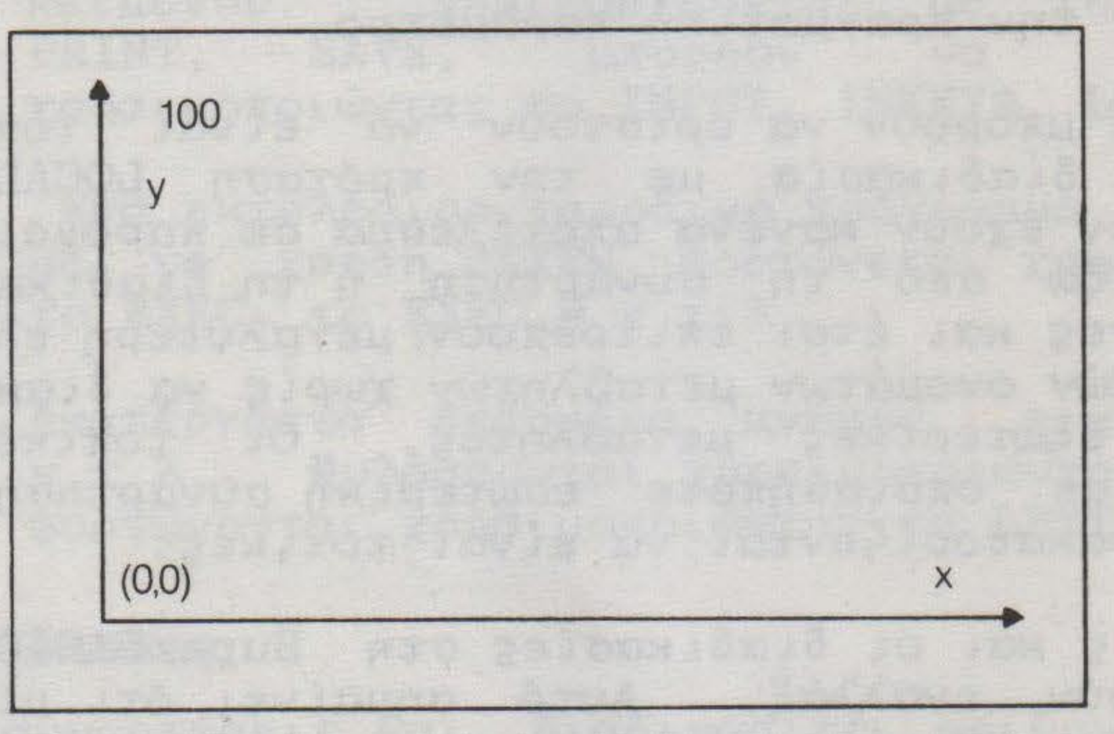
| Διαταγή | Λειτουργία |
|---|---|
| DEFine FuNction | ορίζει μία συνάρτηση |
| DEFine PROCedure | ορίζει μία διαδικασία |
| RETurn | αφήνει μια συνάρτηση ή μία διαδικασία (επιστρέφει δεδομένα από μία συνάρτηση) |
| LOCal | ορίζει τα τοπικά δεδομένα σε μια συνάρτηση ή διαδικασία |

## Γραφικά

Είναι σπουδαίο να συνειδητοποιήσετε ότι η οθόνη του QL έχει όχι τετραγωνικά pixels και ότι με την αλλαγή της μορφής θα αλλάξει και το σχήμα των pixels. 'Ετσι αν οι γραφικές διαδικασίες ήταν απλά βασισμένες στα pixels θα σχεδίαζαν διαφορετικά σχήματα στις δύο μορφές. Για παράδειγμα, στη μία μορφή θα είχαμε έναν κύκλο ενώ το ίδιο σχήμα σε μία άλλη μορφή θα ήταν μια έλλειψη.

Οι διαδικασίες των γραφικών βεβαιώνουν ότι οποιαδήποτε μορφή είναι σε χρήση, παράγονται συπεπή σχήματα. Δεν είναι δυνατό να χρησιμοποιήσετε μια απλή αρίθμηση pixels για να προσδιορίσετε τα μεγέθη των σχημάτων, έτσι οι γραφικές διαδικασίες χρησιμοποιούν μία αυθαίρετη κλίμακα και σύστημα συντεταγμένων για να προσδιορίζουν τα μεγέθη και τις θέσεις των σχημάτων.

Οι γραφικές διαδικασίες χρησιμοποιούν το σύστημα γραφικών συντεταγμένων, π.χ. να σχεδιάζουν σχετικά με την αρχή των γραφικών που είναι στην κάτω αριστερή γωνία του συγκεκριμένου ή default παραθύρου. Σημειώστε ότι αυτό δεν είναι το ίδιο με την αρχή των pixels που χρησιμοποιείται για να ορίσει τη θέση των παράθυρων και των blocks κ.τ.λ. Η αρχή των γραφικών επιτρέπει να χρησιμοποιηθεί ένα καθορισμένο σύστημα Καρτεσιανών συντεταγμένων. 'Ενας δρομέας γραφικών ενημερώνεται μετά από κάθε γραφική λειτουργία. Οι ακόλουθες λειτουργίες μπορούν είτε να είναι σχετικές ή ανεξάρτητες με αυτόν τον δρομέα, π.χ. σχετικά με την αρχή των γραφικών.




Το σύστημα γραφικών συντεταγμένων

Ο συντελεστής κλίμακας είναι τέτοιος ώστε η πλήρης απόσταση στην κάθετη διεύθυνση στο συγκεκριμένο ή default παράθυρο έχει το μήκος 100 από την αρχή και μπορεί να αλλαχθεί με τη διαταγή SCALE. Η κλίμακα στο x είναι ίση με την κλίμακα στη διεύθυνση y. Παρ' όλα αυτά, το μήκος της γραμμής που μπορεί να γραφεί στη διεύθυνση x εξαρτάται από το σχήμα του παραθύρου. Αυξάνοντας το συντελεστή κλίμακας αυξάνεται το μέγιστο σχήμα του σχήματος που μπορεί να σχεδιαστεί προτού να ξεπεραστεί το μέγεθος του παράθυρου. Αν το γραφικό εξαγόμενο αλλάξει σε ένα διαφορετικό μέγεθος παράθυρου τότε το επόμενο μέγεθος του εξαγόμενου κανονίζεται για να ταιριάξει με το νέο παράθυρο. Αν ένα σχήμα υπερβαίνει το εξαγόμενο παράθυρο του τότε το σχήμα περικόπτεται.

Είναι χρήσιμο να πάρετε το παράθυρο σαν ένα παράθυρο σε ένα μεγαλύτερο γραφικό διάστημα μέσα στο οποίο σχεδιάζονται τα σχήματα. Η διαταγή SCALE επιτρέπει να τοποθετηθεί η αρχή γραφικών για να επιτραπεί να μετακινηθεί το παράθυρο γύρω από το γραφικό χώρο.

Το εξαγόμενο των γραφικών διαδικασιών στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι και το εξαγόμενο γράφεται στο χρώμα INK για εκείνο το κανάλι.

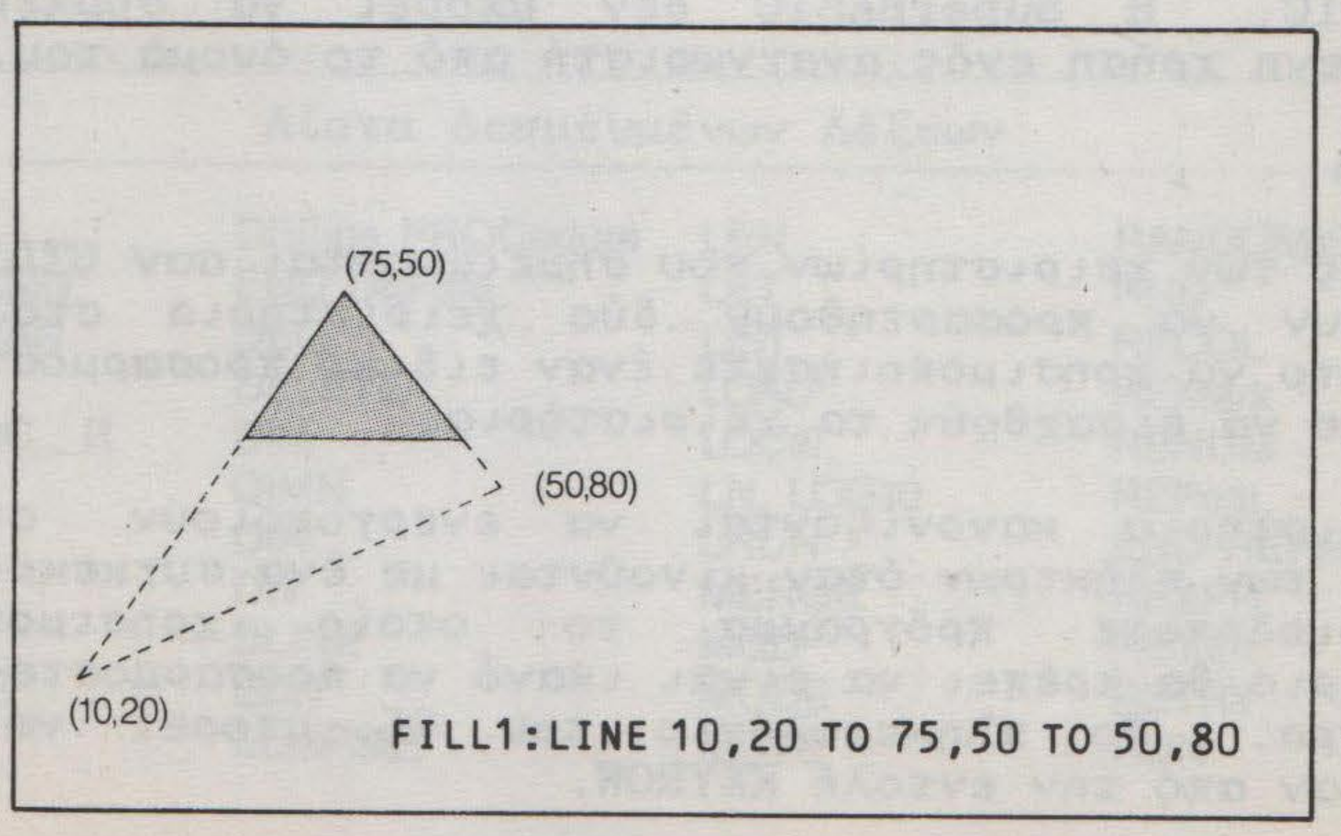
| Εντολή | Λειτουργία | |
|---|---|---|
| CIRCLE | Σχεδιάζει μία έλλειψη ή έναν κύκλο | |
| LINE | Σχεδιάζει μία γραμμή | |
| ARC | Σχεδιάζει ένα τόξο ενός κύκλου | απόλυτα |
| POINT | Σημειώνει ένα σημείο | |
| CIRCLE_R | Σχεδιάζει μία έλλειψη ενός κύκλου | |
| LINE_R | Σχεδιάζει μία γραμμή | |
| ARC_R | Σχεδιάζει το τόξο ενός κύκλου | σχετική |
| POINT_R | Σημειώνει ένα σημείο | |
| SCALE | Τοποθετεί την κλίμακα και μετακινεί την αρχή | |
| FILL | Γεμίζει ένα σχήμα | |
| CURSOR | Τοποθετεί το κείμενο | |

**Γέμισμα των γραφικών**

Τα σχήματα γράφονται με τις διαδικασίες των γραφικών και των (turtle) γραφικών χελώνας και μπορούν ανάλογα με την επιθυμία σας να "γεμιστούν" με μία συγκεκριμένη διάστιξη ή χρώμα. Αν η FILL επιλέγεται τότε το σχήμα γεμίζεται καθώς σχεδιάζονταν.

Ο αλγόριθμος FILL αποθηκεύει έναν κατάλογο από σημεία που πρόκειται να σημειωθούν παρά να σχεδιαστούν. 'Οταν το σχήμα κλείνει υπάρχουν δύο σημεία στην ίδια οριζόντια γραμμή. Αυτά τα δύο σημεία συνδέονται με μία γραμμή στο τωρινό χρώμα μελάνης και η εργασία επαναλαμβάνεται. Το γέμισμα πρέπει πάντα να ξαναεπιλέγεται προτού να σχεδιάσετε ένα νέο σχήμα για να είστε σίγουροι ότι το buffer που χρησιμοποιείτε για να φυλάξει τον κατάλογο από τα σημεία ξανατοποθετείται.

Το ακόλουθο διάγραμμα παριστάνει τη FILL:



**Προειδοποίηση**

Υπάρχει ένας περιορισμός εφαρμογής της FILL. Η FILL δεν πρέπει να χρησιμοποιηθεί για επανεισερχόμενα σχήματα (π.χ. ένα σχήμα που είναι κοίλο). Τα επανεισερχόμενα σχήματα θα πρέπει να χωρίζονται σε μικρότερα σχήματα που δεν είναι επανεισερχόμενα και κάθε υπό-σχήμα να γεμίζεται ανεξάρτητα.

## Αναγνωριστής

'Ενας SuperBASIC αναγνωριστής είναι μια ακολουθία από γράμματα, αριθμούς και υπογραμμίσεις.

define: *letter*:= | a..Z
| A..Z

*number*:= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |

*identifier*:= *letter* *[| *letter* | *number* | _ | ]*

example:
i. a
ii. limit_1
iii. current_guess
iv. counter

'Ενας αναγνωριστής θα πρέπει να ξεκινά με ένα γράμμα που ακολουθείται από μία σειρά γραμμάτων, αριθμών και υπογραμμίσεων και μπορεί να είναι ως 255 χαρακτήρες σε μήκος. Τα μικρά και κεφαλαία γράμματα είναι ισοδύναμα.

Οι αναγνωριστές χρησιμοποιούνται στο SuperBASIC σύστημα για να καθορίζονται μεταβλητές, διαδικασίες, συναρτήσεις, βρόχοι επανάληψης, κ.τ.λ.

**Προειδοποίηση**

Κανένα άλλο νόημα δεν μπορεί να αποδοθεί σε έναν αναγνωριστή εκτός από την ικανότητα του να "αναγνωρίζει" τις δομές στη SuperBASIC. Η SuperBASIC δεν μπορεί να συμπεράνει την προτιθέμενη χρήση ενός αναγνωριστή από το όνομά του.

## Χειριστήριο

Οι πόρτες των χειριστηρίων που σημειώνονται σαν CTL1 και CTL2 επιτρέπουν να προσαρτηθούν δύο χειριστήρια στο QL. Είναι απαραίτητο να χρησιμοποιήσετε έναν ειδικό προσαρμοστή για να βοηθήσετε να εισαχθούν τα χειριστήρια.

Τα χειριστήρια κανονίζονται να ενεργοποιούν συγκεκριμένα πατήματα των πλήκτρων όταν κινούνται με ένα συγκεκριμένο τρόπο και οποιοδήποτε πρόγραμμα το οποίο χρησιμοποιεί ένα χειριστήριο θα πρέπει να είναι ικανό να προσαρμοστεί σε εκείνα τα πλήκτρα. Το πληκτρολόγιο του QL μπορεί να διαβαστεί κατευθείαν από την εντολή KEYROW.

| | CTL1 | CTL2 |
|---|---|---|
| Τρόπος | Πλήκτρο | Πλήκτρο |
| πάνω (up) | ο δρομέας επάνω | F4 |
| κάτω (down) | ο δρομέας κάτω | F2 |
| αριστερά (left) | ο δρομέας αριστερά | F1 |
| δεξιά (right) | ο δρομέας δεξιά | F3 |
| πυρ (fire) | διάστημα | F5 |

Παρατήρηση: Οι πόρτες των χειριστηρίων μπορούν να χρησιμοποιηθούν για να προσθέτουν άλλες πιο γενικής χρήσης συσκευές ελέγχου στο QL.

## Δεσμευμένη Λέξη

Οι δεσμευμένες λέξεις της SuperBASIC είναι αναγνωριστές που καθορίζονται στον Οδηγό Αναφοράς των Δεσμευμένων Λέξεων της SuperBASIC. Οι δεσμευμένες λέξεις έχουν την ίδια μορφή όπως και ένας αναγνωριστής SuperBASIC. Η έννοια της δεσμευμένης λέξης δεν έχει σημασία. Οι δεσμευμένες λέξεις αντηχούν σαν ένας συνδυασμός κεφαλαίων και μικρών γραμμάτων και πάντα αναπαράγονται πλήρως. Το μέρος των κεφαλαίων δείχνει το ελάχιστο που απαιτείται για να πληκτρολογηθεί για να αναγνωρίσει η SuperBASIC τη δεσμευμένη λέξη.

Το σύνολο των δεσμευμένων λέξεων μπορεί να επεκταθεί προσθέτοντας διαδικασίες στο σύστημα και φορτώνοντας τα στο σύστημα του QL. Είναι καλή ιδέα να καθορίζετε πάντα τις διαδικασίες που πρόκειται να χρησιμοποιηθούν με αυτόν τον τρόπο. Τα ονόματα των διαδικασιών σε κεφαλαία θα εμφανίζονται πάντα έτσι από την SuperBASIC και αυτό θα βοηθήσει να προσδιορίσετε την ειδική τους λειτουργία μέσα στο πρόγραμμα και το QL σύστημα.

## Προειδοποίηση

Οι ήδη υπάρχουσες δεσμευμένες λέξεις δεν μπορούν να χρησιμοποιηθούν σαν κανονικοί αναγνωριστές μέσα σε ένα SuperBASIC πρόγραμμα. Οι SuperBASIC δεσμευμένες λέξεις είναι:

Λίστα Δεσμευμένων Λέξεων

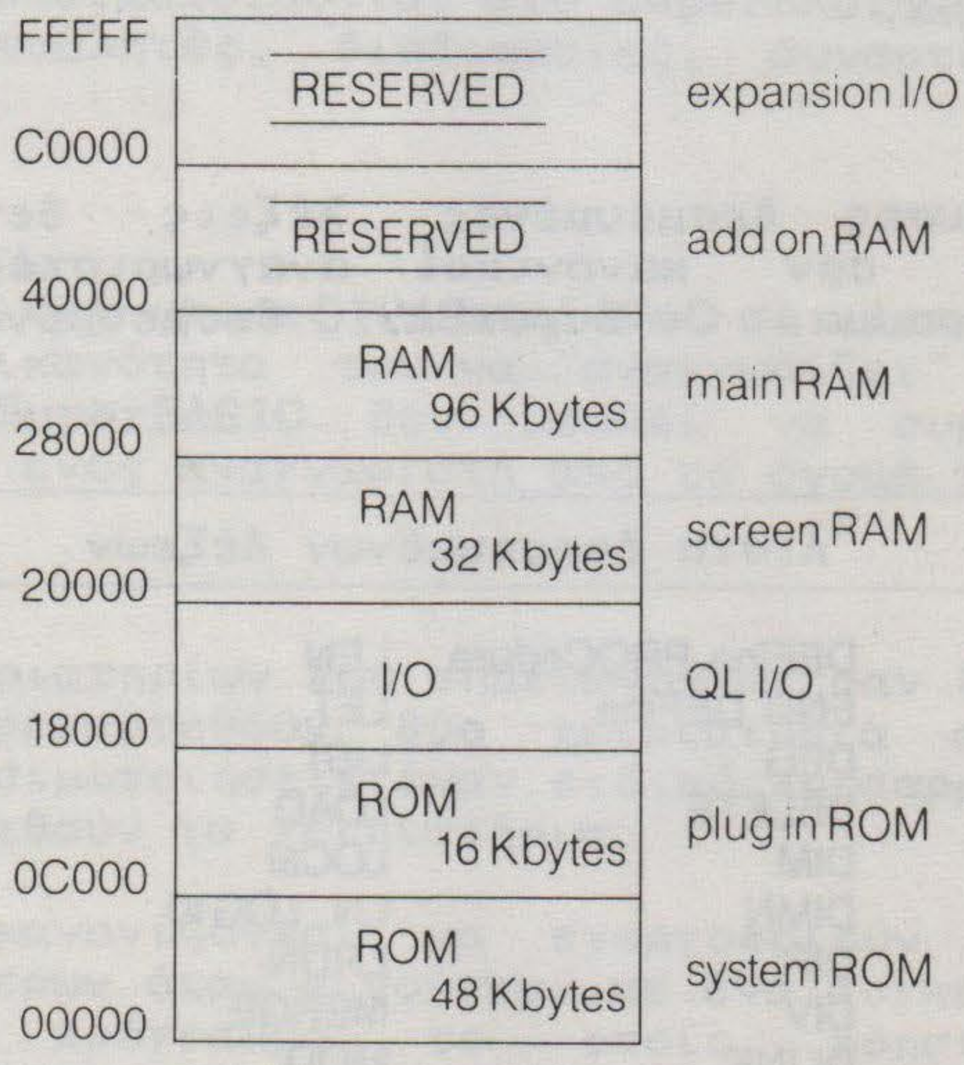
| | | | |
|---|---|---|---|
| ABS | DEFine PROCedure | LEN | RANDOMISE |
| ACOS, ASIN | END DEFine | LET | RND |
| ACOT, ATAN | DEG | LIST | RECOL |
| ADATE | DELETE | LOAD | REMark |
| ARC, ARC_R | DIM | LOCal | RENUM |
| AT | DIMN | LN, LOG10 | REPeat, |
| AUTO | DIR | LRUN | END REPeat |
| BAUD | DIV | MERGE | RESPR |
| BEEP | DLINE | MOD | RETurn |
| BEEPING | EDIT | MODE | RETRY |
| BLOCK | ELLIPSE, | MOVE | RUN |

## Λίστα Δεσμευμένων Λέξεων

| | | | |
|---|---|---|---|
| BORDER | ELLIPSE_R | MRUN | SAVE |
| CALL | EOF | NET | SIN |
| CHR$ | EXEC, EXEC_W | NEW | SCALE |
| CIRCLE | EXIT | NEXT | SCROLL |
| CIRCLE_R | EXP | ON GO TO | SDATE |
| CLEAR | FILL | ON GO SUB | SELect |
| CLOSE | FILL$ | OPEN, OPEN_IN | END SELect |
| CLS | FLASH | OPEN_NEW | SEXEC |
| CODE | FOR | OVER | SQRT |
| CONTINUE | END FOR | PAN | STOP |
| RETRY | FORMAT | PAPER | STRIP |
| COPY, COPY_N | GO SUB | PAUSE | TAN |
| COS | GO TO | PEEK, PEEK_W | TO |
| COT | IF, THEN, ELSE | PEEK_L | TURN |
| CSIZE | END IF | PENUP | TURN TO |
| CURSOR | INK | PENDOWN | UNDER |
| DATA, READ, | INKEY$ | PI | VER$ |
| RESTORE | INPUT | POINT, POINT_R | WIDTH |
| DATE$, DATE | INSTR | POKE, POKE_W | WINDOW |
| DAY$ | INT | POKE_L | |
| DEFine FuNction, | KEYROW | PRINT | |
| END DEFine | LBYTES | RAD | |

## Χάρτης της μνήμης

Το QL περιέχει ένα μικροεπεξεργαστή Motorola 68008, που μπορεί να προσπελάσει 1 Megabyte μνήμης π.χ. από το 00000 ως FFFFF HEX. Η χρήση των διευθύνσεων μέσα σε αυτήν την κλίμακα καθορίζονται από τη Sinclair Research και είναι ως ακολούθως:




Χάρτης φυσικής μνήμης

Η οθόνη RAM οργανώνεται σαν μια σειρά από δεκαέξη bit λέξεις που αρχίζουν από τη διεύθυνση Hex 20000 και προοδευτικά με τη σειρά ψαξίματος π.χ. από τα δεξιά στα αριστερά με κάθε σειρά οθόνης και μετά από το πάνω μέρος ως το κάτω της εικόνας. Τα bits μέσα σε κάθε λέξη οργανώνονται έτσι ώστε ένα pixel προς τα αριστερά να είναι πάντα πιο σημαντικό από ένα pixel προς τα δεξιά (π.χ. ένα μοντέλο pixel στην οθόνη δείχνει το ίδιο σαν το δυαδικό μοντέλο). Παρ' όλα αυτά η οργάνωση της χρωματικής πληροφορίας στις δύο μορφές οθόνης είναι διαφορετική:

| high byte AO=0 | low byte AO=1 | mode |
|---|---|---|
| GGGGGGGG | RRRRRRRR | 512 mode (high res) |
| GFGFGFGF | RBRBRBRB | 256 mode (low res) |

G—green B—blue R—red F—flash

Τοποθετώντας το bit αναβοσβήματος συνδέει την κατάσταση αναβοσβήματος, παγώνει το χρώμα υποβάθρου για το αναβόσβημα στη δοσμένη τιμή από τα R, G και B για εκείνο το pixel. Το αναβόσβημα πάντα επανατοποθετείται στην αρχή κάθε σειράς εμφάνισης.

Σε μορφή υψηλής διακριτικότητας το κόκκινο και το πράσινο όταν καθορίζονται μαζί ερμηνεύεται από τα μηχανήματα σαν άσπρο.

**Προειδοποίηση**

Η χρήση των φυλαγμένων περιοχών στο χάρτη της μνήμης μπορούν να προκαλέσουν ασυμβατότητα με τα μελλοντικά προιόντα της Sinclair. Νοθευμένα εξαγόμενα σε διευθύνσεις καθοριζόμενες να είναι περιφερειακές, I/O διευθύνσεις μπορούν να προκαλέσουν μια απροσδιόριστη συμπεριφορά. Συνιστάται αυτές οι περιοχές να μη γράφονται και να μη χρησιμοποιούνται για κανένα άλλο λόγο. Αν ενοχλήσετε περιοχές που χρησιμοποιούνται σαν Microdrive buffers μπορείτε να καταστρέψετε Microdrive δεδομένα και μπορεί να έχει σαν αποτέλεσμα το χάσιμο πληροφοριών. Αν ενοχλήσετε περιοχές που χρησιμοποιούνται σαν πίνακες του συστήματος μπορεί να προκαλέσετε το σπάσιμο του συστήματος και μπορεί να προκαλέσετε το χάσιμο δεδομένων και προγραμμάτων.

'Ενα I/O μπορεί να εκτελεστεί χρησιμοποιώντας είτε τις σχετικές SuperBASIC εντολές ή τις παγίδες του λειτουργικού συστήματος Qdos.

## Μαθηματικές συναρτήσεις

Η SuperBASIC έχει τις πιο βασικές τριγωνομετρικές και μαθηματικές συναρτήσεις.

| Συνάρτηση | 'Ονομα |
|---|---|
| COS | Συνημίτονο |
| SIN | Ημίτονο |
| TAN | Εφαπτομένη |
| ATAN | Τόξο εφαπτομένης |
| ACOT | Τόξο συνεφαπτομένης |
| ACOS | Τόξο συνημιτόνου |
| ASIN | Τόξο ημιτόνου |
| COT | Συνεφαπτομένη |
| EXP | Εκθετική |
| LN | Φυσικός λογάριθμος |
| LOG10 | Δεκαδικός λογάριθμος |
| INT | Ακέραιος |
| ABS | Απόλυτη τιμή |
| RAD | Μετατροπή σε ακτίνια |
| DEG | Μετατροπή σε μοίρες |
| PI | Τιμή του π |
| RND | Δημιουργία τυχαίου αριθμού |
| RANDOMISE | Αρχή δημιουργού τυχαίων αριθμών |

## Microdrives

Τα Microdrives προμηθεύουν την κύρια σταθερή αποθήκη στο QL. Κάθε μικροκασέτα του Microdrive έχει χωρητικότητα τουλάχιστον 100 Kbytes. Ελεύθερη διαθέσιμη μνήμη κατανέμεται από το Qdos στα Microdrive buffers όταν είναι απαραίτητο για να καλυτερέψουν την εκτέλεση.

Κάθε κενή μικροκασέτα θα πρέπει να φορμάρεται πριν από τη χρήση και μπορεί να κρατήσει ως 255 τομείς με 512 bytes ανά τομέα. Το Qdos φυλάσσει ένα πίνακα περιεχομένων των αρχείων που είναι αποθηκευμένα στη μικροκασέτα. Κάθε Microdrive αρχείο αναγνωρίζεται χρησιμοποιώντας ένα στάνταρτ SuperBASIC αρχείο ή όνομα συσκευής.

Μία μικροκασέτα μπορεί να προστατευτεί από το γράψιμο με τη μετακίνιση του μικρού πιασίματος στη δεξιά μεριά.

'Οταν δέχεστε καινούριες κενές QL μικροκασέτες να τις φορμάρετε μερικές φορές για να γίνουν σωστές.

### Γενική φροντίδα

Φυσικά, κάθε μικροκασέτα περιέχει ένα βρόχο 200 ιντσών ταινίας video υψηλής ποιότητας το οποίο κινείται με ταχύτητα 28 ίντσες το δευτερόλεπτο. Η ταινία συμπληρώνει μια περιστροφή κάθε 7 1/2 δευτερόλεπτα.

- Μην ακουμπήσετε ποτέ την ταινία με τα δάκτυλα σας ή να εισάγετε κάτι στη μικροκασέτα.
- Ποτέ μην ανοίξετε ή κλείσετε τον υπολογιστή με τις μικροκασέτες στη θέση τους.
- Πάντα να φυλάτε τις μικροκασέτες στις θήκες όταν δεν είναι σε χρήση.

- Πάντα να εισάγετε ή να μετακινείτε τις μικροκασέτες από το Microdrive αργά και προσεκτικά.

- Πάντα να είστε σίγουροι ότι η μικροκασέτα είναι σταθερά τοποθετημένη πριν αρχίσετε το Microdrive.

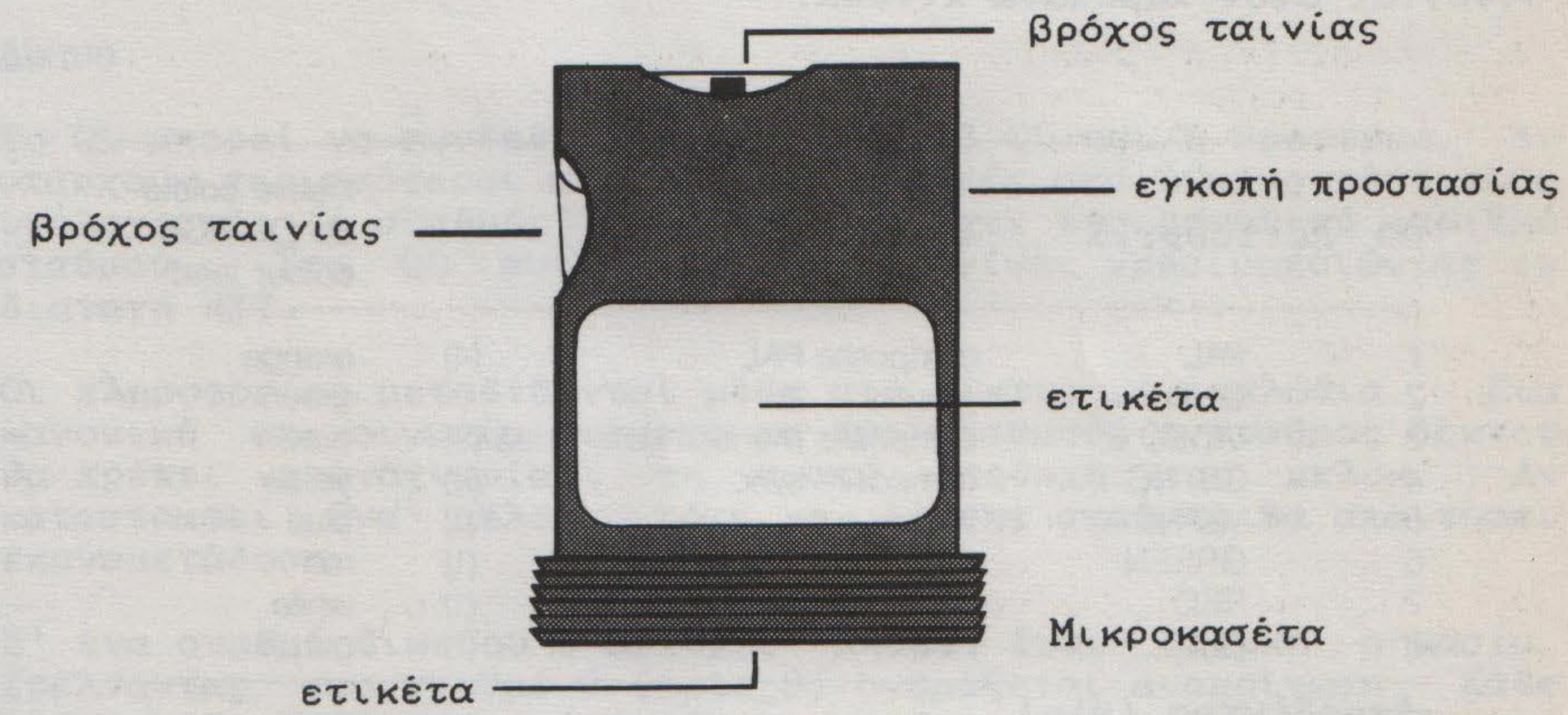
- Ποτέ μην κινείτε το QL με τις μικροκασέτες τοποθετημένες ακόμα κι αν δεν είναι σε ενέργεια.

- Ποτέ μην ακουμπάτε τη μικροκασέτα ενώ είναι σε χρήση.

- Μην εισάγετε και μετακινείτε τις μικροκασέτες επανειλημένα χωρίς να τρέχει το Microdrive.

**Βρόχοι ταινίας**

Αν εμφανιστεί ένας βρόχος στην ταινία σε μία από τις δύο θέσεις που φαίνονται στο σχήμα 1, τότε απαλά σπρώξτε τη μικροκασέτα. Χρησιμοποιήστε ένα εργαλείο χωρίς ύφασμα π.χ. την πλευρά ενός μολυβιού ή στυλό. Ποτέ μην ακουμπάτε την ταινία με τα δάκτυλα σας γι' αυτόν ή για κάποιο άλλο λόγο.




| Διαταγή | Λειτουργία |
|---|---|
| FORMAT | φορμάρει μια μκροκασέτα για χρήση |
| DELETE | διαγράφει ένα αρχείο μία μικροκασέτα |
| DIR | δίνει τον πίνακα περιεχομένων μιας μικροκασέτας |
| SAVE<br>SBYTES<br>SEXEC | φυλάει δεδομένα από μία μικροκασέτα |
| LOAD<br>LBYTES<br>EXEC<br>MERGE | φορτώνει δεδομένα από μία μικροκασέτα |

| | |
|---|---|
| OPEN_IN<br>OPEN_NEW<br>OPEN<br>CLOSE | ανοίγει και κλείνει αρχεία |
| PRINT<br>INPUT<br>INKEY$ | ανοίγει και κλείνει αρχεία |

**Προειδοποίηση**

Αν προσπαθήσετε να γράψετε σε μια προστατευμένη μικροκασέτα το QL μετά από πολλές προσπάθειες θα δώσει το μήνυμα λάθους "bad medium".

## Οθόνη

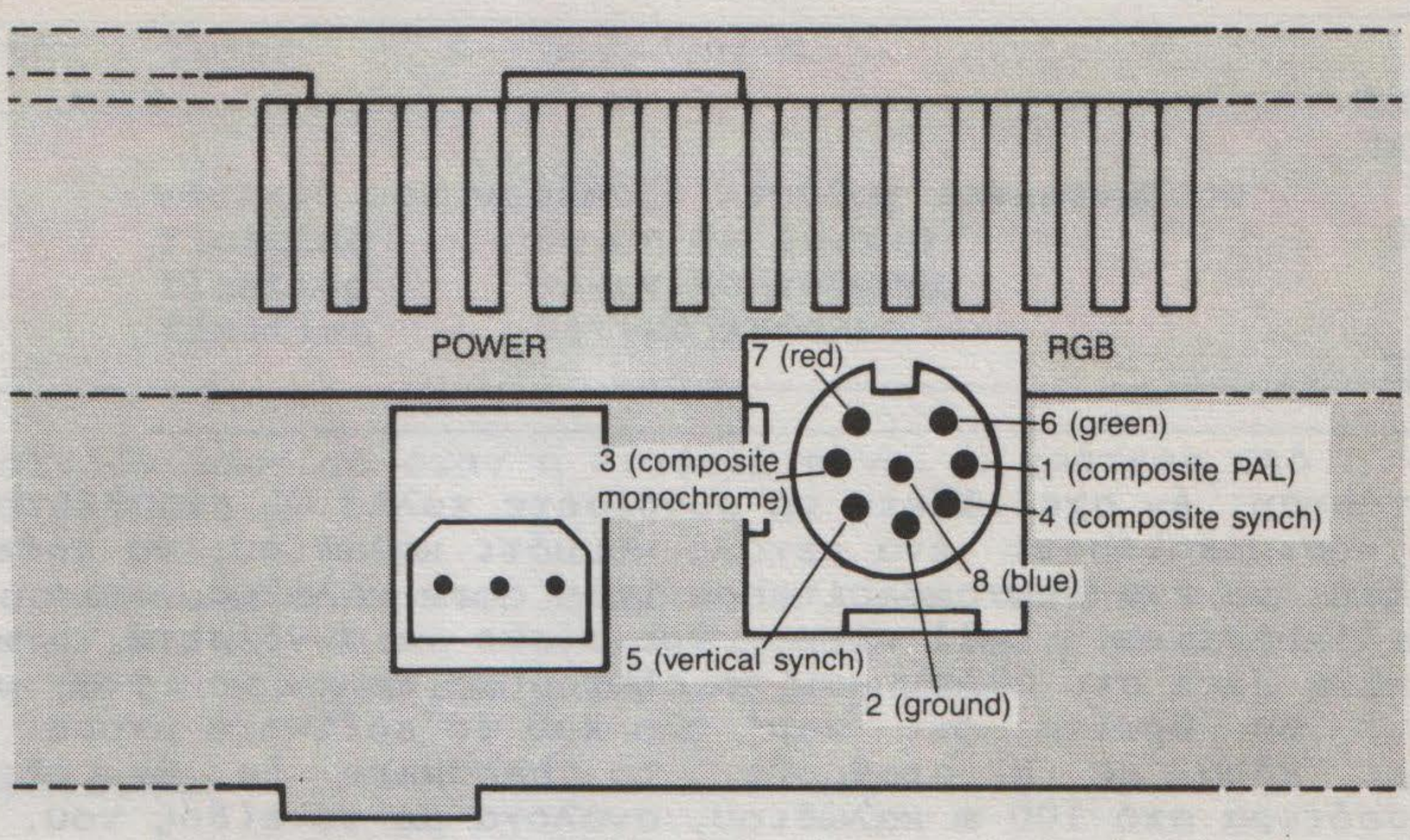
Μια οθόνη μπορεί να συνδεθεί με το QL διαμέσου της RGB πρίζας στο πίσω μέρος του υπολογιστή. Η σύνδεση γίνεται διαμέσου μιας 8-way DIN πρίζας και καλωδίου για έγχρωμη οθόνη ή 3-way DIN πρίζας και καλωδίου για μονόχρωμη οθόνη. Οι συνδέσεις δίνονται στον παρακάτω πίνακα.

| pin | Λειτουργία | Σήμα | | sleeve colour on QL RGB colour lead |
|---|---|---|---|---|
| 1 | PAL | composite PAL | (4) | orange |
| 2 | GND | ground | | green |
| 3 | VIDEO | composite monocrhome video | (3) | brown |
| 4 | CSYNC | composite sync | (2) | yellow |
| 5 | VSYNC | vertical sync | (1) | blue |
| 6 | GREEN | green | (1) | red |
| 7 | RED | red | (1) | white |
| 8 | BLUE | blue | (1) | purple |

**Ακροδέκτης (pin)**

Μια μονόχρωμη οθόνη μπορεί να συνδεθεί χρησιμοποιώντας ένα screened lead με μια 3 way ή μια 8-way DIN πρίζα στο τέλος του QL. Η σύνδεση στο άκρο της οθόνης θα ποικίλλει ανάλογα με την οθόνη αλλά είναι συνήθως μία phono πρίζα. Η οθόνη θα πρέπει να έχει ένα 75 ohm 1V pk-pk σύνθετο video μη αντιστρέψιμο εισαγόμενο (το οποίο είναι το στάνταρ της βιομηχανίας). Και οι δύο 3-way DIN και phono πρίζες είναι συχνά διαθέσιμες στα καταστήματα ηλεκτρονικών.

Μία RGB (έγχρωμη) οθόνη μπορεί να συνδεθεί χρησιμοποιώντας μια έξοδο με μια 8-way DIN πρίζα στο άκρο του QL. Η σύνδεση στο άκρο της οθόνης θα ποικίλλει ανάλογα με την οθόνη (δεν υπάρχει ένα βιομηχανικό standard) και συνήθως τροφοδοτείται με αυτή. 'Ενα κατάλληλο καλώδιο είναι διαθέσιμο με μια 8-way DIN πρίζα από τη Sinclair Research Limited.




Διάγραμμα συνδεσμολογίας του Μόνιτορ

## Δίκτυο

Το QL μπορεί να συνδεθεί με έως άλλα 63 QL και/ή Spectrum. Αν υπάρχουν περισσότεροι από δύο υπολογιστές στο δίκτυο τότε κάθε υπολογιστής (ή σταθμός) θα πρέπει να έχει ένα μοναδικό αριθμό σταθμού. Στο QL αυτό μπορεί να γίνει χρησιμοποιώντας τη διαταγή NET.

Οι πληροφορίες μεταδίδονται μέσα στο δίκτυο με καλώδια. Για κανονική επικοινωνία ανάμεσα σε δύο σταθμούς ο σταθμός δέκτης θα πρέπει να αναγνωρίσει τη σωστή υποδοχή του μπλοκ. Αν καταστραφεί ένα μπλοκ, τότε ο δέκτης σταθμός θα απαιτήσει επαναμετάδοση.

Σ' ένα σταθμό δικτύου ο αριθμός μηδέν έχει ειδική σημασία. Στέλνοντας στο σταθμό 0 (neto_0) ονομάζεται ανακοίνωση. Κάθε μήνυμα που στέλνεται μ' αυτόν τον τρόπο μπορεί να διαβαστεί από κάθε σταθμό ο οποίος ακούει στο neti_0.

'Ενας σταθμός του δικτύου που βρίσκεται σε κατάσταση ακρόασης (π.χ. NET3:LOAD neti_3) μπορεί να λάβει δεδομένα από κάθε σταθμό.

| Διαταγή | Λειτουργία |
|---|---|
| NET | ορίζει έναν αριθμό σταθμού δικτύου |
| OPEN | ανοίγει ένα κανάλι δικτύου |
| CLOSE | κλείνει ένα κανάλι δικτύου |
| PRINT<br>INPUT<br>INKEY$ | δίκτυο I/O |

| | |
|---|---|
| LOAD<br>SAVE<br>LBYTES<br>SBYTES<br>EXEC<br>SEXEC<br>LRUN<br>MRUN<br>MERGE | φορτώνει και φυλάσει διαμέσου του δικτύου |

Παρατήρηση: Αν σχεδιάζετε να συνδέσετε πολλά QL στο δίκτυο, ή να χρησιμοποιήσετε ένα μεγάλο κομμάτι καλωδίου, θα πρέπει να τυλίξετε με ένα low-capacitance twin core καλώδιο, όπως το 3 amp light-flex ή bell-wire. Φροντίστε να συνδέσετε τα κέντρα του κάθε jack στο άλλο, και τα εξωτερικά τους το ένα με το άλλο. θα βρείτε ότι παρ' όλο που το software μπορεί να τα βγάλει πέρα με N σταθμούς, το hardware δε θα οδηγήσει περισσότερα από 100 m καλώδιου, ανάλογα με το είδος του.

Αν συνδέσετε μόνο λίγες μηχανές με τις τροφοδοτημένες leads, δε χρειάζετε να ανησυχείτε.

## Τελεστές

| Τελεστής | Τύπος | Λειτουργία |
|---|---|---|
| _____ | floating<br>string | λογικό τύπος 2 σύγκριση |
| == | αριθμητικό<br>string | σχεδίου ίσο ** (τύπος 3 σύγκριση |
| + | αριθμητικό | πρόσθεση |
| - | αριθμητικό | αφαίρεση |
| / | αριθμητικό | διαίρεση |
| * | αριθμητικό | πολλαπλασιασμός |
| < | αριθμητικό<br>string | μικρότερο από (τύπος 2 σύγκριση) |
| > | αριθμητικό<br>string | μεγαλύτερο από (τύπος 2 σύγκριση) |
| <= | αριθμητικό<br>string | μικρότερο από ή ίσο με (τύπος 2 σύγκριση) |
| >= | αριθμητικός<br>string | μεγαλύτερο από ή ίσο με (τύπος 2 σύγκριση) |
| <> | αριθμητικός<br>string | όχι ίσο με (τύπος 3 σύγκριση) |
| & | string | αλληλουχία |
| && | bitwise | AND |
| ¦¦ | bitwise | OR |
| ^^ | bitwise | XOR |
| ~ | bitwise | NOT |
| OR | λογικό | OR |
| AND | λογικό | AND |
| XOR | λογικό | XOR |
| NOT | λογικό | NOT |
| MOD | ακέραιο | υπόλοιπο |
| DIV | ακέραιο | διαίρεση ακεραίων |

| Τελεστής | Τύπος | Λειτουργία |
|---|---|---|
| INSTR | string | τύπος 1 string σύγκριση |
| ^ | floating | ύψωση σε δύναμη |
| - | floating | αρνητικό πρόσημο |
| + | floating | θετικό πρόσημο |

** σχεδόν ίσο-ίσο όταν η διαφορά είναι μικρότερη από 0.0000001.

Αν η συγκεκριμένη λογική λειτουργία είναι αληθής τότε μια τιμή μη ίση με το μηδέν θα επιστραφεί. Αν η λειτουργία είναι ψευδής θα επιστραφεί μια τιμή ίση με μηδέν.

**Προτεραιότητα**

Η προτεραιότητα των τελεστών της SuperBASIC ορίζεται στον πίνακα 1. Αν η σειρά της ισοδυναμίας σε μία έκφραση δεν μπορεί να εξαχθεί από αυτόν τον πίνακα τότε οι σχετικές λειτουργίες εκτελούνται από τα αριστερά προς τα δεξιά. Η προτεραιότητα των SuperBASIC τελεστών μπορεί να παραληφθεί με το να εσωκλείσετε τα σχετικά μέρη της έκφρασης σε παρενθέσεις.

| | |
|---|---|
| Υψηλότερη | αρνητικό και θετικό πρόσημο |
| | string αλληλουχία |
| | INSTR |
| | ύψωση σε δύναμη |
| | πολλαπλασιασμός, διαίρεση, υπόλοιπο και |
| | ακέραια διαίρεση |
| | πρόσθεση και αφαίρεση |
| | λογική σύγκριση |
| | NOT (bitwise ή λογική) |
| | AND (bitwise ή λογική) |
| χαμηλότερο | OR και XOR (bitwise ή λογική) |

**Περιφερειακή επέκταση**

Η σύνδεση επέκτασης επιτρέπει να συνδεθούν επιπλέον περιφερειακά να συνδεθούν στο QL. Οι διαθέσιμες συνδέσεις στη σύνδεση είναι:

| | | | | |
|---|---|---|---|---|
| GND | a | 1 | b | GND |
| D3 | a | 2 | b | D2 |
| D4 | a | 3 | b | D1 |
| D5 | a | 4 | b | D0 |
| D6 | a | 5 | b | ASL |
| D7 | a | 6 | b | DSL |
| A19 | a | 7 | b | RDWL |
| A18 | a | 8 | b | DTACKL |
| A17 | a | 9 | b | BGL |
| A16 | a | 10 | b | BRL |
| CLKCPU | a | 11 | b | A15 |
| RED | a | 12 | b | RESETCPUL |
| A14 | a | 13 | b | CSYNCL |
| A13 | a | 14 | b | E |
| A12 | a | 15 | b | VSYNCH |
| A11 | a | 16 | b | VPAL |
| A10 | a | 17 | b | GREEN |
| A9 | a | 18 | b | BLUE |
| A8 | a | 19 | b | FC2 |
| A7 | a | 20 | b | FC1 |
| A6 | a | 21 | b | FC0 |
| A5 | a | 22 | b | A0 |
| A4 | a | 23 | b | ROMOEH |
| A3 | a | 24 | b | A1 |
| DBGL | a | 25 | b | A2 |
| SP2 | a | 26 | b | SP3 |
| DSMCL | a | 27 | b | IPLOL |
| SP1 | a | 28 | b | BERRL |
| SP0 | a | 29 | b | IPL1L |
| VP12 | a | 30 | b | EXTINTL |
| VM12 | a | 31 | b | VIN |
| VIN | a | 32 | b | VIN |

'Ενα "L" στο τέλος σε ένα όνομα σήματος σημαίνει ότι το σήμα είναι ενεργό χαμηλά.

| Σήμα | Λειτουργία |
|---|---|
| A0..A19 | 68008 address lines |
| RDWL | Read / Write |
| ASL | Address Strobe |
| DSL | Data Strobe |
| BGL | Bus Grant |
| DSMCL | Data Strobe - Master Chip |
| CLKCPU | CPU Clock |
| E | 6800 peripherals clock |
| RED | Red |
| BLUE | Blue |
| GREEN | Green |
| CSYNCL | Composite Sync |
| VSYNCH | Vertical Sync |
| ROMOEH | ROM Output Enable |
| FC0 | Processor Status |
| FC1 | Processor Status |
| FC2 | Processor Status |
| RESETCPUL | Reset CPU |

Σήματα εξόδου (output) περιφερειακών QL.

| Σήμα | Λειτουργία |
|---|---|
| DTACKL | Data acknowledge |
| BRL | Bus request |
| VPAL | Valid Peripheral Address |
| IPL0L | Interrupt Priority Level 5 |
| IPL1L | Interrupt Priority Level 2 |
| BERRL | Bus Error |
| EXTINTL | External Interrupt |
| DBGL | Data bus grab |
| D0..D7 | Data Lines |

QL Peripheral Bi-directional Signals

| **Signal** | **Function** |
|---|---|
| SP0..SP3 | Select peripheral 0 to 3 |
| VIN | 9V DC (nominal) - 500mA maximum |
| VM12 | –12V |
| VP12 | +12V |
| GND | ground |

Η ακόλουθη περιγραφή του μηχανισμού της περιφερειακής επέκτασης στο QL δεν προτίθεται να καλύψει μια πραγματική συσκευή επέκτασης, αλλά περισσότερο να διαβαστεί για να αποκτήσετε μια βασική κατανόηση του μηχανισμού επέκτασης.

Τα περιφερειακά μπορούν να προστεθούν στο QL είτε μόνα τους είτε πολλαπλά, ως το λογικό μέγιστο των 16 συσκευών. 'Ενα μοναδικό περιφερειακό μπορεί να τοποθετηθεί στο Slot της επέκτασης του QL, ενώ τα πολλαπλά περιφερειακά θα πρέπει να συνδεθούν στο Module της επέκτασης του QL, το οποίο με τη σειρά του συνδέεται στα Slot της επέκτασης του QL διαμέσου ενός buffer card.

Σε αυτό το περιεχόμενο ο όρος συσκευής περιλαμβάνει την επέκταση μνήμης. Παρ' όλο που οι περιοχές στο χάρτη της μνήμης του QL που εκχωρείται στη μνήμη της επέκτασης είναι διαφορετικές στις συσκευές επέκτασης ο βασικός μηχανισμός είναι ο ίδιος. Μόνο ένα περιφερειακό επέκτασης μνήμης μπορεί να συνδεθεί με το QL σε κάθε φορά. Το κενό διεύθυνσης που εκχωρείται για την περιφερειακή επέκταση στο Φυσικό χάρτη μνήμης του QL επιτρέπει 16 Kbytes ανά περιφερειακό. Αυτή η περιοχή πρέπει να περιέχει την απαιτούμενη I/O μνήμη για το driver και τον κωδικό για το ίδιο το driver.

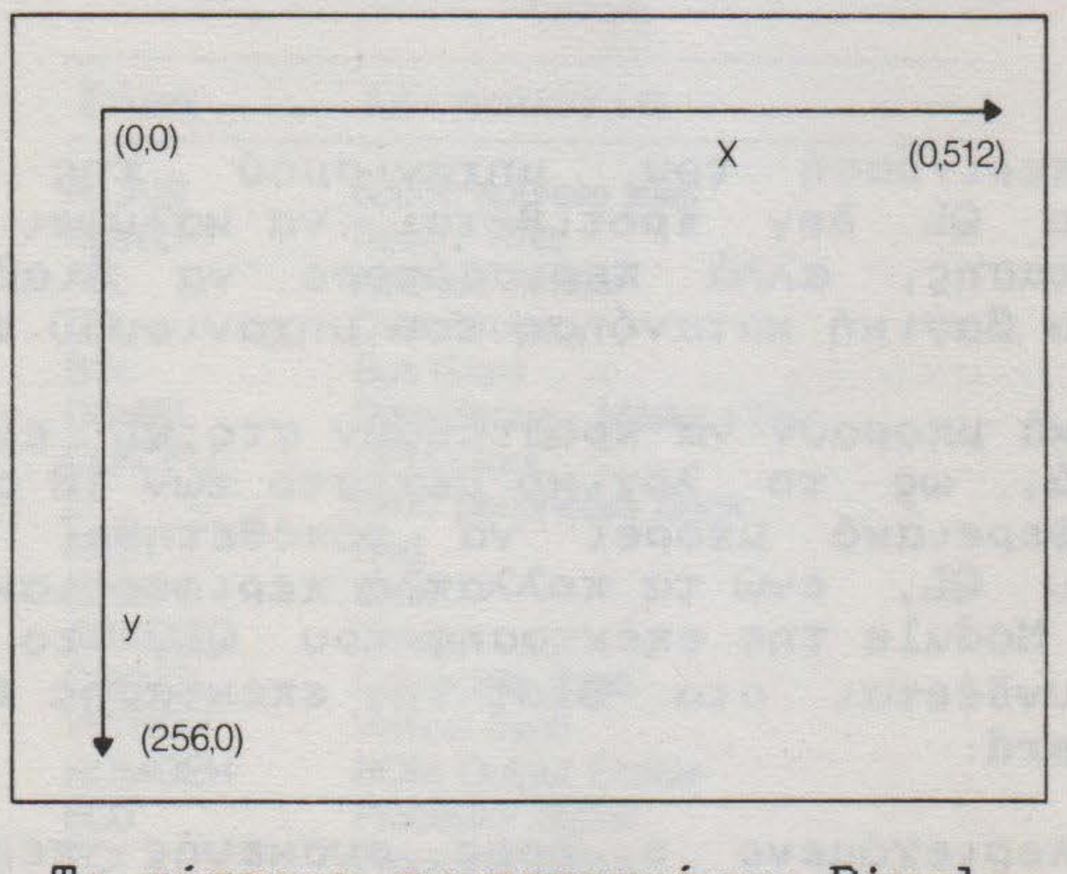
Το Qdos περιλαμβάνει ευκολίες για διαχείριση ουρών και απλό σειριακό I/O που μπορεί να χρησιμοποιηθεί όταν γράφετε τους drivers συσκευής.

Η θέση κάθε περιφερειακής συσκευής στον όλο χάρτη μνήμης του QL καθορίζεται από τις σειρές επιλογής σημάτων: το SP0, SP1, SP2 και SP3. Οι σειρές επιλογής ενεργοποιούν ένα σήμα που αντιστοιχεί στη θέση του slot στο module της επέκτασης του QL, ώστε για μια συσκευή που θα επιλεχθεί το εξαγόμενο διεύθυνσης από τις σειρές διεύθυνσης: τα A14, A15, A16, A17 θα πρέπει να είναι τα ίδια με τα σήματα από τις τρεις σειρές επιλογής ανάλογα.

## Το σύστημα συντεταγμένων pixels

Το σύστημα συντεταγμένων pixels χρησιμοποιείται για να καθορίσει τις θέσεις και τα μεγέθη των παραθύρων, τις θέσεις των τετραγώνων και του δρομέα στην οθόνη του QL. Το σύστημα συντεταγμένων έχει την προέλευση του στην πάνω αριστερή γωνία του default παραθύρου (ή οθόνης) και πάντα υποθέτει ότι οι θέσεις καθορίζονται όπως όταν η οθόνη είναι σε 512 μορφή (μορφή υψηλής διακριτικότητας). Το σύστημα θα χρησιμοποιήσει το κοντινότερο διαθέσιμο στοιχείο για το συγκεκριμένη τοποθέτηση μορφής κάνοντας το σύστημα συντεταγμένων ανεξάρτητο από τη μορφή οθόνης σε χρήση.

Μερικές διαταγές είναι πάντα σχετικές με την προέλευση του default παραθύρου, π.χ. η WINDOW, ενώ άλλες είναι πάντα σχετικές με την τωρινή προέλευση παράθυρου π.χ. η BLOCK.




Το σύστημα συντεταγμένων Pixels

## Πρόγραμμα

'Ενα SuperBASIC πρόγραμμα αποτελείται από μία ακολουθία από SuperBASIC εντολές, όπου κάθε εντολή αρχίζει με έναν αριθμό σειράς. Οι αριθμοί σειράς είναι στην κλίμακα από 1 ως 32767.

| Εντολή | Λειτουργία |
|---|---|
| RUN | αρχίζει ένα φορτωμένο πρόγραμμα |
| LRUN | φορτώνει ένα πρόγραμμα από μία συσκευή και αρχίζει την εκτέλεσή του |
| CTRL SPACE | αναγκάζει ένα πρόγραμμα να σταματήσει |

syntax: *line__number*:= *[*digit*]* {range 1..32767}
*[*line__number statement *[:statement]*]*

example: i.
```
100 PRINT "This is a valid line number"
RUN
```

ii.
```
100 REM a small program
110 FOR foreground = 0 TO 7
120   FOR contrast = 0 TO 7
130     FOR stipple = 0 TO 3
140       PAPER foreground, contrast, stipple
150       CURSOR 0,70
160       FOR n = 0 TO 2
170         SCROLL 2,1
180         SCROLL -2,2
190       END FOR n
200     END FOR stipple
210   END FOR contrast
220 END FOR foreground
RUN
```

## Qdos

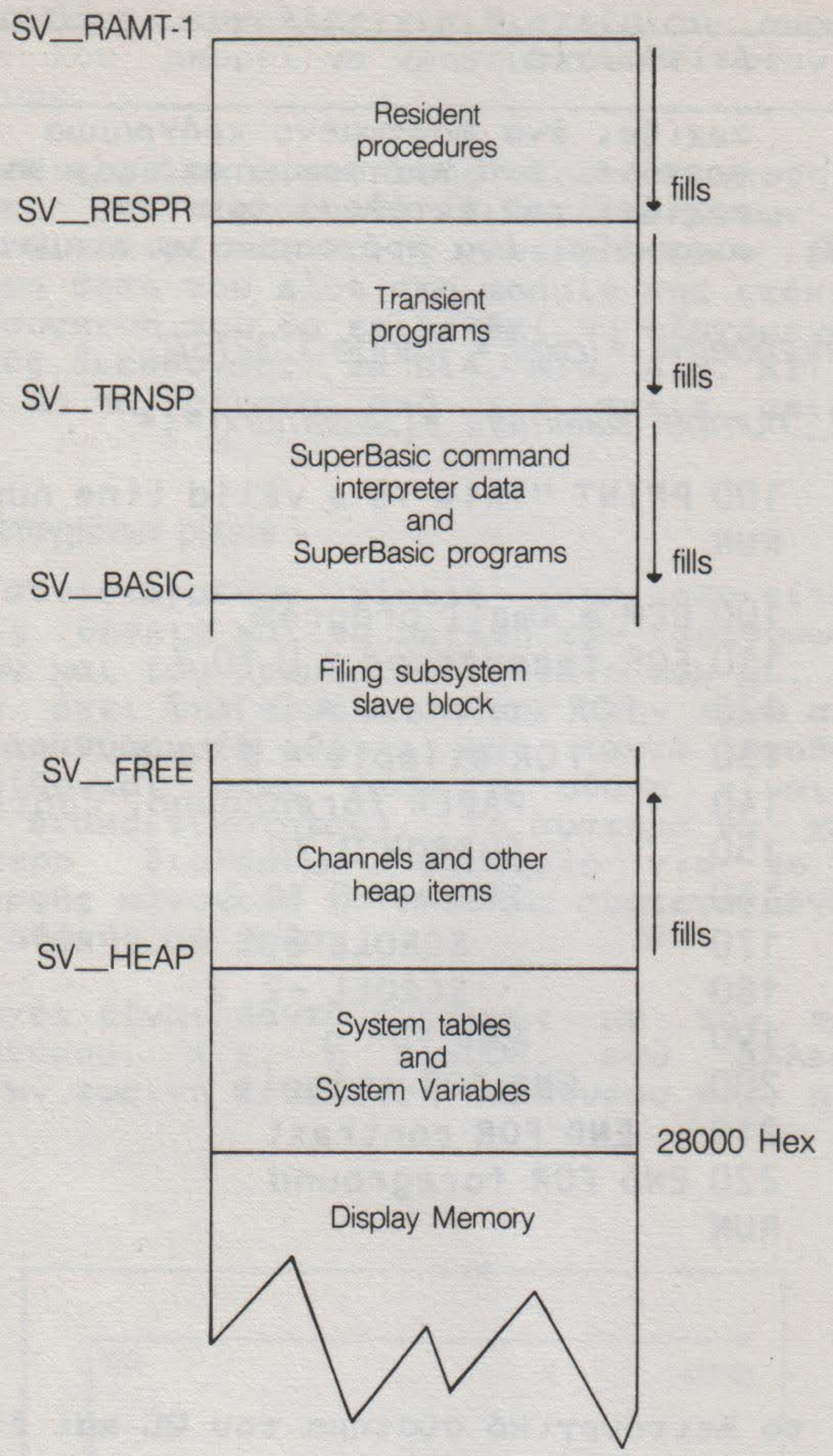
Το Qdos είναι το λειτουργικό σύστημα του QL και διευθύνει:

- Τον προγραμματισμό εργασιών έργων και την εκχώρηση μέσων
- Το I/O της οθόνης (περιλαμβανομένων των παραθύρων)
- Το Microdrive I/O
- Την επικοινωνία δικτύου και σειριακών καναλιών
- Την είσοδο από το πληκτρολόγιο
- Τη διαχείρηση της μνήμης

## Χάρτης μνήμης

Μία πλήρης περιγραφή του Qdos είναι πέρα από το αντικείμενο αυτού του οδηγού αλλά μια σύντομη περιγραφή περιλαμβάνεται.

Το σύστημα RAM έχει μια οργάνωση που επιβάλλεται σε αυτό από το λειτουργικό σύστημα Qdos και καθορίζεται να είναι ως ακολούθως:




Οι όροι SV_RAMT, SV_RESR, SV_TRNSP, SV_BASIC, SV_FREE, SV_HEAP, χρησιμοποιούνται για να αντιπροσωπεύουν διευθύνσεις μέσα στο QL. Αυτοί οι όροι δεν αναγνωρίζονται από τη SuperBASIC ή το Qdos. Επιπλέον, οι διευθύνσεις που αντιπροσωπεύονται είναι δυνατόν να αλλάζουν καθώς το σύστημα τρέχει.

sv_ramt RAM Top

Αυτό θα ποικίλει ανάλογα με τις επεκτάσεις μνήμης που είναι προσαρτημένες στο σύστημα.

sv_respr Resident Procedures

Οι μόνιμες διαδικασίες φορτώνονται στην κορυφή της RAM. 'Οταν φορτωθούν, η εκχώρηση μνήμης μπορεί να τροποποιηθεί μόνο με την επανεκκίνηση του συστήματος. Οι μόνιμες διαδικασίες προστίθενται

στον κατάλογο των ονομάτων της SuperBASIC και έτσι γίνονται επεκτάσεις του SuperBASIC συστήματος.

sv_trnsp Transient Programs

Τα παροδικά προγράμματα φορτώνονται αμέσως κάτω από τα μόνιμα προγράμματα. Κάθε πρόγραμμα θα πρέπει να είναι αυτόνομο, π.χ. να περιέχει χώρο για τα δικά του δεδομένα και το δικό του σωρό (stack). Θα πρέπει να είναι ανεξάρτητο από θέση ή θα πρέπει να φορτωθεί από έναν ειδικά γραμμένο συνδεόμενο φορτωτή (loader). Τα παροδικά προγράμματα εκτελούνται από τη BASIC χρησιμοποιώντας την εντολή EXEC ή από το Qdos με το να το ενεργοποιήσετε σαν μία δουλειά.

Η περιοχή των παροδικών προγραμμάτων μπορεί να χρησιμοποιηθεί για τη φύλαξη των δεδομένων αλλά αυτά τα δεδομένα θα μεταχειρισθούν ακόμα σαν μια εργασία από το Qdos και γι' αυτό δε θα πρέπει να ενεργοποιηθούν.

sv_basic Η SuperBASIC Area

Περιέχει όλα τα φορτωμένα SuperBASIC προγράμματα και τα συγγενικά δεδομένα. Αυτή η περιοχή επεκτείνεται και συστέλλεται χρησιμοποιώντας τον ελεύθερο χώρο όπως απαιτείται.

sv_free Free Space

Ο ελεύθερος χώρος χρησιμοποιείται από το υποσύστημα αρχείων του Qdos για να δημιουργήσει Microdrive Slave Blocks. Π.χ. αντίγραφα των Microdrive μπλοκς που μπορούν να κρατηθούν στη RAM.

sv_heap System Heap

Χρησιμοποιείται από το σύστημα για να αποθηκεύει ορισμούς δεδομένων καναλιών κ.τ.λ. Επίσης προμηθεύει εργασιακή αποθήκη για το I/O υποσύστημα. Τα παροδικά προγράμματα μπορούν να εκχωρούν εργασιακό χώρο για τα ίδια στο σωρό διαμέσου των Qdos καλεσμάτων του συστήματος.

System Tables / System Variables

Αυτή η περιοχή είναι ακριβώς πάνω από τη μνήμη οθόνης. Οι πίνακες συστήματος και ο supervisor stack είναι πάνω από τις μεταβλητές του συστήματος.

## Κλήσεις συστήματος

Οι κλήσεις του συστήματος εκτελούνται από το Qdos σε supervisor μορφή. 'Οταν είναι σε supervisor μορφή το Qdos δε θα επιτρέψει σε καμμία άλλη εργασία να αναλάβει τον επεξεργαστή. Οι κλήσεις του συστήματος που εκτελούνται με αυτόν τον τρόπο λέγονται ότι είναι ατομικές δηλαδή η κλήση συστήματος θα εκτελέσει ως την αποπεράτωση προτού αφήσει τον

επεξεργαστή. Μερικές κλήσεις συστήματος είναι μόνο μερικά ατομικές, δηλαδή μόλις έχουν τελειώσει την κύρια τους λειτουργία θα αφήσουν τον επεξεργαστή, αν είναι απαραίτητο. Εκτός αν απαιτηθεί ειδικά όλες οι I/O κλήσεις συστήματος είναι μερικά ατομικές.

Ο συνηθισμένος μηχανισμός για να γίνει μία κλήση συστήματος είναι με το να κάνετε μία παγίδα σε ένα από τα διανύσματα του συστήματος Qdos με κατάλληλες παραμέτρους στους καταχωρητές του επεξεργαστή. Η δράση από το Qdos που ακολουθεί την κλήση συστήματος είναι εξαρτημένη από τη συγκεκριμένη κλήση και τη γενικότερη κατάσταση του συστήματος την ώρα που γινόταν η κλήση.

**Είσοδος/'Εξοδος**

Το Qdos υποστηρίζει ένα περιβάλλον πολλαπλών εργασιών και γι' αυτό μπορεί να προσπελαστεί ένα αρχείο με περισσότερες από μία εργασίες, κάθε φορά. Το Qdos υποσύστημα των αρχείων μπορεί να αντιμετωπίσει αρχεία τα οποία έχουν ανοιχθεί σαν αποκλειστικά ή σαν μοιρασμένα (shared) αρχεία. Σ' ένα μοιρασμένο αρχείο δεν μπορείτε να γράψετε. Οι QL συσκευές επεξεργάζονται από το σειριακό I/O υποσύστημα. Το υποσύστημα αρχείων και το σειριακό I/O υποσύστημα μαζί αποτελούν το επανακατευθυνόμενο I/O σύστημα. 'Οπως δείχνει το όνομα του, κάθε έξοδος δεδομένων διαμέσου αυτού του συστήματος μπορεί να επανακατευθυνθεί σε οποιαδήποτε άλλη συσκευή που υποστηρίζεται επίσης από αυτό το I/O σύστημα.

Τα ονόματα συσκευών που απαιτούνται από το Qdos είναι τα ίδια όπως αυτά που απαιτούνται από τη SuperBASIC και αναφέρονται στο μέρος εννοιών συσκευές. Η συλλογή των στάνταρτ συσκευών που δίνονται με το QL μπορεί να επεκταθεί.

**Πολλαπλές εργασίες**

Εργασίες θα επιτρέπουν ένα μοίρασμα της CPU στη σύμφωνα με την προτεραιότητα τους και το συναγωνισμό με τις άλλες εργασίες στο σύστημα. Οι εργασίες που τρέχουν κάτω από τον έλεγχο του Qdos μπορεί να είναι σε μία από τις τρεις μορφές:

Ενεργή: Ικανή να τρέξει και να μοιραστεί τις δυνατότητες του συστήματος. Μια εργασία σε αυτή τη μορφή μπορεί να μην τρέχει συνεχώς αλλά θα πάρει ένα μερίδιο της CPU που σύμφωνα με την προτεραιότητά της.

Ανακόπτόμενη: Η εργασία είναι ικανή να τρέχει αλλά περιμένει για μια άλλη εργασία ή I/O. Μια εργασία μπορεί να ανακόπτεται αόριστα για μια συγκεκριμένη περίοδο χρόνου.

Μη ενεργή: Η εργασία δεν είναι ικανή να τρέχει, η προτεραιότητά της είναι 0 και έτσι δεν μπορεί ποτέ να πάρει ένα μερίδιο της CPU.

Το Qdos θα επανακαταγράψει το σύστημα αυτόματα σε μια κλίμακα σχετική με την κλίμακα των 50 Hz. Το σύστημα επίσης θα επανακαταγραφεί μετά από συγκεκριμένα καλέσματα του συστήματος.

Παράδειγμα

Αυτό το πρόγραμμα παράγει μια εικόνα πάνω στην οθόνη του ρολογιού πραγματικού χρόνου, που τρέχει σαν μία ανεξάρτητη εργασία.

Πρώτα τρέξτε αυτό το πρόγραμμα με μία φορμαρισμένη μικροκασέτα στο microdrive 2. Αυτό παράγει έναν τίτλο σε γλώσσα μηχανής που ονομάζεται "clock".

Μετά πληκτρολογήστε:

```
EXEC mdv2_clock
```

και μια συνεχής εμφάνιση ώρας θα εμφανιστεί στο πάνω δεξιά μέρος του παραθύρου εντολών.

```
100  c=RESPR(100)
110  FOR i=0 TO 68 STEP 2
120    READ x:POKE_W i+c,x
130  END FOR i
140  SEXEC mdv2_clock,c,100,256
1000 DATA 29439,29697,28683,20033,17402
1010 DATA 48,13944,200,20115,12040
1020 DATA 28691,20033,17402,74,-27698
1030 DATA 13944,236,20115,8279,-11314
1040 DATA 13944,208,20115,16961,16962
1050 DATA 30463,28688,20035,24794
1060 DATA 0,7,240,10,272,200
```

Η σειρά 1060 κυβερνά τη θέση και το χρώμα του παραθύρου του ρολογιού - τα μέρη της ημερομηνίας είναι με τη σειρά:

χρώμα πλαισίου / πλάτος, χαρτί / χρώμα μελάνης, πλάτος παραθύρου, ύψος, x-αρχή, y-αρχή.

που εισάγονται από τη POKE_W σαν λέξεις. Οι συντεταγμένες x-αρχή και y-αρχή πρέπει να είναι 272 και 202 στη μορφή μόνιτορ ή 240 και 216 στη μορφή TV.

Δημιουργείτε τη λέξη χαρτί και μελάνι, για παράδειγμα, σαν 256 * χαρτί + μελάνι. 'Ετσι άσπρο χαρτί, κόκκινη μελάνη είναι 256 * 7 + 2 = 1794.

## Επανάληψη

Η επανάληψη στη SuperBASIC ελέγχεται από δύο βασικές δομές προγράμματος. Κάθε δομή θα πρέπει να αναγνωρίζεται στη SuperBASIC:

**REPEAT** *identifier*
  *statements*
**END REPEAT** *identifier*

**FOR** *identifier* = *range*
  *statements*
**END FOR** *identifier*

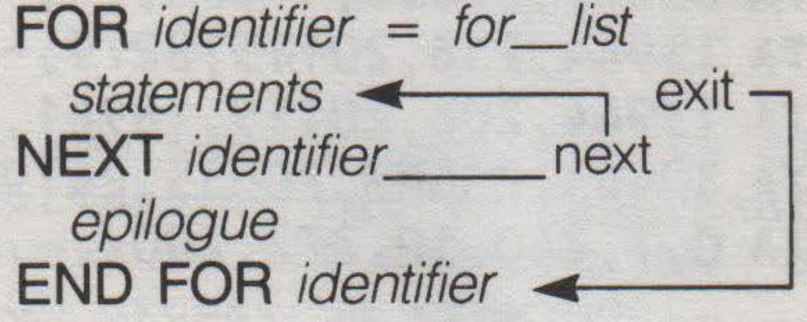
Αυτές οι δύο δομές χρησιμοποιούνται σε συνδυασμό με δύο άλλες SuperBASIC προτάσεις:

NEXT *identifier* EXIT *identifier*

Η εκτέλεση μιας NEXT εντολής είτε θα περάσει τον έλεγχο στην εντολή που ακολουθεί την κατάλληλη FOR ή REPeat, ή αν μία FOR έχει εξαντληθεί στην εντολή που ακολουθεί το NEXT.

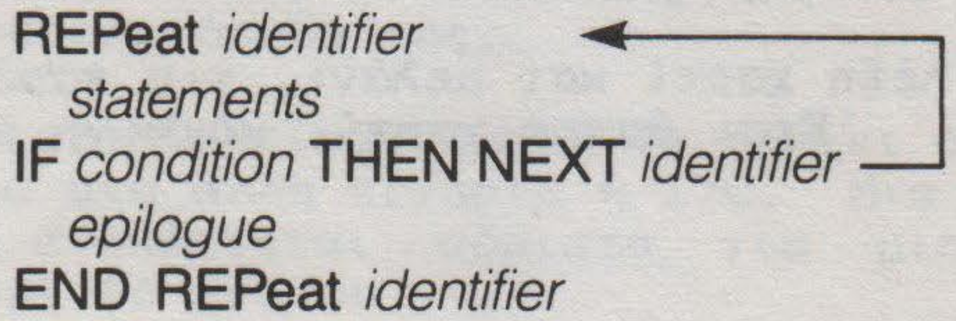
Η εκτέλεση μιας EXIT θα περάσει τον έλεγχο στην εντολή μετά το END FOR ή το END REPeat που επιλέχθηκε από την EXIT. Η EXIT μπορεί να χρησιμοποιηθεί για να βγείτε μέσα από πολλά επίπεδα από φωλιασμένες επαναλαμβανόμενες δομές. Η EXIT θα πρέπει να χρησιμοποιείται πάντα σε REPeat βρόχους για να τερματίσει το βρόχο σε κάποια συνθήκη.

'Ενας συνδυασμός των NEXT, EXIT και END επιτρέπει στους FOR και REPeat βρόχους να τους προστεθεί ένας επίλογος βρόχου. 'Ενας επίλογος βρόχου είναι μία σειρά από SuperBASIC εντολές οι οποίες εκτελούνται με κάποια ειδική συνθήκη που προκύπτει μέσα στο βρόχο.




Ο επίλογος βρόχου εκτελείται μόνο αν ο βρόχος FOR τερματίζει κανονικά. Αν ο βρόχος τερματίζει διαμέσου μιας εντολής EXIT, τότε η εκτέλεση θα συνεχίσει στο END FOR και ο επίλογος βρόχου δε θα εκτελεστεί.

Είναι πιθανό να έχετε μια παρόμοια δομή μέσα σε ένα REPeat βρόχο:




Αυτή η είσοδος μέσα στον επίλογο βρόχου ελέγχεται από την IF εντολή. Ο επίλογος βρόχου θα εκτελεστεί ή δε θα εκτελεστεί ανάλογα από τη συνθήκη στην εντολή IF. Μια εντολή SELect μπορεί επίσης να χρησιμοποιηθεί για να ελέγξει την είσοδο στον επίλογο βρόχου.

## Σχισμή για ROM cartridge

Επιτρέπει να χρησιμοποιηθεί το software μέσα στο σύστημα του QL διαμέσου του Sinclair QL ROM cartridge. Το ROM cartridge μπορεί να περιέχει software για να αλλάξει απευθείας τη συμπεριφορά του SuperBASIC συστήματος. Το cartridge μπορεί να περιέχει:

i. Software για να αντικαταστήσει το SuperBASIC σύστημα. Για παράδειγμα:

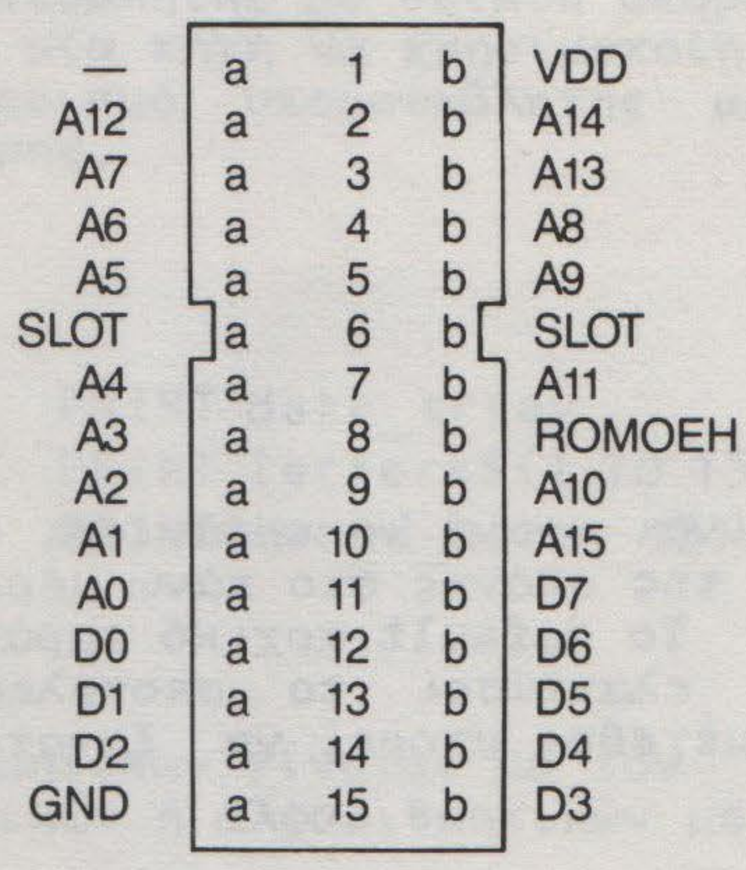
   Assemblers
   Compilers
   debuggers
   Εφαρμογές software
   κ.λ.π.

ii. Software για να επεκτείνει το SuperBASIC σύστημα. Για παράδειγμα:

   Ειδικές διαδικασίες
   κ.τ.λ.

Δεν είναι δυνατό να χρησιμοποιήσετε ZX ROM cartridges στο QL.

**pin out**




Η β' πλευρά είναι η πάνω πλευρά του συνδέσμου, ενώ η πλευρά α' είναι η χαμηλότερη.

| Signal | Function |
|---|---|
| AO..A15 | Address lines |
| DO..D8 | Data lines |
| ROMOEH | ROM Output Enable |
| VDD | 5V |
| GND | Ground |

**Προειδοποίηση**

Ποτέ μη βάζετε ή βγάζετε ένα ROM cartridge όταν το QL είναι στη πρίζα.

## Μόνιτορ

**Μορφή 512**

Η οθόνη είναι 512 pixels σε μήκος και 256 pixels σε ύψος. Μόνο τα κύρια χρώματα:

μαύρο
κόκκινο
πράσινο
άσπρο

μπορούν να εμφανιστούν σε αυτή τη μορφή.

Η μορφή χαμηλής διακριτικότητας επίσης έχει ένα hardware αναβόσβημα.

**Μορφή 256**

Η οθόνη είναι 256 στοιχεία σε μήκος και 256 σε ύψος. Ολόκληρη η σειρά των βασικών χρωμάτων προσφέρεται σε αυτή τη μορφή:

μπλε
κόκκινο
μωβ
πράσινο
γαλάζιο
κίτρινο
άσπρο

**Προειδοποίηση**

Μια τηλεόραση δεν είναι ικανή να εμφανίσει ολόκληρη την οθόνη του QL. Κομμάτια της οθόνης στο πάνω μέρος και στις πλευρές δε θα αναπαραχθούν. Το default αρχικό παράθυρο θα το λάβει υπόψη του και θα ελαττώσει το αποτελεσματικό μέγεθος της εικόνας. Το πληρε μέγεθος μπορεί να ξανατοποθετηθεί με την εντολή WINDOW.

| Διαταγή | Λειτουργία |
|---|---|
| MODE | βάζει τη μορφή της οθόνης |

## Τεμαχισμός

Είναι δυνατό κάτω από συγκεκριμένες συνθήκες να αναφερθείτε σε περισσότερα από ένα στοιχεία σε μία μεταβλητή με δείκτες δηλαδή να κομματιάσετε τη μεταβλητή. Το κομμάτιασμα της μεταβλητής μπορεί να θεωρηθεί σαν ένας καθορισμός μιας υπομεταβλητής ή μια σειρά από υπομεταβλητές στη SuperBASIC. Κάθε κομμάτι μπορεί να καθορίσει μια συνεχόμενη ακολουθία από

στοιχεία που ανήκουν σε μια συγκεκριμένη διάσταση της αρχικής μεταβλητής. Ο όρος μεταβλητή με δείκτη στο περιεχόμενο μπορεί να περιλαμβάνει μια αριθμητική μεταβλητή με δείκτες μια αλφαριθμητική μεταβλητή με δείκτες ή μία απλή αλφαριθμητική μεταβλητή.

Δεν είναι απαραίτητο να καθορίσετε ένα δείκτη για τον πλήρη αριθμό των διαστάσεων μιας μεταβλητής με δείκτες. Αν παραληφθεί μια διάσταση τότε προσθέτονται κομμάτια που θα επιλέξουν την πλήρη κλίμακα των στοιχείων για εκείνη τη συγκεκριμένη διάσταση, π.χ. το κομμάτι (0 TO). Η SuperBASIC μπορεί να προσθέσει κομμάτια στο τέλος του καταλόγου των δεικτών των μεταβλητών.

*index*:= | *numeric_exp* {single element}
| *numeric_exp* **TO** *numeric_exp* {range of elements}
| *numeric_exp* **TO** {range to end}
| **TO** *numeric_expression* {range from beginning}

*array_reference*:= | *variable*
| *variable* ( [*index* *[,*index*]*] )

'Ενα κομμάτιασμα μεταβλητής με δείκτη μπορεί μα χρησιμοποιηθεί για να καθορίσει μία πηγή να χρησιμοποιηθεί για να καθορίσει μία πηγή ή ένα προορισμό υπομεταβλητής με δείκτες για μία εντολή αντικατάστασης.

i. PRINT data_array
ii. PRINT letters$(1 TO 15)
iii. PRINT two_d_array (3)(2 TO 4)

Τεμαχισμός αλφαριθμητικών γίνεται με τον ίδιο τρόπο όπως ο τεμαχισμός αριθμητικών ή αλφαριθμητικών με δείκτες.

'Ετσι

| | |
|---|---|
| a$(n) | Διαλέγει το n-οστό χαρακτήρα. |
| a$(n TO m) | Διαλέγει όλους τους χαρακτήρες από τον n-οστό ως τον m-οστό συμπεριλαμβανομένων. |
| a$(n TO) | Διαλέγει από τον n-οστό χαρακτήρα μέχρι το τέλος συμπεριλαμβανομένων. |
| a$(1 TO m) | Διαλέγει από την αρχή μέχρι το m-οστό χαρακτήρα συμπεριλαμβανομένων. |
| a$ | Διαλέγει ολόκληρη την αλφαριθμητική. |

Μερικές διάλεκτοι της BASIC έχουν συναρτήσεις LEFT$, MID$, RIGHT$ που δεν είναι απαραίτητοι στη SuperBASIC. Η ισοδυναμία τους ορίζεται ως εξής:

| SuperBASIC | Other BASIC |
|---|---|
| a$(n) | MID$(a$,n,1) |
| a$(n TO m) | MID$ (a$,n,m+1-n) |
| a$(1 TO n) | LEFT$(a$,n) |
| a$(n TO) | RIGHT$ (a$,LEN(a$)+1-n) |

**Προειδοποίηση**

Η εκχώρηση δεδομένων σε μια κομματιασμένη αλφαριθμητική μεταβλητή ή αλφαριθμητική μεταβλητή μπορεί να μην έχει το επιθυμητό αποτέλεσμα. Οι εκχωρήσεις που έγιναν με αυτόν τον τρόπο δε θα αλλάξουν το μήκος της αλφαριθμητικής. Το μήκος της αλφαριθμητικής μεταβλητής με δείκτη ή της αλφαριθμητικής μεταβλητής αλλάζει μόνο όταν μια εκχώρηση γίνεται σε όλη την αλφαριθμητική.

## Ξεκίνημα

Αμέσως αφού ανοίξετε (ή επανατοποθετήσετε) το QL θα εκτελέσει ένα έλεγχο της RAM που θα δώσει νόθο εικόνα πάνω στην οθόνη. Αν ο έλεγχος της RAM περάσει τότε η οθόνη θα καθαριστεί και θα εμφανιστεί η οθόνη.


Μετά το ξεκίνημα το QL εμφανίζει το μήνυμα των συγγραφικών δικαιωμάτων και ρωτά αν το QL χρησιμοποιείται με τηλεόραση ή οθόνη. Το QL θα τοποθετήσει διαφορετικές μορφές αρχικής οθόνης και μεγέθη παραθύρων ανάλογα με την απάντηση.

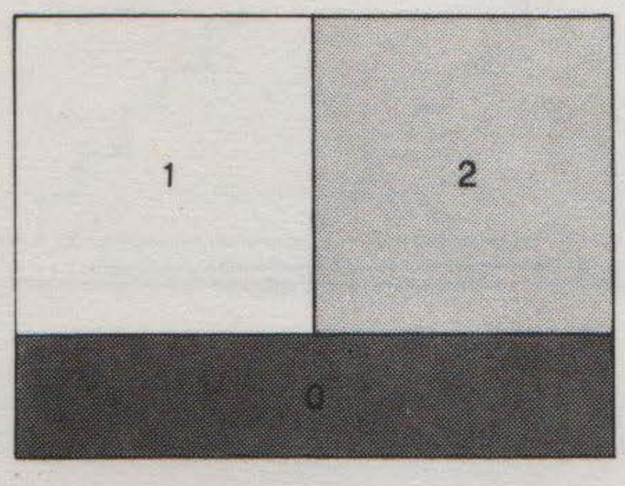
Πατήστε το F1 αν χρησιμοποιείτε μία οθόνη και το F2 αν χρησιμοποιείτε μία τηλεόραση.

Το QL έχει την ικανότητα να ξεκινά το ίδιο από προγράμματα που βρίσκονται είτε στο ROM cartridge slot είτε στο Microdrive 2.

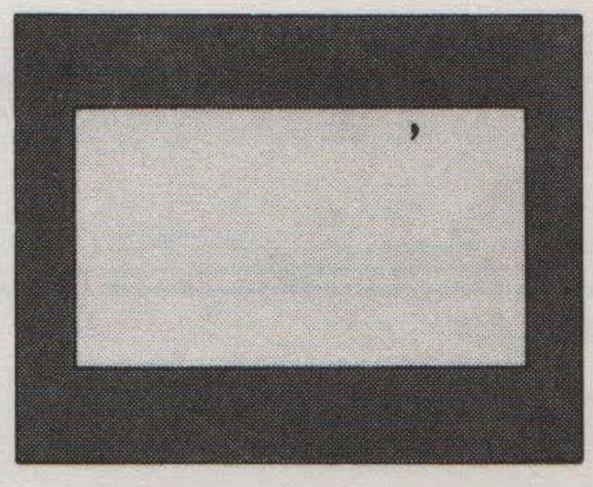
Αν το ROM cartridge περιέχει ένα πρόγραμμα που αρχίζει μόνο του τότε το ξεκίνημα θα συνεχίσει κάτω από τον έλεγχο του προγράμματος στο ROM cartridge. Αν δε βρεθεί κάτι κατάλληλο τότε το QL θα εξετάσει το Microdrive 2 για ένα cartridge. Αν βρεθεί ένα και αν περιέχει ένα αρχείο που να ονομάζεται BOOT τότε φορτώνεται και τρέχει.

## Η default οθόνη

Το QL έχει τρία default κανάλια που συνδέονται σε τρία default παράθυρα.




Μόνιτορ



Τηλεόραση

Το κανάλι 0 χρησιμοποιείται για την εμφάνιση διαταγών και μηνυμάτων λάθους, το κανάλι 1 για τα εξαγόμενα των προγραμμάτων και των γραφικών και το κανάλι 2 για τις λίστες των προγραμμάτων. Το default κανάλι μπορεί να μετατραπεί χρησιμοποιώντας τον προαιρετικό καθοριστή καναλιού στη σχετική διαταγή.

## Προειδοποίηση

Είναι σημαντικό να μην ανοίξετε το QL με μία μικροκασέτα στη θέση της. Αν απαιτείται ξεκίνημα του QL τότε η μικροκασέτα θα πρέπει να εισαχθεί πριν ανάψετε τον υπολογιστή και να πατήσετε είτε το F1 ή το F2.

## Ήχος

Ο ήχος στο QL παράγεται από τον IPC (8049) δεύτερο επεξεργαστή και ελέγχεται προσδιορίζοντας:

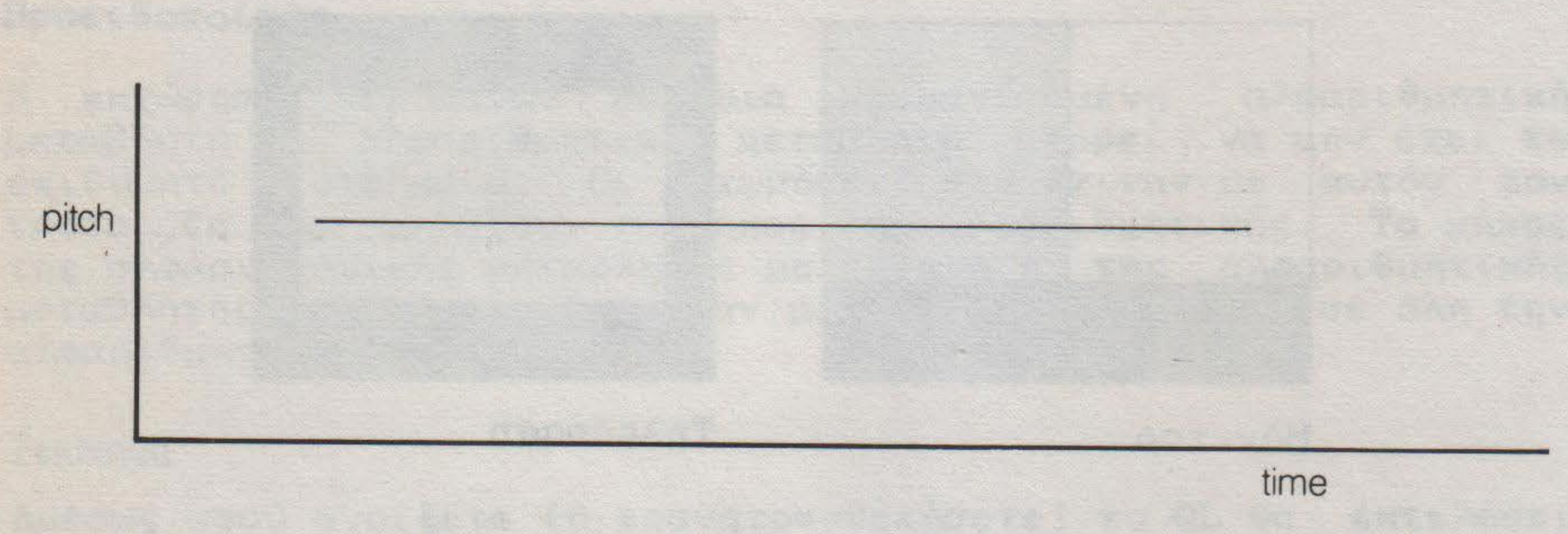
- ως δύο βαθμούς
- η κλίμακα μέσα στην οποία θα πρέπει να κινείται ο ήχος ανάμεσα στους βαθμούς, te ramp
- πως πρόκειται να συμπεριφερθεί ο ήχος αφότου έχει φτάσει έναν από τους συγκεκριμένους βαθμούς, the wrap
- αν θα πρέπει να κτιστεί κάποια randomness μέσα στον ήχο, π.χ. παρεκτροπές από το ramp
- αν θα πρέπει να κτιστεί κάποια fuzziness μέσα στον ήχο, π.χ. παρεκτροπές σε κάθε κύκλο του ήχου.

Η fuzziness τείνει να έχει σαν αποτέλεσμα σε βομβώδεις ήχους ενώ η randomness, ανάλογα με τις άλλες παραμέτρους, θα έχει σαν αποτέλεσμα "μελωδικούς ήχους ή θόρυβο.

Η πολυπλοκότητα του ήχου μπορεί να κτιστεί στάδιο με στάδιο σταδιακά κτίζοντας περισσότερο πολύπλοκους ήχους. Αυτός, είναι ο καλύτερος τρόπος να αντιμετωπίσετε τον ήχο στο QL.

Καθορίστε μια διάρκεια και ένα μοναδικό pitch. Το καθορισμένο pitch θα ηχεί για τον καθορισμένο χρόνο.

**ΕΠΙΠΕΔΟ 1**

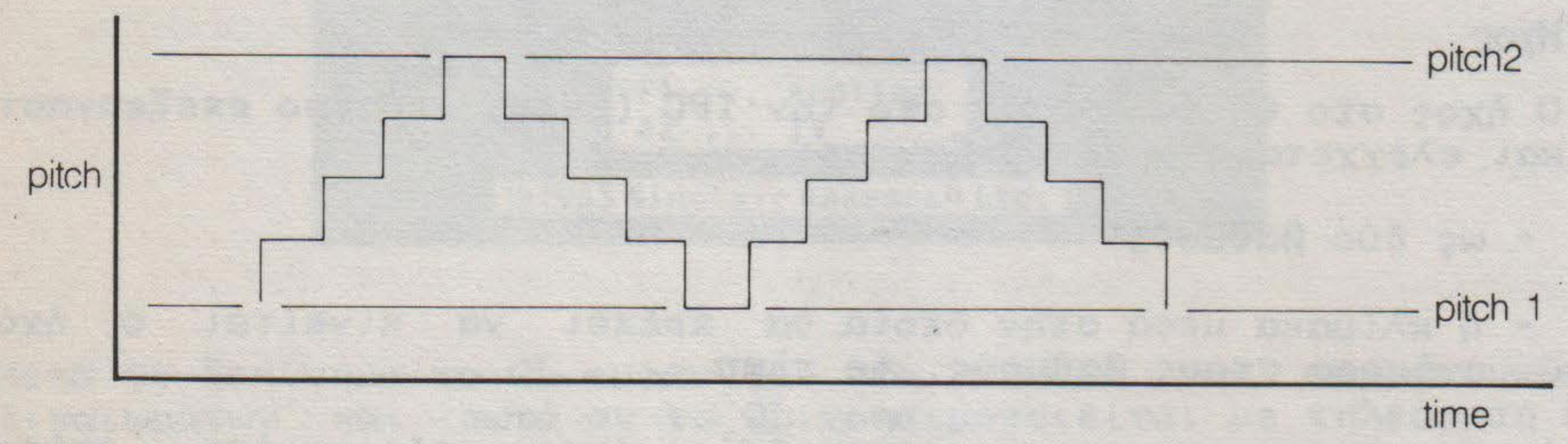



Αυτή είναι η απλούστερη εντολή ήχου, εκτός από την εντολή για να σταματήσει τον ήχο στο QL.

**ΕΠΙΠΕΔΟ 2**

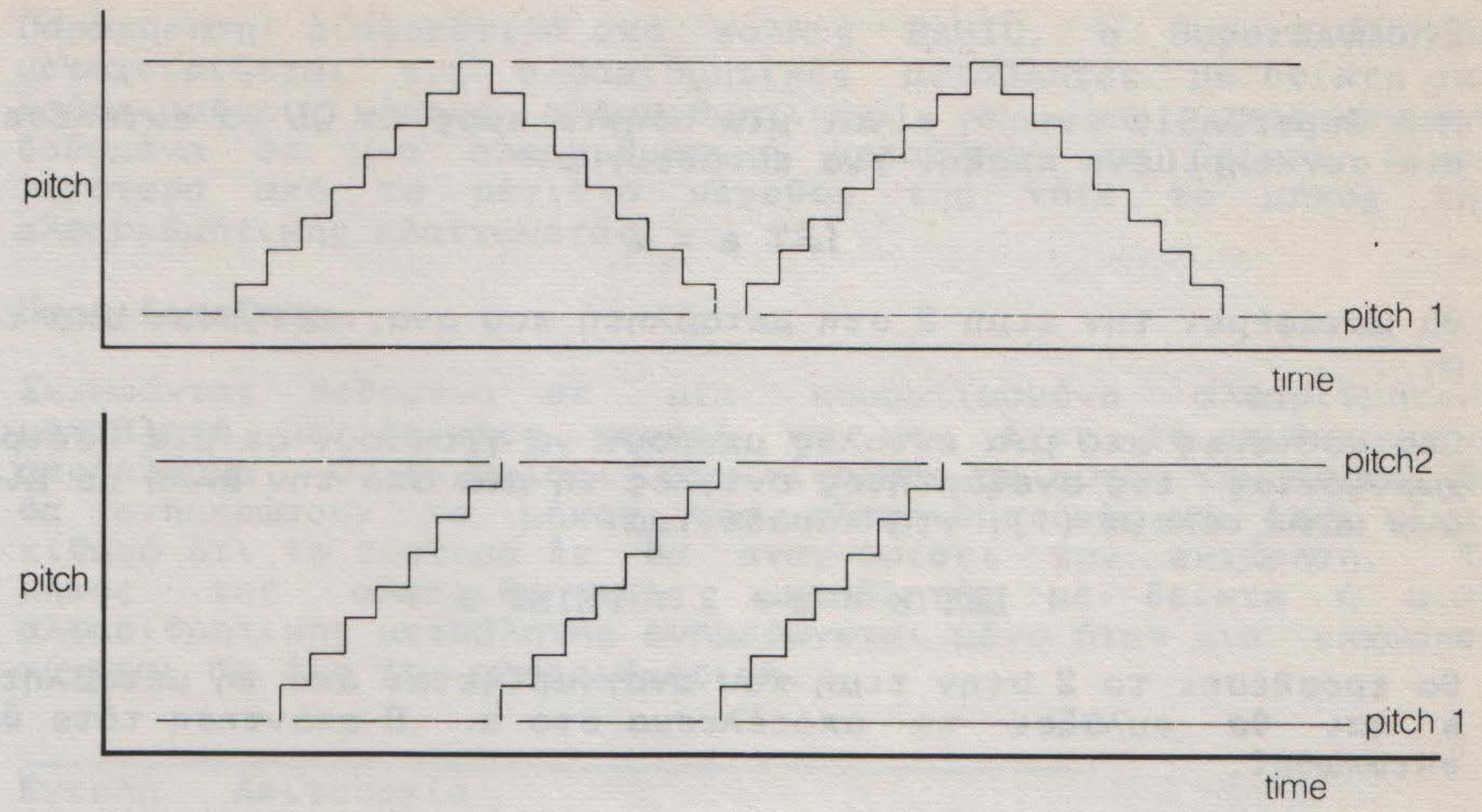
'Ενα δεύτερο pitch και ένα gradient μπορούν να προστεθούν στην εντολή. Ο ήχος τότε θα κινηθεί ανάμεσα στα δύο pitches στην κλίμακα που καθορίζεται από το gradient.

Οι ήχοι που παράγονται σε αυτό το επίπεδο μπορούν να ποικίλλουν ανάμεσα: σε ημιμουσικά beeps, growls, zaps και moans. Είναι καλύτερο να πειραματιστείτε:




**ΕΠΙΠΕΔΟ 3**

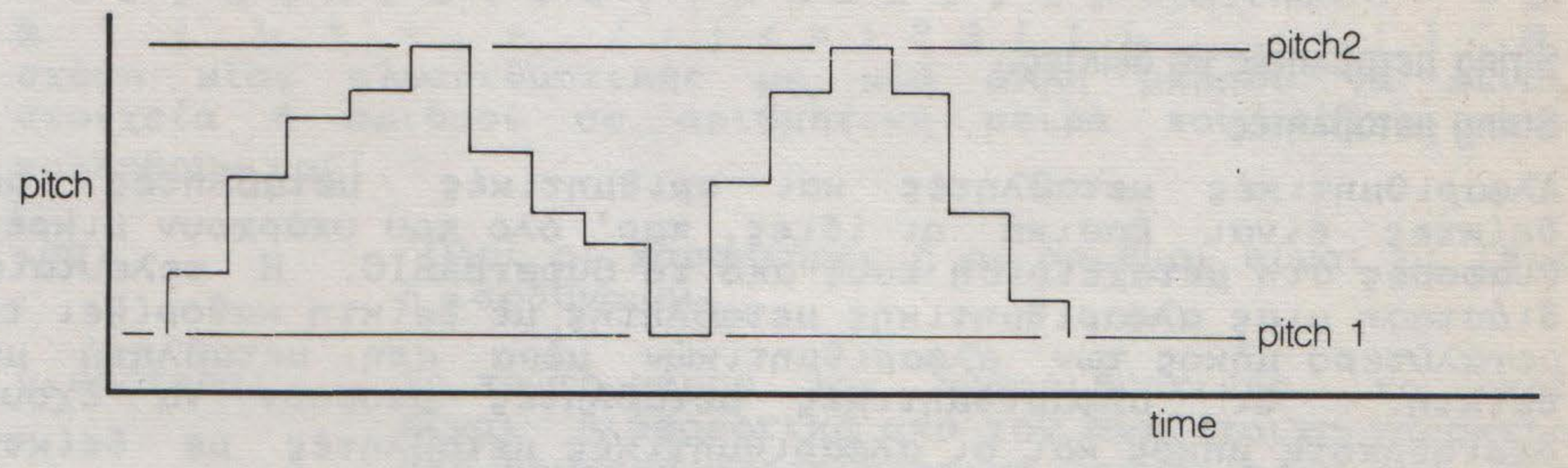
Μπορεί να προστεθεί μία παράμετρος η οποία ελέγχει πως συμπεριφέρεται ο ήχος όταν γίνεται ίσος με ένα από τα καθορισμένα pitches. Ο ήχος μπορεί να γίνει να "bounce" ή να "wrap". Ο αριθμός των wraps μπορεί να καθοριστεί, μαζί με το wrap πάντα. Είναι ακόμα πιο σπουδαίο να πειραματιστείτε:




**ΕΠΙΠΕΔΟ 4**

Μπορεί να προσθεθεί η randomness στον ήχο. Αυτή είναι μία παρεκτροπή από το συγκεκριμένο βήμα ή gradient.

Ανάλογα με το μέγεθος της randomness που προστίθεται σχετικά με τα pitches και το gradient θα παράγει μια πολύ μεγάλη και απρόσμενη σειρά από ήχους.




**ΕΠΙΠΕΔΟ 5**

Μπορεί να προστεθεί μεγαλύτερη ποικιλία προσδιορίζοντας τη "fuzziness". Η fuzziness προσθέτει ένα random συντελεστή στο pitch συνεχόμενα. Τείνει να κάνει τον ήχο να buzz.

Συνδυάζοντας όλα τα παραπάνω αποτελέσματα μπορεί να κάνει μια μεγάλη κλίμακα από ήχους, πολλούς από αυτούς απρόσμενους. Ο ήχος του QL εξερευνείται καλύτερα μέσα από τον πειραματισμό. Με το να προσδιορίσετε ένα διάλειμμα χρόνου μηδέν, ο ήχος μπορεί να γίνει να επαναλαμβάνετε συνέχεια και έτσι και έτσι μια ακολουθία από BEEP εντολές μπορεί να χρησιμοποιηθεί ωσότου ο παραγμένος ήχος να είναι αυτός που απαιτείται. Μια προειδοποίηση, οι ελαφρές αλλαγές στην τιμή μιας μοναδικής παραμέτρου μπορούν να έχουν τρομερά αποτελέσματα στον παραγμένο ήχο, γι' αυτό μην πανικοβάλλεστε!

## Εντολή

Μια SuperBASIC εντολή είναι μία οδηγία προς το QL να εκτελέσει μια συγκεκριμένη πράξη, για παράδειγμα:

```
LET a = 2
```

θα μεταφέρει την τιμή 2 στη μεταβλητή που αναγνωρίζεται από το α.

Περισσότερες από μια εντολές μπορούν να γραφτούν σε μια σειρά χωρίζοντας τις ανεξάρτητες εντολές τη μία από την άλλη με μία άνω κάτω τελεία (:), για παράδειγμα:

```
LET a = a + 2 : PRINT a
```

θα προσθέσει το 2 στην τιμή που αναγνωρίζεται από τη μεταβλητή a και θα φυλάξει το αποτέλεσμα στο a. Η απάντηση τότε θα εκτυπωθεί.

Αν σε μια σειρά δεν προηγείται ο αριθμός σειράς τότε είναι μια άμεση εντολή και η SuperBASIC εκτελεί την εντολή αμέσως. Αν στην εντολή προηγείται ένας αριθμός σειράς τότε η εντολή γίνεται μέρος ενός SuperBASIC προγράμματος και προστίθεται σε μία SuperBASIC περιοχή προγράμματος για εκτέλεση αργότερα.

Μερικές SuperBASIC εντολές μπορούν να έχουν ένα αποτέλεσμα σε άλλες προτάσεις πάνω στο υπόλοιπό της λογικής σειράς στην οποία εμφανίζονται, π.χ. IF, FOR, REPeat, REM κ.τ.λ. Είναι χωρίς νόημα να χρησιμοποιείτε συγκεκριμένες SuperBASIC εντολές σαν άμεσες εντολές.

## String μεταβλητές με δείκτες.

## String μεταβλητές

Αλφαριθμητικές μεταβλητές και αριθμητικές μεταβλητές με δείκτες είναι βασικά οι ίδιες, παρ' όλο που υπάρχουν μικρές διαφορές στη μεταχείριση τους από τη SuperBASIC. Η τελευταία διάσταση μιας αλφαριθμητικής μεταβλητής με δείκτη καθορίζει το μεγαλύτερο μήκος των αλφαριθμητικών μέσα στη μεταβλητή με δείκτη. Οι αλφαριθμητικές μεταβλητές μπορούν να έχουν οποιοδήποτε μήκος και οι αλφαριθμητικές μεταβλητές με δείκτη και οι αλφαριθμητικές μεταβλητές μπορούν να τεμαχιστούν.

Τα μήκη των αλφαριθμητικών και στις δύο μεριές μιας αλφαριθμητικής εκχώρησης πρέπει να είναι ίσα. Αν τα μεγέθη δεν είναι τα ίδια τότε είτε η δεξιά αλφαριθμητική κόβεται για να χωρέσει ή το μήκος της αριστεράς ελαττώνεται για να ταιριάξει. Αν γίνει μια εκχώρηση σε μία κομματιασμένη αλφαριθμητική τότε αν είναι απαραίτητο το άνοιγμα καθορισμένο από το κομμάτι θα γεμίσει με κενά.

Δεν είναι απαραίτητο να προσδιορίσετε την τελική διάσταση της αλφαριθμητικής μεταβλητής με δείκτη. Αν δεν προσδιορίσετε τη διάσταση επιλέγει όλη την αλφαριθμητική ενώ προσδιορίζοντας ένα μοναδικό στοιχείο θα διαλέξει ένα μοναδικό χαρακτήρα και προσδιορίζοντας ένα κομμάτι θα καθορίσει μία υπο-αλφαριθμητική.

Παρατήρηση: Διαφορετικά από πολλές BASIC, η SuperBASIC δε μεταχειρίζεται τις αλφαριθμητικές μεταβλητές με δείκτη σαν καθορισμένου μήκους αλφαριθμητικές. Αν τα αποθηκευμένα δεδομένα σε μια αλφαριθμητική μεταβλητή με δείκτη είναι λιγότερα από το μέγιστο μέγεθος της τότε το μήκος της αλφαριθμητικής ελαττώνεται.

**Προειδοποίηση**

Εκχωρόντας δεδομένα σε μία κομματιασμένη αλφαριθμητική μεταβλητή με δείκτες μπορεί να μην έχει το επιθυμούμενο αποτέλεσμα. Οι εκχωρίσεις που γίνονται με αυτόν τον τρόπο δε θα ενημερώσουν το μήκος της αλφαριθμητικής και έτσι είναι πιθανό ότι το σύστημα δε θα αναγνωρίσει την εκχώρηση. Το μήκος της αλφαριθμητικής μεταβλητής με δείκτη ή μιας αλφαριθμητικής μεταβλητής ενημερώνεται μόνο όταν μια εκχώρηση γίνεται σε όλη την αλφαριθμητική.

| Εντολή | Λειτουργία |
|---|---|
| FILL$ | παράγει μία αλφαριθμητική |
| LEN$ | βρίσκει το μήκος μιας αλφαριθμητικής |

**Σύγκριση αλφαριθμητικών**

**Τάξη**

.(δεκαδικό σημείο / τελεία) ψηφία ή αριθμοί σε αριθμητική σειρά A a B b C c D d E e F f G g H h I i J j K k L l M m N n O o P p Q q R r S t U u V v W w X x Y y Z z διάστημα ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ ¦ ] ^ _ / { ¦ } ~ @ H σχέση μιας αλφαριθμητικής με μία άλλη μπορεί να είναι: στοιχεία ή αριθμοί σε αριθμητική σειρά που πρόκειται να κολληθούν μαζί.

ίσα: 'Ολοι οι χαρακτήρες ή οι αριθμοί είναι οι ίδιοι ή ισοδύναμοι.

μικρότερα: Το πρώτο μέρος της αλφαριθμητικής, το οποίο είναι διαφορετικό από τον αντίστοιχο χαρακτήρα στη δεύτερη αλφαριθμητική, είναι πριν από αυτό στη καθορισμένη σειρά.

μεγαλύτερα: Το πρώτο μέρος της αλφαριθμητικής το οποίο είναι διαφορετικό από τον αντίστοιχο χαρακτήρα στη δεύτερη αλφαριθμητική, είναι μετά από αυτό στην καθορισμένη σειρά.

**Μορφή σύγκρισης**

μορφή 0 εξαρτάται από το είδος των γραμμάτων - σύγκριση χαρακτήρα με χαρακτήρα

μορφή 1 ανεξάρτητη από το είδος των γραμμάτων - χαρακτήρας με χαρακτήρα.

μορφή 2 εξαρτάται από το είδος των γραμμάτων - οι αριθμοί ταξινομούνται σε αριθμητική σειρά

μορφή 3 ανεξάρτητη από το είδος των γραμμάτων - οι αριθμοί ταξινομούνται με αριθμητική σειρά.

μορφή 0 δε χρησιμοποιείται συνήθως στη SuperBASIC.

**Χρησιμοποίηση**

- Η μορφή 1 Σύγκριση αρχείων και μεταβλητών
- Η μορφή 2 SuperBASIC <, <=, =, >=, > INSTR και <>
- Η μορφή 3 SuperBASIC == ισοδυναμία.

## Ορισμοί σύνταξης

Η σύνταξη της SuperBASIC καθορίζεται χρησιμοποιώντας μία μορφή σημείωσης μιας μη αυστηρής "μετάγλωσσης". Χρησιμοποιούνται τέσσερεις μορφές δόμησης:

```
        || Διαλέξτε ένα από
        [] Τα περικλειόμενα στοιχεία ή στοιχείο μπορούν
           να είναι προαιρετικά
        ** Τα περικλειόμενα στοιχεία επαναλαμβάνονται
        .. Εύρος
        {} Παρατήρηση
π.χ.    |A|B|      το A ή το B
        [A]        το A είναι προαιρετικό
        *A*        το A επαναλαμβάνεται
        A..Z       A,B,C κ.τ.λ.
        {αυτή είναι μία παρατήρηση}
```

'Ενας SuperBASIC αναγνωριστής είναι:

Μια ακολουθία από γράμματα, ψηφία, υπογραμμίσεις, αρχίζοντας με ένα γράμμα και τελειώνοντας με ένα % ή $ προαιρετικά.

*letter*:= | A..Z
| a..z

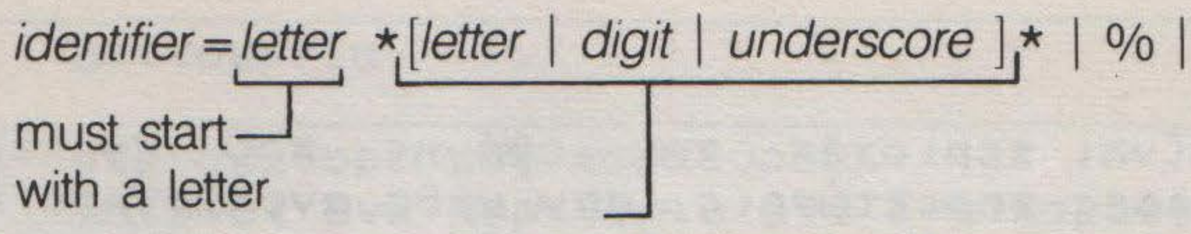
{a letter is one of: ABCDEFGHIJKLMNOPQRSTUVWXYZ}
or abcdefghijklmnopqrstuvwxyz

*digit*:= | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

{a digit is 0 or 1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9}

*underscore*:= _

{an underscore is _}




Επαναλαμβάνεται

κάτι προαιρετικά

## Γραφικά χελώνας

Η SuperBASIC έχει ένα σύνολο από εντολές για γραφικά χελώνας:

| Διαταγή | Λειτουργία |
|---|---|
| PENUP | σταματά τη σχεδίαση |
| PENDOWN | αρχίζει τη σχεδίαση |
| MOVE | κινεί τη χελώνα |
| TURN | γυρίζει τη χελώνα |
| TURNTO | γυρίζει σε μία συγκεκριμένη κατεύθυνση |

Το σύνολο των εντολών είναι το λιγότερο και κανονικά μπορούν να χρησιμοποιηθούν μέσα σε μία άλλη διαδικασία για να επεκταθούν οι εντολές. Για παράδειγμα:

```
100 DEFine PROCedure forward(distance)
110   MOVE distance
120 END DEFine
130 DEFine PROCedure backwards(distance)
140   MOVE -distance
150 END DEFine
160 DEFine PROCedure left(angle)
170   TURN angle
180 END DEFine
190 DEFine PROCedure right(angle)
200   TURN -angle
210 END DEFine
```

Αυτές θα καθορίσουν μερικές από τις πιο διάσημες εντολές γραφικών χελώνας.

Αρχικά, η πένα της χελώνας είναι σηκωμένη και η χελώνα δείχνει στις 0 μοίρες, που αυτό είναι προς τη δεξιά μεριά του παραθύρου.

Η εντολή FILL θα δουλέψει επίσης με σχήματα που είναι σχεδιασμένα με γραφικά χελώνας. Επίσης τα συνηθισμένα και τα γραφικά χελώνας μπορούν να αναμιχθούν, παρ' όλο που η διεύθυνση της χελώνας δεν καθορίζεται από τις συνηθισμένες εντολές των γραφικών.

## Παράθυρα

Τα παράθυρα είναι περιοχές της οθόνης που συμπεριφέρονται, στις περισσότερες περιπτώσεις, σαν κάθε ανεξάρτητο παράθυρο να ήταν μια οθόνη με τα δικά της δικαιώματα, π.χ. το παράθυρο θα κάνει scroll όταν έχει γεμίσει με κείμενο, μπορεί να καθαριστεί με την εντολή CLS, κ.τ.λ.

Τα παράθυρα μπορούν να καθοριστούν και να συνδεθούν με το κανάλι όταν αυτό είναι ανοικτό. Το τρέχον σχήμα του παραθύρου μπορεί να αλλαχθεί με την εντολή WINDOW καθώς και το περιθώριο με την εντολή BORDER. Τα εξαγόμενα μπορεί να κατευθυνθούν σε ένα παράθυρο με το να εκτυπώσετε στο σχετικό κανάλι. Τα εισαγόμενα μπορούν να κανονιστούν ότι έχουν έρθει από ένα συγκεκριμένο παράθυρο με το να εισάγετε από το σχετικό κανάλι. Αν είναι έτοιμα για εισαγωγή περισσότερα από ένα κανάλια τότε η εισαγωγή μπορεί να αλλάξει ανάμεσα στα έτοιμα κανάλια πατώντας

CTRL C

Ο δρομέας θα αναβοσβήνει στο επιλεγμένο παράθυρο.

Τα παράθυρα μπορούν να χρησιμοποιηθούν για γραφικά και μη γραφικά εξαγόμενα την ίδια στιγμή. Τα μη γραφικά εξαγόμενα είναι σχετικά με την τρέχουσα θέση του δρομέα που μπορεί να τοποθετηθεί οπουδήποτε μέσα στο συγκεκριμένο παράθυρο με την εντολή CURSOR και σε οποιαδήποτε σειρά - στήλη με την εντολή AT. Τα εξαγόμενα των γραφικών είναι σχετικά με ένα δρομέα γραφικών που μπορεί να τοποθετηθεί και να χειριστεί με τις γραφικά διαδικασίες.

## Μέρη

Ορισμένες εντολές (CLS, PAN κ.τ.λ.) θα δεχθούν μια προαιρετική παράμετρο για να καθορίσουν μέρος από το τρέχον παράθυρο για τη λειτουργία τους. Αυτή η παράμετρος καθορίζεται όπως παρακάτω:

| μέρος | περιγραφή |
|---|---|
| 0 | όλη η οθόνη |
| 1 | πάνω και χωρίς τη σειρά του δρομέα |
| 2 | το κάτω της οθόνης μαζί με τη σειρά του δρομέα |
| 3 | όλη η σειρά του δρομέα |
| 4 | η σειρά δεξιά της και μαζί με το δρομέα |

| Εντολή | Λειτουργία |
|---|---|
| WINDOW | επανακαθορισμός παραθύρου |
| BORDER | παίρνει ένα περιθώριο από ένα παράθυρο |
| PAPER | καθορίζει το χρώμα του χαρτιού για ένα παράθυρο |
| INK | καθορίζει το χρώμα μελάνης για ένα παράθυρο |
| STRIP | ορίζει το χρώμα της ταινίας για ένα παράθυρο |
| PAN | ρολάρει πλαγίως τα περιεχόμενα ενός παραθύρου |
| SCROLL | ρολάρει τα περιεχόμενα ενός παραθύρου |
| AT | τοποθετεί τη θέση εκτύπωσης |
| CLEAR | καθαρίζει ένα παράθυρο |
| CSIZE | βάζει το μέγεθος ενός χαρακτήρα |
| FLASH | αναβοσβήνει ένα χαρακτήρα |
| RECOL | ξαναχρωματίζει ένα παράθυρο |

# QL

## Δεσμευμένες Λέξεις (Keywords)

## ΔΕΣΜΕΥΜΕΝΕΣ ΛΕΞΕΙΣ (KEYWORDS)

Ο Οδηγός Αναφοράς των Δεσμευμένων Λέξεων καταγράφει όλες τις δεσμευμένες λέξεις της SuperBASIC με αλφαβητική σειρά. Μία σύντομη εξήγηση της λειτουργίας των δεσμευμένων λέξεων δίνεται που ακολουθείται από ένα χαλαρό ορισμό της σύνταξης και παραδειγμάτων χρήσης. Μία εξήγηση του ορισμού σύνταξης δίνεται στον Οδηγό Αναφοράς των Εννοιών κάτω από την αναφορά της έννοιας Σύνταξη.

Με κάθε δεσμευμένη λέξη δηλώνεται σε ποια αν υπάρχει, ομάδα λειτουργιών χρησιμοποιείται, π.χ. το **DRAW** είναι μία γραφική λειτουργία και μπορούν να αποκτηθούν παραπέρα πληροφορίες από το τμήμα γραφικά του Οδηγού Αναφοράς Εννοιών. Μερικές φορές είναι απαραίτητο να ασχοληθείτε με περισσότερες από μια δεσμευμένη λέξη ταυτόχρονα π.χ. τα **IF, ELSE, THEN, END, IF,** καταγράφονται όλα κάτω από το **IF.**

'Ενας κατάλογος δίνεται που επιχειρεί να καλύψει όλους τους πιθανούς τρόπους με τους οποίους θα μπορούσε να περιγράψετε μια δεσμευμένη λέξη της SuperBASIC. Για παράδειγμα, η εντολή καθαρίσματος της οθόνης, το **CLS,** καταγράφεται επίσης κάτω από το καθάρισε οθόνη και οθόνη καθάρισε.

# ABS (Μαθηματικές συναρτήσεις)

Η ABS δίνει την απόλυτη τιμή μιας παραμέτρου. θα δώσει την παράμετρο, αυτή είναι θετική και θα δώσει μηδέν μείον την παράμετρο αν αυτή είναι αρνητική.

syntax: ABS(*numeric_expression*)

example:
i. `PRINT ABS(0.5)`
ii. `PRINT ABS(a-b)`

# ACOS, ASIN ACOT, ATAN (Μαθηματικές συναρτήσεις)

Οι **ACOS** και **ASIN** θα υπολογίσουν το τόξο συνημιτόνου και το τόξο ημιτόνου αντίστοιχα. Η **ACOT** υπολογίζει το τόξο συνεφαπτομένης και η **ATAN** το τόξο εφαπτομένης. Δεν υπάρχει όριο στο μέγεθος της παραμέτρου.

syntax: *angle*:= *numeric_expression* [in radians]

ACOS(*angle*) ASIN(*angle*)
ACOT(*angle*) ATAN(*angle*)

example:
i. `PRINT ATAN(angle)`
ii. `PRINT ASIN(1)`
iii. `PRINT ACOT(3.6574)`
iv. `PRINT ATAN(a-b)`

# ADATE (Ρολόι)

Η **ADATE** επιτρέπει την αλλαγή της ένδειξης του ρολογιού.

---

syntax: *seconds:= numeric__expression*
ADATE *seconds*

example:
i. `ADATE 3600` - θα προχωρήσει το ρολόι 1 ώρα.
ii. `ADATE -60` - θα κινήσει το ρολόι πίσω 1 λεπτό

# ARC
# ARC__R (Γραφικά)

Η **ARC** θα σχεδιάσει το τόξο ενός κύκλου ανάμεσα σε δύο καθορισμένα σημεία στο προσαρτημένο παράθυρο στο default ή καθορισμένο κανάλι. Τα άκρα του τόξου καθορίζονται χρησιμοποιώντας το σύστημα γραφικών συντεταγμένων.

Μπορούν να γραφούν πολλαπλά τόξα με μία μόνο εντολή **ARC.**

Τα σημεία των δύο άκρων του τόξου μπορούν να καθοριστούν είτε σε απόλυτες συντεταγμένες (σχετικές στη γραφική αρχή) ή σε σχετικές συντεταγμένες (σχετικές με το δρομέα γραφικών). Αν το πρώτο σημείο παραλείπεται τότε γράφεται ένα τόξο από το δρομέα γραφικών στο καθορισμένο σημείο διαμέσου της καθορισμένης γωνίας.

Το **ARC** θα σχεδιάζει πάντα με απόλυτες συντεταγμένες, ενώ το **ARC_R** θα σχεδιάζει πάντα σχετικά με το δρομέα γραφικών:

syntax: *x:= numeric__expression*
*y:= numeric__expression*
*angle:= numeric__expression* {in radians}
*point:= x,y*

*parameter__2:=* | **TO** *point, angle* 1
| *,point* **TO** *point,angle* 2

*parameter__1:=* | *point* **TO** *point,angle* 1
| **TO** *point,angle* 2

**ARC** [*channel,*] *parameter__1* * [*parameter__2*] *
**ARC__R** [*channel,*] *parameter__1* * [*parameter__2*] *

όπου: το 1 θα σχεδιάσει από το καθορισμένο σημείο μέχρι το επόμενο καθορισμένο σημείο μέσα από την καθορισμένη γωνία

το 2 θα σχεδιάσει από το σχεδιασμένο τελευταίο σημείο στο καθορισμένο σημείο μέσα από την καθορισμένη γωνία.

example: i. ARC 15,10 TO 40,40, PI/2

{σχεδιάζει ένα τόξο από το 10,10 στο 50,50 στρέφοντας κατά π/2, όλα τα σημεία είναι απόλυτα}

ii. ARC TO 50,50,PI/2

{σχεδιάζει ένα τόξο από το τελευταίο σημείο που σχεδιάστηκε στο 47,34 και στρέφοντας κατά P1/4 ακτίνα, όλα τα σημεία είναι σχετικά}

iii. ARC_R 10,10 TO 55,45,0.5

# AT (Παράθυρα)

Η **AT** επιτρέπει να τροποποιηθεί η θέση εκτύπωσης πάνω σε ένα νοητό δίκτυο σειρών/στηλών βασισμένο στο τρέχον μέγεθος χαρακτήρα. Η **AT** χρησιμοποιεί μια τροποποιημένη μορφή του συστήματος συντεταγμένων των pixels όπου (σειρά 0, στήλη 0) είναι στο πάνω αριστερά άκρο του παράθυρου.

syntax: *line:= numeric_expression*
*column:= numeric_expression*

**AT** [*channel*,] *line* , *column*

example: `AT 10,20 : PRINT "This is at line 10 column 20"`

# AUTO

Η **AUTO** επιτρέπει την αυτόματη αρίθμηση εντολών, όταν εισάγονται προγράμματα, κατευθείαν από τον υπολογιστή. Το **AUTO** θα δημιουργήσει τον επόμενο αριθμό σε ακολουθία και θα εισάγει τότε το διορθωτή σειράς της SuperBASIC, καθώς πληκτρολογείται η σειρά. Αν υπάρχει ήδη η σειρά τότε ένα αντίγραφο της παρουσιάζεται μαζί με τον αριθμό σειράς. Πατώντας το **ENTER** σε οποιοδήποτε σημείο στη σειρά θα ελέγξει τη σύνταξη όλης της σειράς και θα την εισάγει στο πρόγραμμα.

Η **AUTO** ακυρώνεται πατώντας

CTRL + space

syntax:
*first_line*:= *line_number*
*gap*:= *numeric_expression*

**AUTO** [*first_line*] [,*gap*]

example:
```
i.   AUTO       Αρχίζει στη γραμμή 100 με διάστημα 10
ii.  AUTO 10,5    "     "     "    10  "     "      5
iii. AUTO ,7      "     "     "   100  "     "      7
```

# BAUD (Επικοινωνίες)

Το **BAUD** καθορίζει την τιμή της baud για την επικοινωνία μέσα από τα δύο σειριακά κανάλια. Η ταχύτητα των καναλιών δεν μπορεί να οριστεί ανεξάρτητα.

syntax:
*rate*:= *numeric_expression*

**BAUD** *rate*

Η τιμή της αριθμητικής έκφρασης θα πρέπει να είναι μία από τις υποστηριγμένες baud κλίμακες στο QL:

75
300
600
1200
2400
4800
9600
19200 (μόνο μεταβίβαση)

Αν μία επιλεγμένη baud κλίμακα δεν υποστηρίζεται τότε θα δημιουργηθεί λάθος.

example:
```
i.  BAUD 9600
ii. BAUD print_speed
```

# BEEP ('Ηχος)

Η **BEEP** ενεργοποιεί τις λειτουργίες ήχου που είναι ενσωματωμένες στο QL. Το **BEEP** μπορεί να δεχθεί ένα μεταβλητό αριθμό παραμέτρων για να δώσει ποικίλα επίπεδα του ελέγχου πάνω στον ήχο που παράγεται. Η ελάχιστη προδιαγραφή απαιτεί για να συγκεκριμενοποιηθεί μόνο μια διάρκεια και ένα τόνο. Το **BEEP** όταν χρησιμοποιείται χωρίς παραμέτρους θα σκοτώσει κάθε ήχο που παράγεται.

syntax:

*duration*:= *numeric_expression* {range −32768 .. 32767}
*pitch*:= *numeric_expression* {range 0 .. 255}
*grad_x*:= *numeric_expression* {range −32768 .. 32767}
*grad_y*:= *numeric_expression* {range −8 .. 7}
*wrap*:= *numeric_expression* {range 0 .. 15}
*fuzzy*:= *numeric_expression* {range 0 .. 15}
*random*:= *numeric_expression* {range 0 .. 15}

**BEEP** [ *duration, pitch*
[, *pitch_2* , *grad_x*, *grad_y*
[, *wrap*
[, *fuzzy*
[, *random* ] ] ] ] ]

*duration* (διάρκεια) ορίζει τη διάρκεια του ήχου σε μονάδες των 72 microseconds. Μια διάρκεια του 0 θα τρέχει τον ήχο ώσπου να τερματιστεί από μία άλλη εντολή **BEEP**.

*pitch* (ύψος) προσδιορίζει το ύψος του ήχου. 'Ενας τόνος του 1 είναι ψηλός και του 255 είναι χαμηλός.

*pitch_2* (τόνος_2) προσδιορίζει ένα δεύτερο επίπεδο ύψους του ήχου ανάμεσα στο οποίο ο ήχος θα "ταλαντεύεται"

*grad_x*
*grad_y* (βαθμός_x) καθορίζει τα χρονικά διαστήματα μεταξύ των δύο διαβαθμίσεων του ύψους του ήχου. Το **grad_x** και **grad_y** ελέγχουν το ρυθμό με τον οποίο ο ήχος ταλαντεύεται ανάμεσα στις δύο διαβαθμίσεις.

*wrap* θα αναγκάσει τον ήχο να ανακυκλωθεί σε συγκεκριμένο αριθμό φορών. Αν το **wrap** είναι ίσο με 15 ο ήχος θα ανακυκλώνει για πάντα.

*fuzzy* περιγράφει το ποσοστό τυχαιότητας που προστίθεται στον ήχο.

*random*

# BEEPING ('Ηχος)

Η **BEEPING** είναι μια συνάρτηση που θα δώσει μηδέν (ψευδής) αν το QL δεν ηχεί τώρα και όχι μηδέν (αληθής) αν ηχεί.

syntax: BEEPING

example:
```
100 DEFine PROCedure be_quiet
110   BEEP
120 END DEFine
130 IF BEEPING THEN be_quiet
```

# BLOCK (Παράθυρα)

Η **BLOCK** θα γεμίσει ένα ορθογώνιο καθορισμένου μεγέθους και σχήματος, στη καθορισμένη θέση που είναι σχετική με την αρχή του προσαρτημένου παράθυρου στο συγκεκριμένο ή default κανάλι.

Το **BLOCK** χρησιμοποιεί το σύστημα συντεταγμένων (pixels) σημειοστοιχείων.

syntax:
*width*:= *numeric_expression*
*height*:= *numeric_expression*
*x*:= *numeric_expression*
*y*:= *numeric_expression*

BLOCK [*channel*,] *width*, *height*, *x*, *y* , *colour*

example: i.
```
BLOCK 10, 10, 5, 5, 7
```
**Ενα ορθογώνιο 10 X 10 pixels στη θέση 5,5**

ii.
```
100 REMark "bar chart"
110 CSIZE 3,1
120 PRINT "bar chart"
130 LET bottom = 100 : size = 20 : left = 10
140 FOR bar = 1 to 10
150   LET colour = RND(0 TO 255)
160   LET height = RND(2 TO 20)
170   BLOCK size, height, left+bar*size,
        bottom-height,0
180   BLOCK size-2, height-2, left+bar*size+1,
        bottom-height+1,colour
190 END FOR bar
```

Για τηλεοράσεις χρησιμοποιήστε LET colour=RND(0 TO 7)

# BORDER Παράθυρα

Η **BORDER** θα προσθέσει ένα περιθώριο στο προσαρτημένο περιθώριο στο συγκεκριμένο ή default κανάλι.

Για όλες τις λειτουργίες που θα ακολουθήσουν εκτός της **BORDER** το μέγεθος του περιθωρίου, μειώνεται για να αφήσει χώρο για το **BORDER.** Αν χρησιμοποιηθεί μια άλλη εντολή **BORDER** τότε το όλο μέγεθος του αρχικού περιθωρίου ξανατοποθετείται σχετικά με το περιθώριο που έχει προστεθεί, έτσι πολλαπλές εντολές **BORDER** έχουν το αποτέλεσμα να αλλάξουν το μέγεθος και το χρώμα ενός μόνο περιθωρίου. Πολλαπλά περιθώρια δε δημιουργούνται εκτός αν γίνει συγκεκριμένη ενέργεια.

**Αν η BORDER χρησιμοποιείται χωρίς να προσδιοριστεί ένα χρώμα τότε δημιουργείται ένα διαφανές περιθώριο του συγκεκριμένου πλάτους.**

syntax: *width*:= *numeric_expression*

BORDER [*channel*,] *size* [, *colour*]

example:

i.
```
BORDER 10,0,7                    {black and white stipple border}
```

ii.
```
100 REMark Lurid Borders
110 FOR thickness = 50 to 2 STEP -2
120    BORDER thickness, RND(0 TO 255)
130 END FOR thickness
140 BORDER 50
```

Για τηλεοράσεις χρησιμοποιήστε LET colour=RND(0 TO 7)

# CALL (Qdos)

Η γλώσσα μηχανής μπορεί να προσπελαστεί κατευθείαν από τη SuperBASIC χρησιμοποιώντας την εντολή **CALL**. Η **CALL** μπορεί να δεχθεί μέχρι 13 παραμέτρους διπλής λέξης που θα τοποθετηθούν στα δεδομένα στους καταχωρητές δεδομένων διευθύνσεων του 68008 (D1 μέχρι D7,A0 μέχρι A5) στη σειρά.

Δεν επιστρέφονται δεδομένα από την **CALL**.

syntax: *address*:= *numeric__expression*
*data*:= *numeric__expression*

CALL *address*, *[*data*]* {13 data parameters maximum}

example:
i. `CALL 262144,0,0,0`
ii. `CALL 262500,12,3,4,1212,6`

**Προειδοποίηση:** Ο καταχωρητής διευθύνσεων A6 δε θα πρέπει να χρησιμοποιείται σε ρουτίνες που καλούνται χρησιμοποιώντας αυτήν την εντολή. Για να επιστρέψετε στη SuperBASIC χρησιμοποιείστε τις εντολές:

```
MOVEQ #0,D0
RTS
```

# CHR$ (BASIC)

Η **CHR$** είναι μια συνάρτηση που θα επιστρέψει το χαρακτήρα του οποίου η τιμή καθορίζεται σα μια παράμετρος. Η **CHR$** είναι η αντίθετη της **CODE**.

syntax: CHR$ (*numeric__expression*)

example:
i. `PRINT CHR$(27)`
ii. `PRINT CHR$(65)`

i Τυπώνει το χαρακτήρα esc του ASCII
ii Τυπώνει το χαρακτήρα A

# CIRCLE
# CIRCLE_R (Γραφικά)

Η **CIRCLE** θα σχεδιάσει έναν κύκλο (ή μια έλλειψη με ορισμένη γωνία) στην οθόνη σε μια καθορισμένη θέση και μέγεθος. Η **CIRCLE** χρησιμοποιεί το σύστημα γραφικών συντεταγμένων. Ο κύκλος θα σχεδιαστεί στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι.

Η **CIRCLE** χρησιμοποιεί το σύστημα γραφικών συντεταγμένων και μπορεί να χρησιμοποιήσει απόλυτες συντεταγμένες (π.χ. σχετικές με την αρχή γραφικών) και σχετικές συντεταγμένες (π.χ. σχετικές με το δρομέα των γραφικών). Για τις σχετικές συντεταγμένες χρησιμοποιείστε το **CIRCLE_R.**

Μπορούν να σχεδιαστούν πολλαπλοί κύκλοι και ελλείψεις με ένα μόνο κάλεσμα της **CIRCLE.** Κάθε ομάδα παραμέτρων θα πρέπει να χωρίζεται από την άλλη με το σύμβολο (;).

Η δεσμευμένη λέξη **ELLIPSE** μπορεί να αντικαταστήσει τη **CIRCLE** αν χρειαστεί.

syntax:

*x*:= *numeric_expression*
*y*:= *numeric_expression*
*radius*:= *numeric_expression*
*eccentricity*:= *numeric_expression*
*angle*:= *numeric_expression* {range 0..2π}

*parameters*:= | *x*, *y*, 1
| *radius*, eccentricity, angle 2

όπου:

- το 1 θα σχεδιάσει έναν κύκλο
- το 2 θα σχεδιάσει μια έλλειψη συγκεκριμένης εκκεντρότητας και γωνίας.

CIRCLE [*channel*,] parameters *[; *parameters*] *

| | |
|---|---|
| x | οριζόντια απόσταση από την αρχή γραφικών ή το δρομέα γραφικών |
| y | κάθετη απόσταση από την αρχή γραφικών ή το δρομέα γραφικών |
| *radius* | ακτίνα του κύκλου |
| *eccentricity* | ο λόγος μεταξύ του μεγάλου και μικρού άξονα μιας έλλειψης |
| *angle* | η κατεύθυνση του μεγάλου άξονα της έλλειψης σχετικά με την κάθετο της οθόνης. Η γωνία πρέπει να καθορίζεται σε ακτίνια. |

example: i. CIRCLE 50,50,20
ii. CIRCLE 50,50,20,0.5,0

i Ενας κύκλος με κέντρο 50,50 και ακτίνα 20
ii Μια έλλειψη στη θέση 50,50 μέγιστο άξονα 20 εκκεντρότητα 0.5 ευθυγραμμισμένη με τον κατακόρυφο άξονα

# CLEAR

Η **CLEAR** θα καθαρίσει την περιοχή μεταβλητών της SuperBASIC για το τρέχον πρόγραμμα και θα ελευθερώσει χώρο μνήμης για το Qdos.

syntax: CLEAR

example: CLEAR

**Παρατήρηση:** Η **CLEAR** μπορεί να χρησιμοποιηθεί για να ξανατοποθετήσει το σύστημα της SuperBASIC σε μια γνωστή κατάσταση. Για παράδειγμα, αν ένα πρόγραμμα διακόπτεται (ή σταματά λόγω ενός λάθους) καθώς είναι σε μία διαδικασία, τότε η SuperBASIC είναι ακόμα σε αυτή τη διαδικασία ακόμα κι αν το πρόγραμμα έχει σταματήσει. Η **CLEAR** θα ξανατοποθετήσει τη SuperBASIC (Δείτε **CONTINUE, RETRY**).

# CLOSE (Συσκευές)

Η **CLOSE** θα κλείσει το συγκεκριμένο κανάλι. 'Οποιο παράθυρο είναι συνδεμένο με το κανάλι θα απενεργοποιηθεί.

syntax: *channel*:= #*numeric__expression*

CLOSE *channel*

example:
i. `CLOSE #4`
ii. `CLOSE #input_channel`

# CLS (Παράθυρο)

θα καθαρίσει το προσαρτημένο παράθυρο στο καθορισμένο ή default κανάλι στο τρέχον χρώμα χαρτιού, εξαιρώντας το περιθώριο αν έχει καθοριστεί. Το **CLS** θα δεχθεί μια προαιρετική παράμετρο που καθορίζει αν μόνο ένα μέρος του παραθύρου θα πρέπει να καθαριστεί.

syntax: *part*:= *numeric__expression*

CLS [*channel*,] [*part*]

όπου:

μέρος=0 ολόκληρη η οθόνη (default αν δεν υπάρχει παράμετρος)

μέρος=1 το πάνω μέρος εξαιρουμένης της σειράς του δρομέα

μέρος=2 το κάτω μέρος εξαιρούμενης της σειράς του δρομέα

μέρος=3 όλη η σειρά του δρομέα

μέρος=4 το δεξιό άκρο της σειράς του δρομέα συμπεριλαμβανομένης της θέσης του

example:
i. `CLS`
ii. `CLS 3`
iii. `CLS #2,2`

i Καθαριζει ολόκληρο το παράθυρο
ii Καθαρίζει τη γραμμή του δρομέα
iii Καθαρίζει το κάτω μέρος του παραθύρου στο κανάλι 2

# CODE

Η **CODE** είναι μια συνάρτηση που επιστρέφει τον εσωτερικό κωδικό που χρησιμοποιείται για να παραστήσει το συγκεκριμένο χαρακτήρα. Αν καθορίζεται με μία αλφαριθμητική τιμή τότε η **CODE** θα επιτρέψει την εσωτερική αναπαράσταση του πρώτου χαρακτήρα της.

Η **CODE** είναι το αντίθετο του **CHR$**.

syntax: **CODE** (*string_expression*)

example:
i. `PRINT CODE("A")` {prints 65}
ii. `PRINT CODE("SuperBASIC")` {prints 83}

# CONTINUE
# RETRY (Χειρισμός λάθους)

Η **CONTINUE** επιτρέπει τη συνέχιση ενός προγράμματος το οποίο έχει σταματήσει. Το **RETRY** επιτρέπει την επανεκτέλεση μιας εντολής ενός προγράμματος που έχει αναφέρει ένα λάθος.

syntax:
CONTINUE
RETRY

example:
```
CONTINUE
RETRY
```

**Προειδοποίηση:** 'Ενα πρόγραμμα μπορεί να συνεχίσει μόνο αν:

[1] Δεν έχουν προστεθεί νέες σειρές στο πρόγραμμα.

[2] Δεν έχουν προστεθεί νέες μεταβλητές στο πρόγραμμα.

[3] Δεν έχει αλλάξει η αρίθμηση εντολών.

Η τιμή των ήδη υπαρκτών μεταβλητών μπορεί να αλλάξει.

# COPY
# COPY_N (Συσκευές)

Το **COPY** θα αντιγράψει ένα αρχείο από μια συσκευή εισόδου σε μια έξοδο ωσότου να παρατηρηθεί η ένδειξη τέλους ενός αρχείου. Η **COPY_N** δε θα αντιγράψει την επικεφαλίδα (αν υπάρχει) που σχετίζεται με ένα αρχείο και θα επιτρέψει σε Microdrive αρχεία να αντιγραφούν σωστά σε έναν άλλο τύπο συσκευής.

Οι επικεφαλίδες συνδέονται με τις συσκευές με πίνακα περιεχομένων και θα πρέπει να μετακινηθούν χρησιμοποιώντας το **COPY_N** όταν αντιγράφετε σε συσκευές χωρίς πίνακα περιεχομένων, π.χ. **mdv1** είναι μια συσκευή με πίνακα περιεχομένων, το **ser1** είναι μία συσκευή χωρίς.

syntax: COPY *device* TO *device*
COPY_N *device* TO *device*

θα πρέπει να είναι δυνατό να εισάγετε από τη συσκευή πηγής και θα πρέπει να είναι δυνατό να το εξάγει στη συσκευή προορισμού.

example:
i. COPY mdv1_data_file TO con_
ii COPY neti_3 TO mdv1_data
iii. COPY_N mdv1_test_data TO ser1

i. Αντιγράφει στο default παράθυρο

ii. Αντιγράφει δεδομένα από σταθμό δικτύου στο mdv_data

iii. Αντιγράφει από mdv1_test_data στη σειριακή πόρτα 1 χωρίς επικεφαλίδα.

# COS (Μαθηματικές συναρτήσεις)

Η **COS** θα υπολογίζει το συνημίτονο του καθορισμένου ορίσματος.

syntax: *angle*:= *numeric__expression*

COS (angle)

example: i. `PRINT COS(theta)`

ii. `PRINT COS(3.141592654/2)`

# COT (Μαθηματικές συναρτήσεις)

Η **COT** θα υπολογίσει τη συνεφαπτομένη του καθορισμένου ορίασματος.

syntax: *angle*:= *numeric__expression*

COT (*angle*)

example: i. `PRINT COT(3)`

ii. `PRINT COT(3.141592654/2)`

# CSIZE (Παράθυρο)

Καθορίζει ένα νέο μέγεθος χαρακτήρα για το προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι. Το συνηθισμένο μέγεθος είναι 0,0 σε μορφή 512 και 2,0 σε μορφή 256.

Το πλάτος (width) καθορίζει το οριζόντιο μέγεθος του χαρακτήρα. Το ύψος (height) καθορίζει το κάθετο μέγεθος του χαρακτήρα. Το μέγεθος του χαρακτήρα κανονίζεται για να γεμίσει το διατιθέμενο διάστημα.

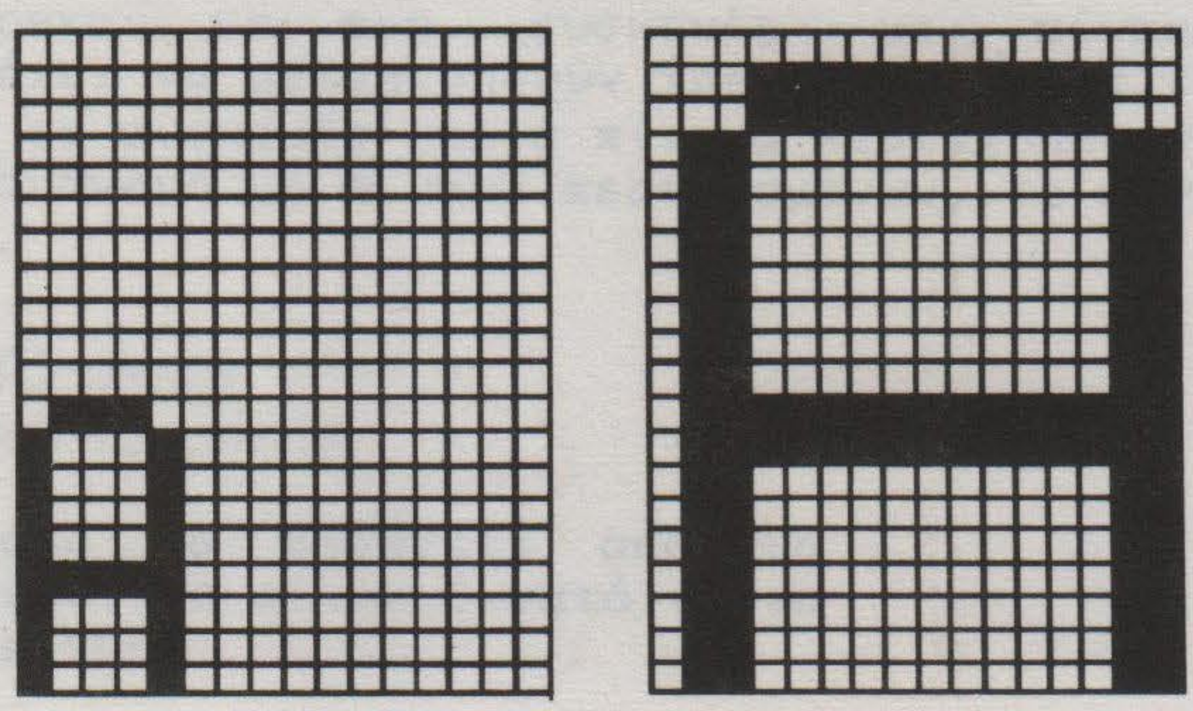


Figure A Character Square

| Πλάτος | Μέγεθος | Υψος | Μέγεθος |
|---|---|---|---|
| 0 | 6 pixels | 0 | 10 pixels |
| 1 | 8 pixels | 1 | 20 pixels |
| 2 | 12 pixels | | |
| 3 | 16 pixels | | |

syntax:
*width*:= *numeric_expression* {range 0..3}
*height*:= *numeric_expression* {range 0..1}

**CSIZE** [*channel*,] *width*, *height*

example:
i. `CSIZE 3,0`
ii. `CSIZE 3,1`

# CURSOR (Παράθυρα)

Η **CURSOR** επιτρέπει στο δρομέα οθόνης να τοποθετηθεί οπουδήποτε στο προσαρτημένο παράθυρο στο καθορισμένο ή default κανάλι.

Η **CURSOR** χρησιμοποιεί το σύστημα συντεταγμένων pixels σχετικό προς την αρχή του παράθυρου και καθορίζει τη θέση για την πάνω αριστερά γωνία του δρομέα, το μέγεθος του δρομέα εξαρτάται από το μέγεθος του χαρακτήρα που χρησιμοποιείται.

Αν η **CURSOR** χρησιμοποιείται με τέσσερεις παραμέτρους τότε το πρώτο ζευγάρι ερμηνεύεται σαν γραφικές συντεταγμένες (χρησιμοποιώντας το σύστημα γραφικών συντεταγμένων) και το δεύτερο ζευγάρι σαν τη θέση του δρομέα (στο σύστημα συντεταγμένων pixels) σχετικά ως προς το πρώτο σημείο.

Αυτό επιτρέπει στα διαγράμματα να σχεδιαστούν σχετικά εύκολα.

syntax: *x*:= *numeric_expression*
*y*:= *numeric_expression*

CURSOR [*channel*,] *x*, *y* [,*x*, *y*]

example:
i. `CURSOR 0,0`
ii. `CURSOR 20,30`
iii. `CURSOR 50,50,10,10`

# DATA
# READ
# RESTORE (BASIC)

Οι εντολές **READ, DATA** και **RESTORE** επιτρέπουν σε ενσωματωμένα δεδομένα, που περιέχονται σε ένα SuperBASIC πρόγραμμα να μεταβιβαστούν σε μεταβλητές την ώρα της εκτέλεσης του.

Η **DATA** χρησιμοποιείται για να σημειώσει και να προσδιορίσει τα δεδομένα, το **READ** παίρνει τα δεδομένα και τα μεταβιβάζει στις μεταβλητές και η **RESTORE** επιτρέπει σε καθορισμένα δεδομένα να επιλεχθούν.

**DATA:** επιτρέπει σε δεδομένα να καθοριστούν μέσα σε ένα πρόγραμμα. Τα δεδομένα μπορούν να διαβαστούν από μια επόμενη εντολή **READ** και να μεταβιβαστούν σε μεταβλητές. Μια **DATA** εντολή αγνοείται από τη SuperBASIC όταν απαντάται κατά τη διάρκεια ομαλής επεξεργασίας.

syntax: DATA *[*expression,*]*

**READ:** διαβάζει δεδομένα που περιέχονται στις εντολές **DATA** και τα μεταβιβάζει σε μία λίστα μεταβλητών. Αρχικά ο δείκτης δεδομένων τοποθετείται στην πρώτη εντολή DATA του προγράμματος και αυξάνει μετά από κάθε διάβασμα (READ). Ξανατρέξιμο του προγράμματος δεν επαναφέρει το δείκτη δεδομένων γι αυτό συνιστάται γενικά σε κάθε πρόγραμμα να υπάρχει μια εντολή άμεση RESTORE.

Αν προσπαθήσουμε READ για την οποία δεν υπάρχει DATA θα εμφανιστεί μήνυμα λάθους.

syntax: READ *[*identifier*,]*

**RESTORE:** ξανατοποθετεί το δείκτη δεδομένων, π.χ. τη θέση από την οποία οι επόμενες εντολές **READ** θα διαβάσουν τα δεδομένα τους. Αν η **RESTORE** ακολουθείται από έναν αριθμό σειράς τότε ο δείκτης δεδομένων τοποθετείται σε εκείνη τη σειρά.

Αν δεν προσδιοριστεί παράμετρος τότε ο δείκτης δεδομένων ξανατοποθετείται στην αρχή της πρώτης εντολής **DATA.**

syntax: RESTORE [*line_number*]

example: i.
```
100 REMark Data statement example
110 DIM weekdays$(7,4)
120 RESTORE
130 FOR count= 1 TO 7 :
      READ weekdays$(count)
140 PRINT weekday$
150 DATA "MON","TUE","WED","THUR","FRI"
160 DATA "SAT","SUN"
```

ii.
```
100 DIM month$(12,9)
110 RESTORE
120 REMark Data statement example
130 FOR count= 1 TO 12 :
      READ month$(count)
140 PRINT month$
150 DATA "January","February","March"
160 DATA "April","May","June"
170 DATA "July","August","September"
180 DATA "October","November","December"
```

**Προειδοποίηση:** Μια έμεση **RESTORE** δεν εκτελείται πριν τρέξει ένα πρόγραμμα. Αυτό επιτρέπει σε ένα πρόγραμμα να τρέξει με διαφορετικά σύνολα δεδομένων. Γιαυτό συμπεριλάβετε μια **RESTORE** ή εκτελέστε μια άμεση RESTORE ή **CLEAR** πριν τρέξετε το πρόγραμμα.

# DATE$
# DATE (Ρολόι)

Η **DATE$** είναι μια συνάρτηση που θα επιστρέψει την ημερομηνία και την ώρα που περιέχονται στο ρολόι του QL.

Η μορφή της αλφαριθμητικής που επιστρέφεται από το **DATE$** είναι:

"yyyy mmm dd hh:mm:ss"

όπου:

yyyy είναι το έτος 1984, 1985 κ.τ.λ.

mmm είναι ο μήνας 01=Ιάν., 02=Φεβ. κ.τ.λ.

dd είναι η μέρα 01 ως 28, 29, 30, 31

hh είναι η ώρα από 0 ως 23

mm είναι το λεπτό από 0 ως 59

ss είναι τα δευτερόλεπτα από 0 ως 59

Η **DATE** θα δώσει την ημερομηνία σαν έναν αριθμό κινητής υποδιαστολής που μπορεί να χρησιμοποιηθεί για να φυλάξει ημερομηνίες και χρόνους σε ένα συμπυκνωμένο τύπο.

Αν η **DATE$** χρησιμοποιείται με μία αριθμητική παράμετρο τότε η παράμετρος θα ερμηνευθεί σαν μία ημερομηνία σε μορφή κινητής υποδιαστολής και θα μετατραπεί σε αλφαριθμητική ημερομηνία.

syntax: DATE$
DATE$(*numeric_expression*)

example: i. PRINT DATE$
Παίρνει το χρόνο από το ρολόι.

ii. PRINT DATE$(234567)
Μετατρέπει το 234567 σε ημερομηνία.

# DAY$ (Ρολόι)

Η **DAY$** είναι μία συνάρτηση που θα δώσει την τρέχουσα μέρα της εβδομάδας. Αν προσδιοριστεί παράμετρος, τότε η **DAY$** θα ερμηνεύσει αυτή σαν μία ημερομηνία και θα επιστρέψει την ανάλογη μέρα της εβδομάδας.

syntax: DAY$

DAY$ (*numeric_expression*)

example:

i. `PRINT DAY$`

Παίρνει την ημέρα από το ρολόι.

ii. `PRINT DAY$(234567)`

Δίνει την ημέρα που ορίζεται από 234567 δευτερόλεπτα.

# DEFine FuNction
# END DEFine

Συναρτήσεις και Διαδικασίες

Η εντολή DEFine FUNction καθορίζει μια συνάρτηση στη Super BASIC. Η ακολουθία των εντολών μεταξύ της εντολής DEFine FUNction και της END DEFine αποτελούν τη συνάρτηση. Ο καθορισμός της συνάρτησης μπορεί επίσης να περιλαμβάνει μια

λίστα τυπικών παραμέτρων η οποία θα εφοδιάζει με δεδομένα τη συνάρτηση. Οι τυπικές και πραγματικές παράμετροι πρέπει να περικλείονται σε αγκύλες. Αν η συνάρητηση δεν απαιτεί παραμέτρους τότε δεν χρειάζεται να συμπεριλάβουμε ένα άδειο σύνολο αγκύλης στον καθορισμό της συνάρτησης. Οι τυπικές παράμετροι πέρνουν τον τύπο και τα χαρακτηριστικά τους από τις πραγματικές παραμέτρους. Ο τύπος των δεδομένων τα οποία επιστρέφονται από την συνάρτηση εξαρτάται από τον τύπο που καθορίζεται στο τέλος του αναγνωριστή της συνάρτησης. Ο τύπος των δεδομένων που επιστρέφονται στη εντολή RETURN πρέπει να συμφωνούν.

Μια απάντηση επιστρέφεται με την εντολη RETURN στη μεταβλητή που προστίθεται μετά το RETURN. Ο τύπος των δεδομένων που επιστρέφονται είναι ο ίδιος με τον τύπο της μεταβλητής.

Μια συνάρτηση δραστηριοποιείται αν το όνομά της συμπεριληφθεί μέσα σε μία έκφραση.

Κλήσεις συναρτήσεων στην SuperBASIC μπορεί να αυτοκαλούνται, δηλ. μια συνάρτηση μπορεί να καλεί τον εαυτό της άμεσα ή έμμεσα διαμέσου μιας ακολουθίας κλήσεων.

syntax:

*formal__parameters*:= (*expression* *|, *expression*| *)
*actual__parameters*:= (*expression* *|, *expression*| *)

*type*:= | $
| %
|

**DEF FuNction** *identifier type* |*formal__parameters*|
|**LOCal** *identifier* *|, *identifier*| *|
*statements*
**RETurn** *expression*
**END DEFine**

Η εντολή RETURN μπορεί να μπεί σε οποιαδήποτε θέση μέσα στο σώμα της συνάρτησης. Οι τοπικές μεταβλητές ορίζονται πριν από κάθε εκτελέσιμη εντολή της συνάρτησης.

example:

```
10 DEFine FuNction mean(a, b, c)
20    LOCal answer
30    LET answer = (a + b + c)/3
40    RETurn answer
50 END DEFine
60 PRINT mean(1,2,3)
```

Παρατήρηση: Για να αυξηθεί η ακρίβεια των προγραμμάτων, το όνομα της συνάρτησης μπορεί να μπεί στο τέλος της END FUNction παρόλον ότι η SuperBASIC δεν θα το ελέγξει.

# DEFine PROCedure END DEFine (Συναρτήσεις και διαδικασίες)

Η **DEFine PROCedure** καθορίζει μία SuperBASIC διαδικασία. Η ακολουθία των εντολών ανάμεσα στην εντολή **DEFine PROCedure** και την εντολή **END DEFine** καθορίζει τη διαδικασία. Ο ορισμός διαδικασίας μπορεί επίσης να περιλαμβάνει μία λίστα από τυπικές παραμέτρους που θα εφοδιάσουν με δεδομένα τη διαδικασία. Οι τυπικές παράμετροι, θα πρέπει να περικλείονται σε παρενθέσεις για τον ορισμό διαδικασίας, αλλά οι παρενθέσεις δεν είναι απαραίτητες όταν η διαδικασία εκτελείται. Αν η διαδικασία δεν απαιτεί παραμέτρους, τότε δεν είναι ανάγκη να περιλάβετε ένα κενό σύνολο παρενθέσεων στον ορισμό διαδικασίας.

Οι τυπικές παράμετροι παίρνουν τον τύπο και τα χαρακτηριστικά τους από τις αντίστοιχες πραγματικές παραμέτρους.

Οι μεταβλητές μπορούν να οριστούν να είναι τοπικές σε μία διαδικασία. Οι τοπικές μεταβλητές δεν έχουν επίδραση σε παρόμοια ονομασμένες μεταβλητές έξω από τη διαδικασία. Αν είναι απαραίτητο, οι τοπικές μεταβλητές με δείκτες θα πρέπει να παίρνουν διαστάσεις μέσα στις τοπικές εντολές.

Η διαδικασία καλείται εισάγοντας το όνομά της σαν το πρώτο κομμάτι σε μία SuperBASIC εντολή μαζί με μία λίστα από πραγματικές παραμέτρους. Καλέσματα διαδικασιών στη SuperBASIC είναι επαναληπτικά, δηλαδή μια συνάρτηση μπορεί να καλέσει τον εαυτό της απευθείας ή έμμεσα διαμέσου μιάς ακολουθίας άλλων κλήσεων.

Είναι πιθανό να θεωρηθεί ο ορισμός μιας διαδικασίας σαν ένας ορισμός εντολής σε SuperBASIC. Πολλές από τις εντολές συστήματος είναι οι ίδιες ορισμένες σαν διαδικασίες.

syntax:

*formal_parameters*:= (*expression* *[, *expression*]*)
*actual_parameters*:= *expression* *[, *expression*]*

**DEFine PROCedure** *identifier* [*formal_parameters*]
  [**LOCal** *identifier* *[, *identifier*]*]
  *statements*
  [RETurn]
END DEFine

Το **RETURN** μπορεί να εμφανιστεί σε οποιαδήποτε θέση μέσα στο σώμα της διαδικασίας. Αν εμφανιστεί η εντολή LOCAL πρέπει να βρίσκεται πριν από την πρώτη εκτελέσιμη εντολή της διαδικασίας. Η εντολή END DEFine θα λειτουργήσει σαν ένα αυτόματο RETURN.

example: i.
```
100 DEFine PROCedure start_screen
110    WINDOW 100,100,10,10
120    PAPER 7 : INK 0 : CLS
130    BORDER 4,255
140    PRINT "Hello Everybody"
150 END DEFine
160 start_screen
```

ii.
```
100 DEFine PROCedure slow_scroll(scroll_limit)
110    LOCal count
120    FOR count = 1 TO scroll_limit
130       SCROLL 2
140    END FOR count
150 END DEFine
160 slow_scroll 20
```

**Παρατήρηση:** Για να βελτιώσετε τη νομιμότητα των προγραμμάτων, το όνομα της διαδικασίας μπορεί να προστεθεί στην END DEFine εντολή, παρ' όλο που το όνομα δε θα ελεγχθεί από τη SuperBASIC.

# DEG (Μαθηματικές συναρτήσεις)

Η **DEG** είναι μία συνάρτηση που θα μετατρέψει μια γωνία εκφρασμένη σε ακτίνια σε μια γωνία εκφρασμένη σε μοίρες.

syntax: DEG(*numeric_expression*)

example: `PRINT DEG(PI/2)` {will print 90}

# DELETE (Microdrives)

Το **DELETE** θα απομακρύνει ένα αρχείο από τον πίκανα περιεχομένων της μικροκασέτας στο καθορισμένο (microdrive).

syntax: DELETE *device*

Ο ορισμός συσκευής πρέπει να είναι ενος microdrive

example:
i. `DELETE mdv1_old_data`
ii. `DELETE mdv1_letter_file`

# DIM (Μεταβλητές με δείκτες)

Προσδιορίζει μία μεταβλητή με δείκτες στη SuperBASIC. Οι αλφαριθμητικές, ακέραιες και πραγματικές μεταβλητές με δείκτες μπορούν να προσδιοριστούν. Οι αλφαριθμητικές μεταβλητές με δείκτες διαχειρίζονται συγκεκριμένου μήκους αλφαριθμητικές με δείκτες και ο τελικός δείκτης λαμβάνεται σαν το μήκος της αλφαριθμητικής.

Οι δείκτες μεταβλητών με δείκτες παίρνουν τιμές από το 0 ως τη μέγιστη τιμή του δείκτη που καθορίζεται από την εντολή **DIM**. 'Ετσι το **DIM** θα ενεργοποιήσει μία μεταβλητή με δείκτες με ένα ή περισσότερα στοιχεία σε κάθε διάσταση από ότι καθορίζεται πραγματικά.

'Οταν μία μεταβλητή με δείκτη καθορίζεται παίρνει αρχική τιμή μηδέν αν είναι μια αριθμητική μεταβλητή με δείκτες και μήκος μηδέν αν είναι αλφαριθμητική μεταβλητή.

syntax: *index*:= *numeric_expression*
*array*:= *indentifier*(*index* *[, *index*]*)
**DIM** *array* *[, *array*]*

example:
i. `DIM string_array$(10,10,50)`
ii. `DIM matrix(100,100)`

# DIMN (Μεταβλητές με δείκτες)

Η **DIMN** είναι μια συνάρτηση που θα δώσει τη μεγαλύτερη τιμή μιας συγκεκριμένης διάστασης μιας καθορισμένης μεταβλητής με δείκτες. Αν μία διάσταση δεν καθοριστεί τότε η πρώτη διάσταση υποτίθεται. Αν η καθορισμένη διάσταση δεν υπάρχει ή ο αναγνωριστής δεν είναι μια μεταβλητή με δείκτη τότε επιστρέφεται η τιμή μηδέν.

syntax: *array*:= *identifier*
*index*:= *numeric_expression* {1 for dimension 1, etc.}

**DIMN**(*array* [,dimension] )

example: Θεωρούμε μια μεταβλητή με δείκτες που ορίζεται ως:

```
DIM a(2,3,4)
i.    PRINT DIMN(A,1)      i θα τυπώσει 2
ii.   PRINT DIMN(A,2)      ii θα τυπώσει 3
iii.  PRINT DIMN(A,3)      iii θα τυπώσει 4
iv.   PRINT DIMN(A)        iv θα τυπώσει 2
v.    PRINT DIMN(A,4)      v θα τυπώσει 0
```

# – DIR (Microdrives)

Η **DIR** θα πάρει και θα εμφανίσει στο προσαρτημένο παράθυρο στο συγκεκριμενο ή σαν default κανάλι τον πίνακα περιεχομένων της μικροκασέτας στο συγκεκριμένο microdrives.

syntax: DIR *device*

Ο καθορισμός της συσκευής θα πρέπει να είναι μία έγκυρη συσκευή microdrives.

Η μορφή του πίνακα περιεχομένων που εμφανίζεται από τη **DIR** είναι:

free_sectors:= ο αριθμός ελεύθερων τομέων

available_sectors:= ο μέγιστος αριθμός των τομέων σ' αυτή τη μικροκασέτα

file_name:= ένα όνομα αρχείου της SuperBASIC

screen format:
*Volume name*
*free_sectors* / *available_sectors* sectors
*file_name*
–
*file_name*

example:
i. `DIR mdv1_`
ii. `DIR "mdv2_"`
iii. `DIR "mdv" & microdrive_number$ & "_"`

screen format:
```
BASIC_
183 / 221 sectors
demo_1
demo_1_old
demo_2
```

# DIV (Τελεστής)

Η **DIV** είναι ένας τελεστής που θα εκτελέσει μια ακέραια διαίρεση.

syntax: *numeric__expression* **DIV** *numeric__expression*

example: i. `PRINT 5 DIV 2`
ii. `PRINT -5 DIV 2`

# DLINE (BASIC)

Η **DLINE** θα διαγράψει μια μοναδική σειρά ή μία ομάδα από σειρές από ένα πρόγραμμα σε SuperBASIC.

syntax: *range*:= | *line__number* **TO** *line__number* 1
| *line__number* **TO** 2
| **TO** *line__number* 3
| *line__number* 4

**DLINE** *range** [,*range*] *

όπου:

- το 1 θα διαγράψει μια ομάδα από σειρές

- το 2 θα διαγράψει από τη συγκεκριμένη σειρά ως το τέλος

- το 3 θα διαγράψει από την αρχή ως τη συγκεκριμένη σειρά

- το 4 θα διαγράψει τη συγκεκριμένη σειρά

example: i. `DLINE 10 TO 70, 80, 200 TO 400`

(θα διαγράψει τις σειρές από το 10 ως και τη 70, τη σειρά 80 και τις σειρές 200 ως και 400)

ii. `DLINE`

(δε θα διαγράψει τίποτα)

# EDIT

Η εντολή **EDIT** εισάγει το διορθωτή γραμμής της SuperBASIC.

Η εντολή **EDIT** είναι στενά συνδεδεμένη με την εντολή **AUTO,** ενώ η μόνη διαφορά είναι στα defaults. Η **EDIT** defaults σε μια αύξηση σειράς σε μηδέν και για αυτό θα διορθώσει μια σειρά εκτός αν μία δεύτερη παράμετρος καθορίζεται για να ορίσει μια αύξηση σειράς.

Αν η συγκεκριμένη σειρά υπάρχει ήδη τότε η σειρά εμφανίζεται και η διόρθωση μπορεί να αρχίσει. Αν η σειρά δεν υπάρχει τότε ο αριθμός σειράς εμφανίζεται και η σειρά μπορεί να εισαχθεί.

Ο δρομέας μπορεί να μετακινηθεί μέσα στη σειρά της διόρθωσης χρησιμοποιώντας τα συνηθισμένα πλήκτρα του QL.

| | |
|---|---|
| → cursor right | δρομέας δεξιά |
| ← cursor left | δρομέας αριστερά |
| ↑ cursor up | δρομέας πάνω. Δίνει προηγούμενη σειρά |
| ↓ cursor down | δρομέας κάτω. Δίνει επόμενη σειρά |
| CTRL → | διαγράφει δεξιό χαρακτήρα |
| CTRL ← | διαγράφει αριστερό χαρακτήρα |

'Οταν η σειρά είναι σωστή πατώντας το **ENTER** θα εισάγει τη σειρά μέσα στο πρόγραμμα.

Αν έχει καθοριστεί μια αύξηση τότε η επόμενη σειρά στην ακολουθία θα διορθωθεί αλλιώς η διόρθωση θα τερματιστεί.

syntax: *increment:= numeric__expression*

**EDIT** *line__number* [,*increment*]

example:
i. `EDIT 10` (διόρθωση σειράς 10 μόνο)
ii. `EDIT 20,10` (διόρθωση των σειρών 20,30 κ.τ.λ.)

# EOF (Συσκευές)

Η **EOF** είναι μία συνάρτηση που θα καθορίσει αν η συνθήκη τέλος ενός αρχείου έχει εμφανιστεί σε ένα συγκεκριμένο κανάλι. Αν το **EOF** χρησιμοποιείται χωρίς καθορισμό καναλιού τότε η **EOF** θα καθορίσει αν το τέλος ενός προγράμματος με ενσωματωμένες εντολές δεδομένων (DATA) έχει βρεθεί.

syntax: **EOF** [(*channel*)]

example:
i.
```
IF EOF(#6) THEN STOP
```
ii.
```
IF EOF THEN PRINT "Out of data"
```

# EXEC
# EXEC_W (Qdos)

Η **EXEC** και η **EXEC_W** θα φορτώσουν μια ακολουθία από προγράμματα και θα τα εκτελέσουν παράλληλα.

Το **EXEC** θα επιστρέψει στον επεξεργαστή εντολών αφού έχει αρχίσει η εκτέλεση όλων των επεξεργασιών. Το **EXEC_W** θα περιμένει ως ότου όλες οι επεξεργασίες να έχουν τερματιστεί πριν επιστρέψει.

syntax: *program*:=device Χρησιμοποιείται για να καθορίσει ένα αρχείο σε Microdrive που περιέχει το πρόγραμμα.

**EXEC** *program*

example:
i.
```
EXEC mdv1_communcations
```
ii.
```
EXEC_W mdv1_printer_process
```

# EXIT (Επανάληψη)

Η **EXIT** θα συνεχίσει την επεξεργασία μέχρι το **END** της ονομασμένης **FOR** ή **REPeat** δομής.

syntax: EXIT *identifier*

example: i.
```
100 REM start looping
110   LET count = 0
120 REPeat loop
130   LET count = count + 1
140   PRINT count
150   IF count = 20 THEN EXIT loop
160 END REPeat loop
```
Ο βρόχος θα τελειώσει όταν ο μετρητής γίνει ίσος με 20.

ii.
```
100 FOR n = 1 TO 1000
110   REM program statements
120   REM program statements
130   IF RND >.5 THEN EXIT n
140 END FOR n
```
Ο βρόχος θα τελειώσει όταν ένας τυχαίος αριθμός που θα δημιουργηθεί θα είναι μεγαλύτερος του 0.5

# EXP (Μαθηματικές συναρτήσεις)

Η EXP θα δώσει την τιμή του e υψωμένη στη δύναμη της καθορισμένης παραμέτρου.

syntax: **EXP**(*numeric_expression*) {range −500..500}

example: i.
```
PRINT EXP(3)
```
ii.
```
PRINT EXP(3.141592654)
```

# FILL (Γραφικά)

Η **FILL** θα ενεργοποιεί ή θα απενεργοποιεί το γέμισμα των γραφικών. Η **FILL** θα γεμίσει κάθε κλειστό σχήμα σχεδιασμένο με διαδικασίες γραφικών ή γραφικών χελώνας καθώς ζωγραφίζεται. Τα ανοιχτά σχήματα θα πρέπει να χωρίζονται σε μικρότερα κλειστά σχήματα για να εγγυηθούν ένα πετυχημένο γέμισμα.

'Οταν έχετε τελειώσει το γέμισμα, θα πρέπει να κληθεί η **FILL 0**.

syntax: *switch*:= *numeric_expression* {range 0..1}

**FILL** [*channel*,] *switch*

example: i.
```
FILL 1:LINE 10,10 TO 50,50 TO 30,90 TO 10,10:FILL 0
```
θα σχεδιάσει ένα γεμάτο τρίγωνο.

ii.
```
FILL 1:CIRCLE 50,50,20:FILL 0
```
θα σχεδιάσει ένα γεμάτο κύκλο.

# FILL$ (Αλφαριθμητική μεταβλητή με δείκτες)

Η **FILL$** είναι μία συνάρτηση που θα δώσει μία αλφαριθμητική ενός συγκεκριμένου μήκους γεμισμένη με την επανάληψη ενός ή δύο χαρακτήρων.

syntax: FILL$ (*string_expression*,*numeric_expression*)

Η αλφαριθμητική έκφραση που εφοδιάζεται στο **FILL** θα πρέπει να έχει μήκος ενός ή δύο χαρακτήρων

example:
```
i.    PRINT FILL$("a",5)          {will print aaaaa}
ii.   PRINT FILL$("oO",7)         {will print oOoOoOo}
iii.  LET a$ = a$ & FILL$(" ", 10)
```

# FLASH (Παράθυρο)

Η **FLASH** ανοίγει και κλείνει τη κατάσταση αναβοσβήματος. Η **FLASH** έχει αποτέλεσμα μόνο στη μορφή της χαμηλής διακριτικότητας. Η **FLASH** θα επηρεάσει το προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι.

syntax: *switch*:= *numeric_expression* {range 0..1}
FLASH [*channel*,] *switch*

όπου

- **switch = 0** θα απενεργοποιήσει το **FLASH**
- **switch = 1** θα ενεργοποιήσει το **FLASH**

example:

```
100 PRINT "A ";
110 FLASH 1
120 PRINT "flashing ";
130 FLASH 0
140 PRINT "word"
```

**Προειδοποίηση:** Αν γράψετε πάνω από ένα μέρος ενός χαρακτήρα που αναβοσβήνει μπορεί να παράγει νοθευμένα αποτελέσματα και θα πρέπει να αποφεύγεται.

# FOR
# END FOR

**Επανάληψη**

Η εντολή **FOR** επιτρέπει σε μία ομάδα από εντολές της BASIC να επαναλαμβάνονται σε ένα ελεγχόμενο αριθμό φορών. Η εντολή **FOR** μπορεί να χρησιμοποιηθεί σε μακριά και σε σύντομη μορφή.

Η **NEXT** και **END FOR** μπορούν να χρησιμοποιηθούν μαζί μέσα στην ίδια **FOR** ανακύκλωση για να εφοδιάσουν ένα επίλογο της ανακύκλωσης, π.χ. μία ομάδα από εντολές SuperBASIC που δε θα εκτελεστούν αν βγούμε από την ανακύκλωση μέσω μιας εντολής **EXIT**, αλλά θα εκτελεστούν αν η ανακύκλωση **FOR** τερματιστεί φυσιολογικά.

define: *for__item*:= | *numeric__expression*
| *numeric__exp* **TO** *numeric__exp*
| *numeric__exp* **TO** *numeric__exp* **STEP** *numeric__exp*

*for__list*:= *for__item* *[, *for__item*]*

**Σύντομη μορφή**

Η εντολή **FOR** ακολουθείται στην ίδια λογική σειρά από μία ακολουθία εντολών SuperBASIC. Η ακολουθία των εντολών τότε εκτελείται επανειλημένα κάτω από τον έλεγχο της εντολής **FOR.** Όταν η εντολή **FOR** έχει εξαντληθεί συνεχίζεται η επεξεργασία από την επόμενη σειρά. Η εντολή **FOR** δεν απαιτεί τα αντίστοιχα **NEXT** ή **END FOR** για να τελειώσει. Ανακυκλώσεις **FOR** αυτής της μορφής πρέπει να δικτυώνονται.

syntax: **FOR** *variable* = *for__list* : *statement* *[: *statement*]*

example:
i.
```
FOR i = 1, 2, 3, 4 TO 7 STEP 2 : PRINT i
```
ii.
```
FOR element = first TO last : LET buffer(element) = 0
```

**Μακριά μορφή**

Η εντολή **FOR** είναι η τελευταία εντολή στη σειρά. Οι επόμενες γραμμές περιέχουν μια σειρά από εντολές SuperBASIC που τελειώνουν με μια εντολή **END FOR.** Οι εντολές που περικλείονται ανάμεσα στη **FOR** και την **END FOR** εκτελούνται κάτω από τον έλεγχο της εντολής **FOR**

syntax: **FOR** *variable* = *for__list*
*statements*
**END FOR** *variable*

example:
```
100 INPUT "data please" ! x
110 LET factorial = 1
120 FOR value = x TO 1 STEP -1
130    LET factorial = factorial * value
140    PRINT x !!!! factorial
150    IF factorial>1E20 THEN
160       PRINT "Very large number"
170       EXIT value
180    END IF
190 END FOR value
```

Προειδοποίηση: Μια πραγματική μεταβλητή θα πρέπει να χρησιμοποιείται για να ελέγξει μια ανακύκλωση **FOR.**

# FORMAT (Microdrives)

Η **FORMAT** θα σχηματίσει και θα ετοιμάσει για χρήση τη μικροκασέτα που περιέχεται στη συγκεκριμένη μονάδα.

syntax: FORMAT [channel,] *device*

Η συσκευή καθορίζει τη μονάδα μικροκασέτας που θα χρησιμοποιηθεί για το σχηματισμό και το αναγνωριστικό μέρος της προδιαγραφής χρησιμοποιείται σαν το μέσο ή το όνομα τόμου για εκείνη τη μικροκασέτα. Η **FORMAT** θα γράψει τον αριθμό των καλών τομέων και το συνολικό αριθμό των διατιθέμενων τομέων της μικροκασέτας στο default ή στο συγκεκριμένο κανάλι.

Είναι χρήσιμο να σχηματίσετε μία νέα μικροκασέτα πολλές φορές πριν τη χρήση της. Αυτό τακτοποιεί την επιφάνεια της ταινίας και της δίνει μεγαλύτερη χωρητικότητα.

example: i. `FORMAT mdv1_data_cartridge`
ii. `FORMAT mdv2_wp_letters`

Προειδοποίηση: Η **FORMAT** μπορεί να χρησιμοποιηθεί για να ξανασχηματιστεί μια χρησιμοποιημένη μικροκασέτα. Παρ' όλα αυτά, όλα τα δεδομένα που περιέχονται σε εκείνη τη μικροκασέτα θα χαθούν.

# GOSUB (BASIC)

Για συμβατότητα με άλλες BASIC, η SuperBASIC υποστηρίζει τη **GOSUB**. Η **GOSUB** μεταβιβάζει την επεξεργασία στο συγκεκριμένο αριθμό σειράς. Μια εντολή **RETurn** θα μεταβιβάσει την επεξεργασία πίσω στην εντολή που ακολουθεί τη **GOSUB**.

Ο καθορισμός του αριθμού σειράς μπορεί να είναι μία έκφραση.

syntax: GOSUB *line_number*

example i. `GOSUB 100`
ii. `GOSUB 4*select_variable`

Παρατήρηση: Οι δομές ελέγχου που διατίθενται στη SuperBASIC κάνουν την εντολή **GOSUB** περιττή.

# GOTO (BASIC)

Για συμβιβαστότητα με άλλες BASIC, η SuperBASIC κρατάει την εντολή **GOTO**. Η **GOTO** θα μεταφέρει χωρίς συνθήκη την επεξεργασία στο συγκεκριμένο αριθμό εντολής. Ο καθορισμός του αριθμού εντολής μπορεί να είναι μία έκφραση.

syntax: GOTO *line_number*

example: i. `GOTO program_start`
ii. `GOTO 9999`

Παρατήρηση: Οι διατιθέμενες δομές ελέγχου στη SuperBASIC κάνουν την εντολή **GOTO** περιττή.

# IF
# THEN
# ELSE
# END IF

Η εντολή **IF** επιτρέπει την εξέταση συνθηκών και το αποτέλεσμα αυτής της εξέτασης να ελέγχει τη ροή του προγράμματος που ακολουθεί.

Η εντολή **IF** μπορεί να χρησιμοποιηθεί σε μακριά και σύντομη μορφή:

Η δεσμευμένη λέξη **THEN** ακολουθείται στην ίδια λογική σειρά από μία ακολουθία δεσμευμένων λέξεων της SuperBASIC. Αν αυτή η ακολουθία από εντολές της SuperBASIC, περιέχει μία δεσμευμένη λέξη **ELSE** και αν η έκφραση στην εντολή **IF** είναι αληθής (υπολογιζόμενη όχι μηδέν) τότε οι εντολές ανάμεσα στη δεσμευμένη λέξη **THEN** και την **ELSE** εκτελούνται. Αν η συνθήκη είναι ψευδής (υπολογιζόμενη μηδέν) τότε οι εντολές ανάμεσα στο **ELSE** και το τέλος της σειράς εκτελούνται.

Αν η ακολουθία των εντολών της SuperBASIC δεν περιέχει δεσμευμένη λέξη **ELSE** και αν η έκφραση στην εντολή **IF** είναι αληθής, τότε οι εντολές ανάμεσα στη δεσμευμένη λέξη **IF** και το τέλος της σειράς εκτελούνται. Αν η έκφραση είναι ψευδής τότε η εκτέλεση συνεχίζει στην επόμενη σειρά.

syntax: *statements*:= *statement* *[: *statement*] *
IF *expression* THEN *statements* [:ELSE *statements*]

example:
```
i.    IF a=32 THEN PRINT "Limit" : ELSE PRINT "OK"
ii.   IF test > maximum THEN LET maximum = test
iii.  IF "1"+1=2 THEN PRINT "coercion OK"
```

**Μακριά μορφή**

Η δεσμευμένη λέξη THEN είναι η τελευταία λέξη στη λογική σειρά. Μια ακολουθία από εντολές της SuperBASIC ακολουθεί σε επόμενες σειρές τελειώνοντας με την εντολή ENDIF. Οι εντολές αυτές εκτελούνται αν η εντολή μετά το IF δώσει μηδέν. Η ELSE και η ακολουθία εντολών της SuperBASIC είναι προαιρετικά.

**Μακριά μορφή 2**

Η δεσμευμένη λέξη **THEN** είναι η τελευταία λέξη στη λογική σειρά. Μία ακολουθία από εντολές της SuperBASIC ακολουθεί σε επόμενες σειρές τελειώνοντας με τη δεσμευμένη λέξη **ELSE**. Αν η έκφραση που περιέχεται στην εντολή **IF** υπολογίζεται να μην είναι μηδέν τότε αυτή η πρώτη ακολουθία των εντολών της SuperBASIC εκτελείται. Μετά τη δεσμευμένη λέξη **ELSE**, μια δεύτερη ακολουθία από εντολές της SuperBASIC εισάγεται, τελειώνοντας με τη δεσμευμένη λέξη **END IF**. Αν η έκφραση που υπολογίζεται από την εντολή **IF** είναι μηδέν τότε αυτή η δεύτερη ακολουθία από εντολές της SuperBASIC εκτελείται.

syntax:
IF *expression* THEN
    *statements*
[ELSE
    *statements*]
END IF

example:
```
100 LET limit = 10
110 INPUT "Type in a number" ! number
120 IF number > limit THEN
130    PRINT "Range error"
140 ELSE
150    PRINT "Inside limit"
160 END IF
```

Παρατήρηση: Σε όλες τις τρεις μορφές της εντολής **IF** η **THEN** είναι προαιρετική. Στη σύντομη μορφή μπορεί να αντικατασταθεί από μια διπλή τελεία (:) για να ξεχωρίζει το τέλος του **IF** από την αρχή της επόμενης εντολής. Στη μακριά μορφή μπορεί να

απομακρυνθεί τελείως.

**Δικτύωση**

Οι εντολές **IF** μπορούν να δικτυωθούν τόσο βαθιά όσο ο χρήστης απαιτεί (περιορισμένες από τη διαθέσιμη μνήμη). Παρ' όλα αυτά, μπορεί να δημιουργηθεί σύγχυση ως προς το ποιο **ELSE, END IF** κτλ. αντιστοιχεί με ποιο **IF**. Η SuperBASIC θα αντιστοιχίσει δικτυωμένες εντολές **ELSE** κτλ. στην κοντινότερη πρόταση **IF**, για παράδειγμα:

```
100 IF a = b THEN
110    IF c = d THEN
120       PRINT "error"
130    ELSE
140       PRINT "no error"
150    END IF
160 ELSE
170    PRINT "not checked"
180 END IF
```

Η **ELSE** στη σειρά 40 αντιστοιχεί στο δεύτερο **IF**. Η **ELSE** στη σειρά 70 αντιστοιχεί στο πρώτο **IF** (στη σειρά 10).

# INK (Παράθυρο)

Αυτό βάζει το τρέχον χρώμα μελάνης δηλαδή το χρώμα στο οποίο γράφονται τα αποτελέσματα. Το **INK** θα έχει επίδραση στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι.

syntax: **INK** [*channel*,] *colour*

example:
i. `INK 5`
ii. `INK 6,2`
iii. `INK #2,255`

# INKEY$

Η **INKEY$** είναι μια συνάρτηση που δίνει είσοδο ενός μοναδικού χαρακτήρα είτε από το καθορισμένο ή από το default κανάλι.

Μπορεί να καθοριστεί ένα προαιρετικό διάλειμμα που μπορεί να διαρκέσει ένα συγκεκριμένο χρόνο πριν δώσει είσοδο, μπορεί να δώσει είσοδο αμέσως ή μπορεί να περιμένει για πάντα. Αν δε δοθεί κάποια παράμετρος το **INKEY$** θα δώσει είσοδο αμέσως.

syntax: **INKEY$** [|(*channel*)
|(*channel, time*)
|(*time*)]

where: *time* = 1 .. 32767 Περιμένει ορισμένο αριθμό εικόνων
*time* = –1 Περιμένει για πάντα.
*time* = 0 Επιστρέφει αμέσως

example:
i. `PRINT INKEY$`
ii. `PRINT INKEY$(#4)`
iii. `PRINT INKEY$(50)`
iv. `PRINT INKEY$(0)`
v. `PRINT INKEY$(#3,100)`

i Είσοδος απο το default κανάλι
ii Είσοδος απο το κανάλι 4
iii Περιμένει 50 εικόνες και μετά επιστρέφει οπουδήποτε
iv Επιστρέφει αμέσως (Ρωτά το πληκτρολόγιο)
v Περιμένει 100 εικόνες για είσοδο απο το κανάλι 3
και μετά επιστρέφει οπωσδήποτε.

# INPUT

Η **INPUT** επιτρέπει την εισαγωγή δεδομένων σε ένα SuperBASIC πρόγραμμα απευθείας από το πληκτρολόγιο του QL. Η SuperBASIC σταματά το πρόγραμμα έως ότου έχει εισαχθεί το συγκεκριμένο πλήθος δεδομένων, το πρόγραμμα θα συνεχίσει μετά. Κάθε στοιχείο δεδομένων θα πρέπει να τερματίζει με το πλήκτρο **ENTER.**

Η **INPUT** θα εισάγει δεδομένα είτε από το συγκεκριμένο ή από το default κανάλι.

Αν απαιτείται εισαγωγή από ένα συγκεκριμένο κανάλι κονσόλας ο δείκτης για το συνδεδεμένο παράθυρο σε εκείνο το κανάλι θα εμφανιστεί και θα αρχίσει να αναβοσβήνει.

syntax: *separator*:= | !
| ,
| \
| ;
| TO

*prompt*:= [*channel*,] *expression separator*

INPUT [*prompt*] [*channel*] *variable* *[,*variable*]*

example: i.
```
INPUT ("Last guess "& guess & "New guess?") !
guess
```

ii.
```
INPUT "What is your guess?"; guess
```

iii.
```
100 INPUT "array size?" ! limit
110 DIM array(limit-1)
120 FOR element = 0 to limit-1
130    INPUT ("data for element" & element) !
           array(element)
140 END FOR element
150 PRINT array
```

# INSTR (Τελεστής)

Η **INSTR** είναι ένας τελεστής που καθορίζει, αν ένα δεδομένο αλφαριθμητικό μέρος περιέχεται σε μια συγκεκριμένη αλφαριθμητική. Αν η αλφαριθμητική βρεθεί τότε δίνεται η θέση του μέρους. Αν δε βρεθεί τότε το **INSTR** δίνει μηδέν.

Το μηδέν μπορεί να ερμηνευτεί σαν ψευδές, π.χ. το μέρος της αλφαριθμητικής δεν περιεχόταν στην αλφαριθμητική. Μια μη μηδενική αξία, η θέση του μέρους, μπορεί να ερμηνευθεί σαν αληθές, π.χ. το μέρος περιεχόταν μέσα στη συγκεκριμένη αλφαριθμητική.

syntax: *string__expression* **INSTR** *string expression*

example:

i.
```
PRINT "a" INSTR "cat"
```
ii.
```
PRINT "CAT" INSTR "concatenate"
```
iii.
```
PRINT "x" INSTR "eggs"
```

# INT (Μαθηματικές συναρτήσεις)

Η **INT** θα δώσει το ακέραιο μέρος της ορισμένης πραγματικής αριθμητικής έκφρασης.

syntax: **INT** (*numeric__expression*)

example:

i.
```
PRINT INT(X)
```
ii.
```
PRINT INT(3.141592654/2)
```

# KEYROW

Η συνάρτηση KEYROW ψάχνει την κατάσταση μιας ολόκληρης σειράς πλήκτρων όπως φαίνεται στον κατωτέρω πίνακα. Η KEYROW παίρνει μια παράμετρο η οποία πρέπει να είναι ένας ακέραιος μεταξύ 0 και 7. Με αυτόν τον αριθμό διαλέγεται η σειρά στην οποία βρισκόμαστε. Η τιμή που επιστρέφεται από την KEYROW είναι ένας ακέραιος μεταξύ 0 και 255, ο οποίος δίνει μια δυαδική αντιπροσώπευση για το ποια πλήκτρα έχουν πιεστεί πάντα στην υπόψη σειρά.

Καθόσον η KEYROW χρησιμοποιείται εναλλακτικά με τις κανονικές εντολές εισόδου INKEY$ και INPUT, κάθε χαρακτήρας στον καταχωρητή του πληκτρολογίου σβήνεται με τη χρήση της. 'Ετσι πληκτρολογήσεις που έγιναν πριν από την κλήση της KEYROW δε θα αναγνωριστούν από εντολές INKEY$ ή INPUT.

Να σημειωθεί ότι ταυτόχρονη πληκτρολόγηση πολλών πλήκτρων μπορεί να προκαλέσει μη αναμενόμενα αποτελέσματα. Ιδιαίτερα αν τρία πλήκτρα που βρίσκονται στη γωνία του τετραγώνου των πλήκτρων στον κατωτέρω πίνακα πιεστούν συγχρόνως, ο υπολογιστής θα νομίσει ότι πιέστηκε και το τέταρτο πλήκτρο. Τα τρία ειδικά πλήκτρα CTRL, ALT και SHIFT είναι εξαίρεση αυτού του κανόνα.

syntax: *row*:= *numeric_expression* {range 0..7}

KEYROW (*row*)

example:
```
100 REMark run this program and press a few keys
110 REPeat loop
120    CURSOR 0,0
130    FOR row = 0 to 7
140       PRINT row !!! KEYROW(row) ; " "
150    END FOR row
160 END REPeat loop
```

**ΠΙΝΑΚΑΣ ΠΛΗΚΤΡΟΛΟΓΙΟΥ**

| COLUMN / ROW | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|---|
| 7 | SHIFT | CTRL | ALT | X | V | / | N | , |
| 6 | 8 | 2 | 6 | Q | E | O | T | U |
| 5 | 9 | W | I | TAB | R | - | Y | |
| 4 | L | 3 | H | 1 | A | P | D | J |
| 3 | I | CAPS LOCK | K | S | F | = | G | ; |
| 2 | \| | Z | . | C | B | £ | M | ~ |
| 1 | ENTER | ← | up | ESC | → | \ | SPACE | down |
| 0 | F4 | F1 | 5 | F2 | F3 | F5 | 4 | 7 |

# LBYTES (Microdrives)

Η **LBYTES** θα φορτώσει ένα αρχείο δεδομένων στη μνήμη στη συγκεκριμένη διεύθυνση αρχής.

syntax: *start_address*:= *numeric_expression*

**LBYTES** *device* ,*start_address*

example: i.
```
LBYTES mdv1_screen, 131072
```
{load a screen image}

ii.
```
LBYTES mdv1_program, start_address
```
{load a program at a specified address}

# LEN (Αλφαριθμητική μεταβλητή με δείκτες)

Η **LEN** είναι μια συνάρτηση που θα επιστρέψει το μήκος της συγκεκριμένης αλφαριθμητικής έκφρασης.

syntax: LEN (*string_expression*)

example:
```
i.    PRINT LEN( "LEN will find the length of this
      string")
ii.   PRINT LEN(output_string$)
```

# LET

Η **LET** αρχίζει μια εντολή αντικατάστασης SuperBASIC. Η χρήση της δεσμευμένης λέξης **LET** είναι προαιρετική. Η αντικατάσταση μπορεί να χρησιμοποιηθεί και για αλφαριθμητικές και για αριθμητικές αντικαταστάσεις. Η SuperBASIC θα μετατρέψει αυτόματα τους ακατάλληλους τύπους δεδομένων σε μία κατάλληλη μορφή οπότε είναι δυνατό.

syntax: [LET] *variable* = *expression*

example:
```
i.    LET a = 1 + 2
ii.   LET a$ = "12345"
iii.  LET a$ = 6789
iv.   b$ = test_data
```

# LINE
# LINE_R (Γραφικά)

Η **LINE** επιτρέπει τη σχεδιάση μιας ευθείας γραμμής ανάμεσα σε δύο σημεία στο προσαρτημένο παράθυρο στο default ή συγκεκριμένο κανάλι. Οι άκρες της γραμμής ορίζονται χρησιμοποιώντας το σύστημα γραφικών συντεταγμένων.

Μπορούν να γραφτούν πολλές ευθείες γραμμές με μία μόνο εντολή **LINE**.

Ο κανονικός προσδιορισμός απαιτεί τον καθορισμό των δύο σημείων που αποτελούν τα άκρα της γραμμής, αυτά τα σημεία των άκρων μπορούν να καθοριστούν είτε με απόλυτες συντεταγμένες (σχετικές με την αρχή των γραφικών) ή με σχετικές συντεταγμένες (σχετικές με το δείκτη γραφικών). Αν το πρώτο σημείο παραλειφθεί τότε μια ευθεία γραμμή γραφεται από τη θέση του δείκτη των γραφικών μέχρι το καθορισμένο σημείο. Αν το δεύτερο σημείο παραληφθεί τότε ο δείκτης γραφικών κινείται αλλά δε γράφεται καμμία γραμμή.

Η **LINE** θα γράφει πρώτα με απόλυτες συντεταγμένες, π.χ. σχετικές με την αρχή των γραφικών, ενώ η **LINE_R** θα γράψει σχετικά με το δείκτη των γραφικών

syntax:
*x*:= *numeric_expression*
*y*:= *numeric_expression*
*point*:= *x* , *y*

parameter_2:= | **TO** *point* 1
| ,*point* **TO** *point* 2

parameter_1:= | **TO** *point, angle* 1
| **TO** *point* 2
| *point* 3

**LINE** [*channel*,] *parameter_1* *[, *parameter_2*]*

**LINE_R** [*channel,*] *parameter_1* *[,*parameter_2*]*

όπου:

- το 1 θα γράψει από το ένα καθορισμένο σημείο ως το άλλο καθορισμένο σημείο.
- το 2 θα γράψει από το τελευταίο σχεδιασμένο σημείο ως το καθορισμένο σημείο
- το 3 θα κινηθεί ως το καθορισμένο σημείο δε θα γραφτεί καμμία γραμμή

example:
i. `LINE 0,0 TO 0, 50 TO 50,0 TO 50,0 TO 0,0`
ii. `LINE TO 0.75, 0.5`
iii. `LINE 25,25`

i Ενα τετράγωνο
ii Μια γραμμή
iii Κινεί το δρομέα γραφικών.

# LIST

**Η LIST** επιτρέπει σε μια γραμμή η ομάδα από γραμμές της SuperBASIC να καταγραφούν σε ένα συγκεκριμένο ή default κανάλι.

LIST is terminated by CTRL | space

syntax:
*line*:= | *line_number* TO *line_number* 1
| *line_number* TO 2
| TO *line_number* 3
| *line_number* 4
| 5

LIST [*channel*,] *line* *[,*line*]*

όπου:

- το 1 θα καταγράψει από τη πρώτη συγκεκριμένη γραμμή ως τη δεύτερη συγκεκριμένη γραμμή

- το 2 θα καταγράψει από τη συγκεκριμένη γραμμή ως το τέλος του προγράμματος

- το 3 θα καταγράψει από την αρχή του προγράμματος ως τη συγκεκριμένη γραμμή.

- το 4 θα καταγράψει τη συγκεκριμένη γραμμή

- το 5 θα καταγράψει όλο το πρόγραμμα

example:
i. LIST
ii. LIST 10 to 300
iii. LIST 12,20,50

i Εμφανίζει όλες τις γραμμές
ii Εμφανίζει τις γραμμές 10 έως 30
iii Εμφανίζει τις γραμμές 12,20 και 50 μόνο.

Παρατήρηση: Αν το αποτέλεσμα της **LIST** κατευθύνεται σε ένα κανάλι που έχει ανοιχτεί σαν ένα κανάλι εκτυπωτή τότε το **LIST** θα προμηθεύσει ένα αντίγραφο σε χαρτί.

# LN
# LOG10 (Μαθηματικές συναρτήσεις)

Η **LN** θα επιτρέψει τον φυσικό (Νεπέριο) λογάριθμο του καθορισμένου ορίσματος. Το **LOG10** θα επιστρέψει το δεκαδικό λογάριθμο. Δεν υπάρχει πάνω όριο στην παράμετρο που εφοδιάζεται εκτός από το μεγαλύτερο αριθμό που μπορεί να δώσει ο υπολογιστής.

syntax: **LOG10** (*numeric_expression*)
**LN** (*numeric_expression*)

example:
i. `PRINT LOG10(20)`
ii. `PRINT LN(3.141592654)`

# – LOAD (Microdrives, συσκευές)

Η **LOAD** θα φορτώσει ένα SuperBASIC πρόγραμμα από οποιαδήποτε συσκευή του QL. Η **LOAD** εκτελεί αυτόματα μια **NEW** πριν φορτώσει κάποιο άλλο πρόγραμμα, και έτσι οποιαδήποτε προηγούμενα φορτωμένο πρόγραμμα θα σβηστεί από τη **LOAD.**

Αν μια γραμμή εισόδου κατά τη διάρκεια ενός φορτώματος έχει μη σωστή SuperBASIC σύνταξη, η λέξη **MISTAKE** εισάγεται ανάμεσα στον αριθμό γραμμής και το σώμα της. Κατά την εκτέλεση, μια τέτοιου είδους γραμμή θα δημιουργήσει ένα λάθος.

syntax: **LOAD** *device*

example:
i. `LOAD "mdv1_test_program"`
ii. `LOAD mdv1_games`
iii. `LOAD neti_3`
iv. `LOAD ser1_e`

# LOCal (Συναρτήσεις και διαδικασίες)

Η **LOCAL** επιτρέπει αναγνωριστές να καθοριστούν να είναι **LOCal** σε μια συνάρτηση ή διαδικασία. Οι τοπικοί αναγνωριστές υπάρχουν μόνο μέσα στη συνάρτηση ή διαδικασία στην οποία καθορίζονται, ή σε διαδικασίες και συναρτήσεις που καλούνται από τη συνάρτηση ή διαδικασία μέσα στην οποία καθορίστηκαν. 'Οταν η συνάρτηση ή η διαδικασία τερματίζεται χάνονται. Οι τοπικοί αναγνωριστές είναι ανεξάρτητοι από παρόμοιους ονομασμένους αναγνωριστές έξω από τη συνάρτηση ή διαδικασία που τους καθορίζει. Μεταβλητές με δείκτες μπορούν να οριστούν να γίνουν τοπικές δίνοντας τους διαστάσεις μέσα στην εντολή **LOCal.**

Οι εντολές **LOCal** θα πρέπει να προηγούνται από την πρώτη εκτελέσιμη εντολή στη συνάρτηση ή τη διαδικασία στην οποία χρησιμοποιούνται:

syntax: **LOCal** *identifier* *[, *identifier*]*

example:
i. `LOCal a, b, c(10,10)`
ii. `LOCal temp_data`

Παρατήρηση: Ο καθορισμός μεταβλητών να είναι **LOCAL** επιτρέπει εύχρηστα ονόματα μεταβλητών να χρησιμοποιούνται μέσα σε συναρτήσεις και διαδικασίες χωρίς να κινδυνεύετε να καταστρέψετε παρόμοιες μεταβλητές έξω από τη συνάρτηση ή τη διαδικασία.

# LRUN (Συσκευές, Microdrives)

Η **LRUN** θα φορτώσει το πρόγραμμα και θα τρέξει ένα SuperBASIC πρόγραμμα από μια συγκεκριμένη συσκευή. Η **LRUN** θα εκτελέσει **NEW** πριν φορτώσει ένα άλλο και οποιοδήποτε προηγούμενα φυλαγμένο SuperBASIC πρόγραμμα θα σβηστεί από τη **LRUN.**

Αν μια γραμμή εισόδου κατά τη διάρκεια του φορτώματος έχει λανθασμένη σύνταξη της SuperBASIC, η λέξη **MISTAKE** εισάγεται ανάμεσα στον αριθμό γραμμής και το κυρίως σήμα της. Κατά την εκτέλεση, μια τέτοιου είδους γραμμή θα δώσει ένα λάθος.

syntax: LRUN *device*

example:
i. `LRUN mdv2_TEST`
ii. `LRUN mdv1_game`

# MERGE (Συσκευές, Microdrives)

Η **MERGE** θα φορτώσει ένα αρχείο από τη συγκεκριμένη συσκευή και θα το ερμηνεύσει σαν ένα SuperBASIC πρόγραμμα. Αν το νέο αρχείο περιέχει έναν αριθμό γραμμής που δεν εμφανίζεται στο πρόγραμμα που είναι ήδη στο QL τότε η γραμμή αυτή θα προστεθεί στο πρόγραμμα. Αν το νέο αρχείο περιέχει μια αντικαταστάτρια γραμμή για μια που ήδη υπάρχει τότε η γραμμή αυτή θα αντικατασταθεί. 'Ολες οι άλλες σειρές του παλιού προγράμματος μένουν απείρακτες.

Αν μια γραμμή εισόδου κατά τη διάρκεια του **MERGE** έχει μη σωστή SuperBASIC σύνταξη, η λέξη **MISTAKE** εισάγεται ανάμεσα στον αριθμό της γραμμής και το κυρίως σώμα της. Κατά την εκτέλεση, μιας τέτοιου είδους γραμμής θα δημιουργηθεί ένα λάθος.

syntax: MERGE *device*

example:
i. `MERGE mdv1_overlay_program`
ii. `MERGE mdv1_new_data`

# MOD (Τελεστής)

Η **MOD** είναι ένας τελεστής που δίνει το modulus ή το υπόλοιπο, όταν διαιρείται ένας ακέραιος με έναν άλλο.

syntax: *numeric_expression* MOD *numeric_expression*

example:
i. `PRINT 5 MOD 2` {will print 1}
ii. `PRINT 5 MOD 3` {will print 2}

# MODE (Οθόνη)

Η **MODE** θέτει τη διακριτικότητα της οθόνης και τον αριθμό των χρωμάτων που μπορεί να εμφανίσει. Το **MODE** θα καθαρίσει όλα τα παράθυρα που είναι τώρα στην οθόνη, αλλά θα διατηρήσει τη θέση τους και το σχήμα τους. Αλλάζοντας σε μορφή χαμηλής διακριτικότητας (8 χρώματα) θα θέσει το μικρότερο μέγεθος χαρακτήρα σε 2,0.

syntax: **MODE** *numeric_expression*

όπου 8 ή 256 θα επιλέξει χαμηλή διακριτικότητα

όπου 4 ή 512 θα επιλέξει υψηλή διακριτικότητα

example:
i. `MODE 256`
ii. `MODE 4`

# MOVE (Γραφικά χελώνας)

Η **MOVE** θα κινήσει τη χελώνα γραφικών στο προσαρτημένο παράθυρο στο default ή συγκεκριμένο κανάλι μια συγκεκριμένη απόσταση στην τωρινή κατεύθυνση. Η κατεύθυνση μπορεί να καθοριστεί χρησιμοποιώντας την **TURN** και την **TURN TO** εντολή. Ο συντελεστής της κλίμακας γραφικών χρησιμοποιείται στον καθορισμό πόσο μακριά κινείται η χελώνα. Κάνοντας συγκεκριμένη μια αρνητική απόσταση θα κινήσει τη χελώνα προς τα πίσω.

Η χελώνα κινείται στο προσαρτημένο παράθυρο στο καθορισμένο ή default κανάλι.

syntax: *distance*:= *numeric_expression*

**MOVE** [*channel*,] *distance*

example:
i. `MOVE #2,20`
ii. `MOVE -50`

i Κινεί τη χελώνα στο κανάλι 2, 20 μονάδες μπροστά
ii Κινεί τη χελώνα στο default κανάλι, 50 μονάδες μπροστά

# MRUN (Συσκευές, Microdrives)

Η **MRUN** θα ερμηνεύσει ένα αρχείο σαν ένα SuperBASIC πρόγραμμα και θα το συγχωνεύσει με το τελευταία φορτωμένο πρόγραμμα.

Αν χρησιμοποιηθεί σαν απ' ευθείας εντολή η **MRUN** θα τρέξει το νέο πρόγραμμα από την αρχή. Αν χρησιμοποιηθεί σαν μια εντολή προγράμματος η **MRUN** θα συνεχίσει την επεξεργασία από τη γραμμή που ακολουθεί τη **MRUN**.

Αν μια γραμμή εισόδου κατά την εισαγωγή έχει μη σωστή σύνταξη SuperBASIC, η λέξη **MISTAKE** εισάγεται ανάμεσα στον αριθμό της γραμμής και το κυρίως σώμα της. Κατά την εκτέλεση, μία γραμμή τέτοιου είδους θα δημιουργηθεί ένα λάθος.

syntax: MRUN *device*

example:
i. `MRUN mdv1_chain_program`
ii. `MRUN mdv1_new_data`

# NET (network)

Η **NET** επιτρέπει τον ορισμό του σταθμού του δικτύου. Αν ο αριθμός του σταθμού δεν έχει οριστεί σαφώς τότε το QL υποθέτει αριθμό σταθμού τον 1.

syntax: *station*:= *numeric_expression* {range 1..127}

NET *station*

example:
i. **NET 63**
ii. **NET 1**

Παρατήρηση: Μπορεί να δημιουργηθεί σύγχυση αν περισσότεροι από ένας σταθμοί έχουν τον ίδιο αριθμό σταθμού.

# NEW

Η **NEW** θα καθαρίσει το παλιό πρόγραμμα, τις μεταβλητές και τα κανάλια εκτός του 0,1 και 2.

syntax: **NEW**

example: **NEW**

# NEXT (Επανάληψη)

Η **NEXT** χρησιμοποιείται για να τερματίσει ή για να δημιουργήσει το τέλος της ανακύκλωσης σε **REPeat** και **FOR** ανακυκλώσεις (βρόχους).

Ο αναγνωριστής του NEXT πρέπει να ταιριάζει με αυτόν που ελέγχει την ανακύκλωση.

syntax: NEXT *identifier*

example:

i.
```
10 REMark this loop must repeat forever
11 REPeat infinite_loop
12    PRINT "still looping"
13 NEXT infinite_loop
```

ii.
```
10 REMark this loop will repeat 20 times
11 LET limit = 20
12 FOR index=1 TO limit
13    PRINT index
14 NEXT index
```

iii.
```
10 REMark this loop will tell you when a 30 is found
11 REPeat loop
12    LET number = RND(1 TO 100)
13    IF number <> 30 THEN NEXT loop
14    PRINT number; " is 30"
15 EXIT LOOP
16 END REPeat loop
```

**στη REPeat**

Αν η **NEXT** χρησιμοποιείται μέσα στη δομή **REPeat - END REPeat** θα αναγκάσει την επεξεργασία να συνεχίσει στην εντολή που ακολουθεί την αντίστοιχη εντολή **REPeat.**

**στη FOR**

Η εντολή **NEXT** μπορεί να χρησιμοποιηθεί για να επαναλάβει την ανακύκλωση **FOR** με τη μεταβλητή ελέγχου τοποθετημένη στην επόμενη του τιμή. Αν η ανακύκλωση **FOR** έχει εξαντληθεί τότε η επεξεργασία θα συνεχιστεί στην εντολή που ακολουθεί τη **NEXT,** αλλιώς η επεξεργασία θα συνεχιστεί στην πρόταση μετά τη **FOR.**

# ON...GOTO
# ON...GOSUB

Για να προσφέρει συμβιβαστότητα με άλλες BASIC, η SuperBASIC υποστηρίζει τις εντολές ON GOTO και ON GOSUB. Αυτές οι εντολές επιτρέπουν σε μία μεταβλητή να επιλέξει από έναν κατάλογο από πιθανούς αριθμούς γραμμών μια γραμμή για επεξεργασία σε μια GOTO η GOSUB εντολή. Αν έχουν καθοριστεί πολύ λίγοι αριθμοί γραμμών στον κατάλογο τότε γεννιέται ένα λάθος.

syntax: ON *variable* GOTO *expression* *|, *expression*|*
ON *variable* GOSUB *expression* *|, *expression*|*

example:
i. `ON x GOTO 10, 20, 30, 40`
ii. `ON select_variable GOSUB 1000, 2000, 3000, 4000`

Παρατήρηση: Μπορεί να χρησιμοποιηθεί η SELECT για να αντικαταστήσει αυτές τις δύο εντολές BASIC.

# OPEN
# OPEN_IN
# OPEN_NEW (συσκευές, Microdrives)

Η OPEN επιτρέπει στο χρήστη να συνδέσει ένα λογικό κανάλι σε μια φυσική συσκευή QL για σκοπούς εισόδου/εξόδου.

Αν το κανάλι είναι προς μια Μονάδα Μικροκασέτας τότε το αρχείο της Μονάδας Μικροκασέτας μπορεί να είναι ένα υπαρκτό αρχείο ή ένα καινούριο αρχείο. Στην οποία περίπτωση το OPEN_IN θα ανοίξει ένα ήδη υπαρκτό αρχείο Μονάδας Μικροκασέτας για είσοδο και το OPEN_NEW θα δημιουργήσει ένα νέο αρχείο Μονάδας Μικροκασέτας για έξοδο.

syntax: *channel*:= *#numeric_expression*

OPEN *channel*, *device*

example:
i. `OPEN #5, f_name$`
ii. `OPEN_IN #9, "mdv1_file_name"`

ii Ανοίγει το αρχείο mdv1_file_name

iii. `OPEN_NEW #7, mdv1_data_file`

iii Ανοίγει το αρχείο mdv1_file_name

iv. `OPEN #6, con_10x20a20x20_32`

iv Ανοίγει το κανάλι 6 για την κονσόλα δημιουργεί ενα παράθυρο μεγέθους 10 X 20 pixels στη θέση 20,20 με αποθήκη 32 bytes για το πληκτρολόγιο

v. `OPEN #8, mdv1_read_write_file.`

# OVER (Παράθυρα)

Η OVER επιλέγει τη μορφή που απαιτείται για τη διπλή εκτύπωση στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι. Η επιλεγμένη μορφή παραμένει σε λειτουργία ως την επόμενη χρήση του OVER.

syntax: *switch*:= *numeric_expression* {range −1..1}
OVER {*channel*,} *switch*

όπου

switch = 0 τυπώνει μελάνη στη λωρίδα

switch = 1 τυπώνει μελάνη στη διαφανή λωρίδα

switch = -1 κάνει XOR στα δεδομένα της οθόνης

example: i. OVER 1 {set "overprinting"}
ii.
```
10 REMark Shadow Writing
11 PAPER 7 : INK 0 : OVER 1 : CLS
12 CSIZE 3,1
13 FOR i = 0 TO 10
14    CURSOR i,i
15    IF i=10 THEN INK 2
16    PRINT "Shadow"
17 END FOR i
```

# PAN (Παράθυρα)

Καθαρίζει ολόκληρο το τωρινό παράθυρο το συγκεκριμένο αριθμό στιγμάτων στα αριστερά ή στα δεξιά. Το PAPER ξετυλίγεται για να γεμίσει την καθαρή περιοχή.

Μια προαιρετική δεύτερη παράμετρος μπορεί να καθοριστεί που θα επιτρέψει μόνο σε ενα μέρος της οθόνης να καθαριστεί.

syntax: *distance*:= *numeric__expression*
*part*:= *numeric__expression*

**PAN** [*channel*,] *distance* [, *part*]

όπου:

μέρος = 0 - όλη η οθόνη (ή καμμία παράμετρος)

μέρος = 3 - ολόκληρη η σειρά του δείκτη

μέρος = 4 - το δεξιό άκρο της σειράς του δείκτη συμπεριλαμβανόμενης και της θέσης του δείκτη.

Αν η έκφραση υπολογιζόμενη δίνει θετική τιμή τότε το περιεχόμενο της οθόνης μετατοπίζεται προς τα δεξιά.

example:
i. **PAN #2,50**
ii. **PAN -100**
iii. **PAN 50,3**

i Μετακινεί αριστερά 50 pixels
ii Μετακινεί δεξιά 1000 pixels
iii Μετακινεί ολόκληρη τη γραμμή του δρομέα 50 pixels δεξιά

Προειδοποίηση: Αν χρησιμοποιείται διαστίξεις ή η οθόνη είναι σε μορφή χαμηλής διακριτικότητας για να κρατηθεί το σχέδιο διάστιξης, η οθόνη πρέπει να καθοριστεί σε πολλαπλάσια των 2 στιγμάτων.

# PAPER (Παράθυρα)

Η PAPER θέτει ένα καινούργιο χρώμα χαρτιού (δηλαδή το χρώμα που θα χρησιμοποιηθεί από το CLS, το PAN, το SCROLL κτλ.). Το επιλεγμένο χρώμα χαρτιού παραμένει σε ενέργεια ως την επόμενη χρήση του PAPER. Η PAPER θα θέσει επίσης και το STRIP χρώμα.

Η PAPER θα αλλάξει το χρώμα χαρτιού στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι.

syntax: **PAPER** [*channel*,] *colour*

example:

i.
```
PAPER #3,7
```
'Ασπρο χαρτί στο κανάλι 3

ii.
```
PAPER 7,2
```
'Ασπρη και κόκκινη λωρίδα

iii.
```
PAPER 255
```
Μαύρη και άσπρη λωρίδα

iv.
```
10 REMark Show colours and stipples
11 FOR colour = 0 TO 7
12    FOR contrast = 0 TO 7
13       FOR stipple = 0 TO 3
14          PAPER colour, contrast, stipple
15          SCROLL 6
16       END FOR stipple
17    END FOR contrast
18 END FOR colour
```

# PAUSE

Η PAUSE θα κάνει ένα πρόγραμμα να περιμένει για μια συγκεκριμένη περίοδο χρόνου. Οι καθυστερήσεις συγκεκριμενοποιούνται σε μονάδες των 20 ms στη Μεγάλη Βρεττανία και άλλες των 16.67 ms. Η είσοδος από πληκτρολόγιο θα τερματίσει το PAUSE και θα ξαναρχίσει την εκτέλεση του προγράμματος. Αν δεν καθοριστεί καμμία καθυστέρηση τότε το πρόγραμμα θα σταματήσει επ' άπειρον.

syntax: *delay*:= *numeric_expression*

**PAUSE** [*delay*]

example:

i. **PAUSE 50** Περιμένει 1 δευτερόλεπτο

ii. **PAUSE 500** Περιμένει 10 δευτερόλεπτα

# PEEK
# PEEK_W
# PEEK_L (BASIC)

Η PEEK είναι μία συνάρτηση που δίνει το περιεχόμενο της συγκεκριμένης θέσης της μνήμης.

Η PEEK έχει τρεις μορφές που η καθεμία ανάλογα δίνει ένα byte (8 bits), μία λέξη (16 bits), μία μακριά λέξη (32 bits).

syntax: *address:= numeric_expression*

PEEK(*address*)
PEEK_W(*address*)
PEEK_L(*address*)

example:
i. PRINT PEEK(12245)
ii. PRINT PEEK_W(12)
iii. PRINT PEEK_L(1000)

i Τα περιεχόμενα της θέσης 12245
ii Τα περιεχόμενα της λέξης 12 και 13 bytes
iii Τα περιεχόμενα 4 bytes απο τη θέση 1000

Προειδοποίηση: Για προσπέλαση λέξεων και μακριών λέξεων η συγκεκριμένη διεύθυνση θα πρέπει να είναι μία άρτια διεύθυνση.

# PENUP
# PENDOWN (Γραφικά χελώνας)

Λειτουργεί την "πένα" στα γραφικά χελώνας. Αν η πένα είναι πάνω τότε δε θα γραφτεί τίποτα. Αν η πένα είναι κάτω τότε θα γραφτούν γραμμές καθώς η χελώνα κινείται κατά μήκος της οθόνης.

Η γραμμή θα γραφτεί στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι. Η γραμμή θα γραφεί στο τωρινό χρώμα μελάνης για το κανάλι στο οποίο κατευθύνεται το αποτέλεσμα.

syntax: **PENUP** [*channel*]
**PENDOWN** [*channel*]

example:
i. **PENUP**
ii. **PENDOWN #2**

i Σηκώνει την πένα στο default κανάλι
ii Χαμηλώνει την πένα στο παράθυρο του καναλιού 2

# PI (Μαθηματικές συναρτήσεις)

Η PI είναι η συνάρτηση που δίνει την τιμή του π.

syntax: PI

example: PRINT PI

# POINT

# POINT_R (γραφικά)

Η POINT σχεδιάζει ένα σημείο στην καθορισμένη θέση στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι. Το σημείο σχεδιάζεται χρησιμοποιώντας το σύστημα γραφικών συντεταγμένων σχετικό με την αρχή των γραφικών. Αν χρησιμοποιηθεί το POINT_R, τότε όλα τα σημεία καθορίζονται σχετικά με το δείκτη γραφικών και σχεδιάζονται σχετικά το ένα με το άλλο.

Πολλαπλά σημεία μπορούν να σχεδιαστούν με ένα απλό κάλεσμα του POINT.

syntax:

*x*:= *numeric_expression*
*y*:= *numeric_expression*

*parameters*:= *x* , *y*

**POINT** [*channel*,] *parameters* *[,*parameters*]*

example:

i.
```
POINT 256,128
```
ii.
```
POINT x, x*x
```
iii.
```
10 REPeat example
20    INK RND(255)
30    POINT RND(100),RND(100)
40 END REPeat example
```

i Σχεδιάζει ενα σημείο στη θέση (256,128)
ii Σχεδιάζει ενα σημείο στη θέση (X,X*X)

# POKE
# POKE_W
# POKE_L (BASIC)

Η **POKE** επιτρέπει μια θέση μνήμης να αλλαχθεί. Για την προσπέλαση λέξεων και μακριών λέξεων, η καθορισμένη διεύθυνση θα πρέπει να είναι μία άρτια διεύθυνση.

Η **POKE** έχει τρεις μορφές που η κάθε μία αντίστοιχα προσπελαύνει ένα byte (8 bits), μία λέξη (16 bits) μια μακριά λέξη (32 bits).

syntax:

*address*:= *numeric_expression*
*data*:= *numeric_expression*

POKE *address*, *data*
POKE_W *address*, *data*
POKE_L *address*, *data*

example:

i. `POKE 12235,0`
ii. `POKE_L 131072, 12345`

i Βάζει στο byte 12235 το μηδέν
ii Βάζει στη μακριά λέξη 131072 το 12235

Προειδοποίηση: Η προσπέλαση δεδομένων από περιοχές μνήμης που χρησιμοποιούνται από το Qdos μπορεί να προκαλέσουν "σπάσιμο" του συστήματος και μπορούν να χαθούν δεδομένα. Η προσπέλαση σε τέτοιες περιοχές δε συνιστάται.

# PRINT (Συσκευές, Microdrives)

Επιτρέπει σε εξαγόμενα να σταλθούν στο συγκεκριμένο ή default κανάλι. Η κανονική χρήση του **PRINT** είναι να στείλει δεδομένα στην οθόνη του QL.

syntax: *separator*:= | !
| ,
| \
| ;
| TO numeric__expression

*item*:= | *expression*
| *channel*
| *separator*

PRINT *[*item*]*

Επιτρέπονται πολλαπλοί διαχωριστές εκτύπωσης. Τουλάχιστον, ένας διαχωριστής θα πρέπει να χωρίζει καθορισμούς καναλιών και εκφράσεις.

example: i. `PRINT "Hello World"`

i θα εμφανίσει Hello World στο defaut κανάλι εξόδου (κανάλι 1)

ii. `PRINT #5, "data", 1,2,3,4`

ii θα εμφανίσει τα δεδομένα στο κανάλι 5 (πρέπει να έχει ανοίξει προηγουμένως)

iii. `PRINT TO 20 ; "This is in column 20"`

**Διαχωριστές**

! Καλύτερα να βλέπεται σαν ένα ευφυές διάστημα. Η κανονική δράση του είναι να εισάγει ένα διάστημα ανάμεσα στα στοιχεία εξαγομένων στην οθόνη. Αν το στοιχείο δε θα χωρέσει στη τρέχουσα γραμμή θα ενεργοποιηθεί μία τροφοδοσία σειράς. Αν η τρέχουσα θέση εκτύπωσης είναι στην αρχή μιας γραμμής τότε δε θα εξαχθεί ένα διάστημα. Το ! επηρεάζει το επόμενο στοιχείο που θα εκτυπωθεί και γι' αυτό θα πρέπει να τοποθετηθεί μπροστά από το προς εκτύπωση στοιχείο. Επίσης ένα ; ή ένα ! θα πρέπει να τοποθετούνται, στο τέλος μιας λίστας εκτύπωσης αν τα ίδια διαστήματα πρόκειται να συνεχιστούν για μια σειρά από εντολές **PRINT**.

, Κανονικός διαχωριστής, η SuperBASIC θα πινακοποιεί τα εξαγόμενα κάθε 8 στήλες.

/ θα επιβάλλει μια νέα σειρά.

; θα αφήσει τη θέση εκτύπωσης αμέσως μετά την εκτύπωση του τελευταίου στοιχείου, το εξαγόμενο θα εκτυπωθεί σε μια συνεχόμενη ροή.

**TO** θα εκτελέσει μια λειτουργία διαστημάτων. Το **TO** ακολουθούμενο από μία αριθμητική_έκφραση θα προχωρήσει τη θέση εκτύπωσης στη στήλη που καθορίζεται από την αριθμητική_εκφραση. Αν η απαιτούμενη στήλη είναι μη λογική ή η τρέχουσα θέση εκτύπωσης είναι πέρα από τη καθορισμένη θέση τότε δε θα συμβεί καμμία ενέργεια.

# RAD (Μαθηματική συνάρτηση)

Η **RAD** είναι μια συνάρτηση που θα μετατρέψει μια συγκεκριμένη γωνία σε μοίρες σε μια συγκεκριμένη γωνία σε ακτίνια.

syntax: RAD (*numeric_expression*)

example: **PRINT RAD(180)** {will print 3.141593}

# RANDOMISE (Μαθηματική συνάρτηση)

Η **RANDOMISE** επιτρέπει τη ανατροφοδότηση της γενήτριας των τυχαίων αριθμών. Αν καθοριστεί μία παράμετρος τότε αυτή παίρνεται σαν να είναι η νέα αρχή. Αν δεν καθοριστεί παράμετρος τότε η γενήτρια ανατροφοδοτείται από εσωτερικές πληροφορίες.

syntax: **RANDOMISE** [*numeric_expression*]

example:

i. **RANDOMISE**
ii. **RANDOMISE 3.2235**

i θέτει τη γεννήτρια με εσωτερικά δεδομένα
ii θέτει τη γεννήτρια σε 3.2235

# RECOL (Παράθυρο)

Η **RECOL** θα ξαναχρωματίσει ανεξάρτητα pixels στο προσαρτημένο παράθυρο στο συγκεκριμένο ή το default κανάλι σύμφωνα με κάποιο προκαθορισμένο σχέδιο. Κάθε παράμετρος υποτίθεται ότι καθορίζει, στη σειρά, το χρώμα στο οποίο κάθε pixel επαναχρωματίζεται, π.χ. η πρώτη παράμετρος καθορίζει το χρώμα με το οποίο θα ξαναχρωματιστούν όλα τα μαύρα pixels, η δεύτερη παράμετρος μπλε pixels κ.τ.λ.

Ο καθορισμός χρώματος θα πρέπει να είναι ένα κύριο χρώμα, π.χ. θα πρέπει να είναι μέσα στην κλίμακα από το 0 ως το 7.

syntax:

```
C0:=νέο χρώμα για μαύρο
C1:= "     "     "  μπλέ
C2:= "     "     "  κόκκινο
C3:= "     "     "  μώβ
C4:= "     "     "  πράσινο
C5:= "     "     "  κυανό
C6:= "     "     "  κίτρινο
C7:= "     "     "  άσπρο
```

**RECOL** [*channel* ,] *c0*, *c1*, *c2*, *c3*, *c4*, *c5*, *c6*, *c7*

example:

```
RECOL 2,3,4,5,6,7,1,0
```

Δίνει νέα χρώματα μπλέ για το μώβ κόκκινο για το πράσινο, μώβ για το κυανό κ.τ.λ.

# REMark (Παρατήρηση)

Η **REMark** επιτρέπει την είσοδο επεξηγηματικού κειμένου μέσα σε ένα πρόγραμμα. Το υπόλοιπο της γραμμής αγνοείται από τη **SuperBASIC.**

syntax: REMark *text*

example:

```
REMark This is a comment in a program
```

Η REMARK χρησιμοποιείται για να προσθέτει παρατηρήσεις σε ένα πρόγραμμα που θα βοηθήσουν να γίνει πιο αντιληπτό.

Προειδοποίηση: Η **REMark** θα πρέπει να χρησιμοποιείται για να προσθέτει διευκρινιστικές παρατηρήσεις σε ένα πρόγραμμα οπουδήποτε θα βοηθούσε.

# RENUM

Η **RENUM** επιτρέπει σε μία ομάδα ή μία σειρά ομάδων αριθμών γραμμών SuperBASIC να αλλαχθεί. Αν δεν καθοριστούν οι παράμετροι τότε το **RENUM** θα ξαναριθμήσει το πρόγραμμα από την αρχή ως το τέλος αρχίζοντας από τη γραμμή 100 και αριθμώντας με διαστήματα των 10. Αν η αρχική γραμμή καθοριστεί τότε αριθμοί γραμμών πριν από αυτή δε θα αλλάξουν. Αν η τελική γραμμή καθοριστεί τότε οι αριθμοί γραμμών μετά από αυτή δε θα αλλάξουν. Αν η αρχική και η τελική γραμμή καθοριστεί τότε οι γραμμές που θα επαναριθμηθούν θα αρχίζουν από την αρχική γραμμή και θα αυξάνονται με το ορισμένο βήμα.

Αν μια εντολή **GOTO** ή **GOSUB** περιέχει μία έκφραση που να αρχίζει με έναν αριθμό τότε αυτός ο αριθμός χειρίζεται σαν ένας αριθμός γραμμής και ξαναριθμείται.

syntax:

*start_line*:= *numeric_expression* {start renumber}
*end_line*:= *numeric_expression* {stop renumber}
*start_number*:= *numeric_expression* {base line number}
*step*:= *numeric_expression* {step}

RENUM [*start_line* [TO *end_line*];] [*start_number*] [,*step*]

example:

i. `RENUM`
ii. `RENUM 100 TO 200`

i Επαναριθμεί όλες τις εντολές του προγράμματος αρχίζοντας από 100 με βήμα 10
ii Επαναριθμεί απο το 100 μέχρι το 200 με βήμα 10

Προειδοποίηση: Δεν πρέπει να γίνει καμμία προσπάθεια για να χρησιμοποιήσετε την **RENUM** για να επαναριθμήσετε μη συνεχόμενες σειρές. Π.χ. να κινήσετε σειρές μέσα στο πρόγραμμα. Η **RENUM** δε θα πρέπει να χρησιμοποιείται μέσα σε ένα πρόγραμμα.

# REPeat
# END REPeat (Επανάληψη)

Η **REPeat** επιτρέπει τη δημιουργία γενικών επαναλαμβανόμενων βρόχων. Το **REPeat** θα πρέπει να χρησιμοποιείται με το **EXIT** για μέγιστο αποτέλεσμα. Το **REPeat** μπορεί να χρησιμοποιηθεί και σε μακριά και κοντή μορφή:

**Σύντομη μορφή**

Η δεσμευμένη λέξη **REPeat** και ο αναγνωριστής βρόχου ακολουθούνται στην ιδια λογική γραμμή από μια διπλή τελεία (:) και από μία ακολουθία εντολών της SuperBASIC. Η **EXIT** θα συνεχίσει την κανονική επεξεργασία στην επόμενη λογική γραμμή.

syntax: REPeat *identifier* : *statements*

example:
```
REPeat wait : IF INKEY$ <> "" THEN EXIT wait
```

**Μακριά μορφή**

Η δεσμευμένη λέξη **REPeat** και ο αναγνωριστής του βρόχου είναι οι μόνες εντολές στη λογική γραμμή. Οι επόμενες γραμμές περιέχουν μια σειρά από SuperBASIC εντολές που τελειώνουν με μία εντολή **END REPeat.**

Οι εντολές ανάμεσα στη **REPeat** και τη **END REPeat** εκτελούνται επαναληπτικά από τη SuperBASIC.

syntax:
```
REPeat identifier
  statements
END REPeat identifier
```

example:
```
10 LET number = RND(1 TO 50)
11 REPeat guess
12    INPUT "What is your guess?", guess
13    IF guess = number THEN
14       PRINT "You have guessed correctly"
15       EXIT guess
16    ELSE
17       PRINT "You have guessed incorrectly"
18    END IF
19 END REPeat guess
```

Παρατήρηση: Κανονικά, τουλάχιστον μία εντολή στο βρόχο **REPEAT** θα είναι μία εντολή **EXIT.**

# RESPR (Qdos)

Η **RESPR** είναι μια συνάρτηση που θα διατηρήσει λίγο από το χώρο της διαδικασίας που υπάρχει στη μνήμη για παράδειγμα για να μεγαλώσει τη λίστα διαδικασιών της SuperBASIC.

syntax: *space*:= *numeric__expression*

RESPR (*space*)

example: **PRINT RESPR(1024)**

θα τυπώσει τη διεύθυνση βάσης ενος μπλοκ 1024 bytes

# RETurn (Συναρτήσεις και διαδικασίες)

Η **RETurn** χρησιμοποιείται για να εξαναγκάσει μία συνάρτηση ή διαδικασία να τερματίσει και να επαναρχίσει την επεξεργασία από την εντολή μετά τη διαδικασία ή το κάλεσμα συνάρτησης. 'Οταν χρησιμοποιείται σε μία συνάρτηση η εντολή **RETurn** χρησιμοποιείται για να επιστρέψει την τιμή της συνάρτησης.

syntax: RETurn |*expression*|

example: i.

```
100 PRINT ack (3,3)
110 DEFine FuNction ack(m,n)
120    IF m=0 THEN RETurn n+1
130    IF n=0 THEN RETurn ack (m-1,1)
140    RETurn ack(m-1,ack(m,n-1))
150 END DEFine
```

ii.

```
10 LET warning_flag = 1
11 LET error_number = RND(0 TO 10)
12 warning error_number
13 DEFine PROCedure warning(n)
14   IF warning_flag THEN
15     PRINT "WARNING:";
16     SELect ON n
17       ON n = 1
18         PRINT "Microdrive full"
19       ON n = 2
20         PRINT "Data space full"
21       ON n = REMAINDER
22         PRINT "Program error"
23     END SELect
24   ELSE
25     RETurn
26   END IF
27 END DEFine
```

Δεν είναι υποχρεωτικό να έχετε ένα **RETurn** μέσα σε μία διαδικασία. Αν η επεξεργασία φτάσει το **END DEFine** μιας διαδικασίας, τότε η διαδικασία θα επιστρέψει αυτόματα.

Το RETurn μόνο του χρησιμοποιείται με μία GOSUB.

# RND (Μαθηματική συνάρτηση)

Η **RND** δημιουργεί ένα τυχαίο αριθμό. Μπορούν να καθοριστούν μέχρι δύο παράμετροι για τη **RND**. Αν δεν καθοριστούν καθόλου παράμετροι τότε η **RND** επιστρέφει ένα ψευτοτυχαίο πραγματικό αριθμό στην αποκλειστική κλίμακα 0 ως 1. Αν καθοριστεί μία μοναδική παράμετρος τότε η **RND** επιστρέφει έναν ακέραιο αριθμό στην κλίμακα 0 ως τη συγκεκριμένη παράμετρο. Αν καθοριστούν δύο παράμετροι τότε η **RND** επιστρέφει έναν ακέραιο στην κλίμακα που καθορίζεται από τις δύο παραμέτρους.

syntax: RND( |*numeric_expression*| |TO *numeric_expression*|)

example:

```
i.   PRINT RND

ii.  PRINT RND(10 TO 20)
iii. PRINT RND(1 TO 6)
iv.  PRINT RND(10)
```

i Αριθμός κινητής υποδιαστολής μεταξύ 0 και 1
ii Ακέραιος μεταξύ 10 και 20
iii Ακέραιος μεταξύ 1 και 6
iv Ακέραιος μεταξύ 0 και 10

# RUN (Πρόγραμμα)

Η **RUN** επιτρέπει να αρχίσει ένα SuperBASIC πρόγραμμα. Αν ένας αριθμός γραμμής καθορίζεται μέσα σε μια εντολή **RUN** τότε το πρόγραμμα θα αρχίσει από εκείνο το σημείο, αλλιώς το πρόγραμμα θα αρχίσει από το μικρότερο αριθμό γραμμής.

syntax: RUN [*numeric_expression*]

example:

```
i.   RUN        i Τρέχει απο την αρχή
ii.  RUN 10     ii Τρέχει απο τη γραμμή 10
iii. RUN 2*20   iii Τρέχει απο τη γραμμή 40
```

Παρατήρηση: Παρ' όλο που η **RUN** μπορεί να χρησιμοποιηθει μεσα σε ένα πρόγραμμα, η κανονική χρήση του είναι να αρχίσει την εκτέλεση του προγράμματος πληκτρολογώντας τη σαν μια άμεση εντολή.

# SAVE (Συσκευές, Microdrives)

Η **SAVE** θα φυλάξει ένα SuperBASIC πρόγραμμα σε οποιαδήποτε συσκευή QL.

syntax: *line*:= | *numeric_expression* TO *numeric_expression* 1
| *numeric_expression* TO 2
| TO *numeric_expression* 3
| *numeric_expression* 4
| 5

i Φυλάσει τις γραμμές απο την 20 μέχρι την 70
ii Φυλάσει τις γραμμές 10,20,40
iii Φυλάσει ολόκληρο το πρόγραμμα στο δίκτυο
iv Φυλάσει ολόκληρο το πρόγραμμα στο κανάλι 1

SAVE *device* *[,*line*]*

όπου

- το 1 θα φυλάξει από τη συγκεκριμένη σειρά ως τη συγκεκριμένη σειρά
- το 2 θα φυλάξει από τη συγκεκριμένη σειρά ως το τέλος
- το 3 θα φυλάξει από την αρχή ως τη συγκεκριμένη σειρά
- το 4 θα φυλάξει τη συγκεκριμένη σειρά
- το 5 θα φυλάξει όλο το αρχείο

# SBYTES (Συσκευές, Microdrives)

Η **SBYTES** επιτρέπει σε περιοχές της μνήμης του QL να φυλαχθούν σε μία συσκευή του QL.

syntax: *start_address*:= *numeric_expression*
*length*:= *numeric_expression*

SBYTES *device*, *start_address*, *length*

example: i. SBYTES mdv1_screen_data, 131072,32768
i Φυλάσει απο τη θέση 131072,32768 bytes
ii. SBYTES mdv1_test_program, 50000,10000
ii Φυλάσει απο τη θέση 50000,1000 bytes
iii. SBYTES neto_3, 32768,32678
iii Φυλάσει απο τη θέση 32768,32678 bytes στο δίκτυο
IV. SBYTES ser1, 0,32768
iv Φυλάσει απο τη θέση 0,32768 bytes στο κανάλι 1

# SCALE (Γραφικά)

Η **SCALE** επιτρέπει το συντελεστή κλίμακας που χρησιμοποιείται από τις γραφικές διαδικασίες να μεταβληθεί. Μία κλίμακα της "x" σημαίνει ότι μία κάθετη γραμμή μήκους "x" θα γεμίσει τον κάθετο άξονα του παραθύρου στο οποίο σχεδιάζεται το σχήμα. Μια κλίμακα του 100 είναι η default. Η **SCALE** επίσης επιτρέπει να καθοριστεί η αρχή του συστήματος συντεταγμένων. Αυτό αποτελεσματικά επιτρέπει στο παράθυρο που χρησιμοποιείται για τα γραφικά να κινηθεί μέσα σε ένα πολύ μεγαλύτερο διάστημα γραφικών.

syntax:

*x*:= *numeric__expression*
*y*:= *numeric__expression*

*origin*:= *x*,*y*
*scale*:= *numeric__expression*

**SCALE** [*channel*,] *scale*, *origin*

example:

i. `SCALE 0.5,0.1,0.1`
ii. `SCALE 10,0,0`
iii. `CALE 100,50,50`

```
i θέτει την κλίμακα στο 0.5 με αρχή 0.1,0.1
ii  "    "     "     "   10  "   "  0,0
iii "    "     "     "  100  "   "  50,50
```

# SCROLL (Παράθυρα)

Η **SCROLL** θα ρολάρει το προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι πάνω ή κάτω. Το χαρτί ρολάρει μέσα στο πάνω ή το κάτω μέρος για να γεμίσει τον καθορισμένο χώρο.

Μια προαιρετική τρίτη παράμετρος μπορεί να καθοριστεί για να πετύχει ένα μερικό ρολάρισμα της οθόνης.

syntax:

*part*:= *numeric__expression*
*distance*:= *numeric__expression*

όπου:

**part=0** ολόκληρη οθόνη (default σε καμμία παράμετρο)

**part=1** το πάνω μέρος εξαιρούμενης της σειράς του δείκτη

**part=2** το κάτω μέρος εξαιρούμενης της σειράς του δείκτη

**SCROLL** [κανάλι,] αριθμητική έκφραση [,μέρος]

Αν η απόσταση είναι μια θετική τιμή τότε το περιεχόμενο της οθόνης θα μετατοπιστεί κάτω.

example:

i. SCROLL 10
ii. SCROLL -70
iii. SCROLL -10, 2

i Ρολάρει κάτω 10 pixels
ii Ρολάρει πάνω 70 pixels
iii Ρολάρει πάνω το κάτω μέρος του παραθύρου 10 pixels

# SDATE (Ρολόι)

Η εντολή **SDATE** επιτρέπει την επανατοποθέτηση του ρολογιού του QL.

*year*:= *numeric_expression*
*month*:= *numeric_expression*
*day*:= *numeric_expression*
*hours*:= *numeric_expression*
*minutes*:= *numeric_expression*
*seconds*:= *numeric_expression*

SDATE *year*, *month*, *day*, *hours*, *minutes*, *seconds*

i. SDATE 1984,4,2,0,0,0
ii. SDATE 1984,1,12,9,30,0
iii. SDATE 1984,3,21,0,0,0

# SELect
# END SELect

**Συνθήκες**

Η **SELect** επιτρέπει την επιλογή διαφόρων ενεργειών ανάλογα με την τιμή μιας μεταβλητής.

define: *select_variable* := *numeric_variable*

*select_item* := | *expression*
| *expression* **TO** *expression*

*select_list* := | *select_item* *[, *select_item*]*

**Μακριά μορφή**

Επιτρέπει την επιλογή πολλαπλών ενεργειών ανάλογα με την τιμή της select_variable που είναι το τελευταίο μέρος στη λογική γραμμή. Μια σειρά από SuperBASIC εντολές ακολουθεί που τερματίζεται με την επόμενη **ON** εντολή ή με την εντολή **END SELect.** Αν το select_item είναι μια έκφραση τότε γίνεται ένας έλεγχος μέσα σε 1 μονάδα περίπου στα 10 εις την -7, αλλιώς για expression **TO** expression η κλίμακα εξετάζεται ακριβώς. Η εντολή **ON REMAINDER** επιτρέπει να πιαστούν όλα τα οποία θα αντιδράσουν αν δεν ικανοποιηθούν άλλες συνθήκες επιλογής.

syntax:
```
SELect ON select_variable
   *[[ON select_variable] = select_list
       statements]*
    [ON select_variable] = REMAINDER
       statements
END SELect
```

example:
```
100 LET error_number = RND(1 TO 10)
110 SELect ON error_number
120    ON error_number = 1
130       PRINT "Divide by zero"
140       LET error_number = 0
150    ON error_number = 2
160       PRINT "File not found"
170       LET error_number = 0
180    ON error_number = 3 TO 5
190       PRINT "Microdrive file not found"
200       LET error_number = 0
210    ON error_number = REMAINDER
220       PRINT "Unknown error"
230 END SELect
```

Αν η select_variable χρησιμοποιείται στο σώμα της εντολής **SELect** τότε θα πρέπει να ταιριζει με την επιλογή που δόθηκε στην επικεφαλίδα επιλογής.

**Σύντομη μορφή**

Η σύντομη μορφή της εντολής **SELect** επιτρέπει να γίνουν απλές επιλογές μιας γραμμής. Μια ακολουθία από SuperBASIC εντολές ακολουθεί την ίδια λογική γραμμή όπως η εντολή **SELect.** Αν η καθορισμένη συνθήκη στην εντολή επιλογής ικανοποιείται τότε η ακολουθία των SuperBASIC εντολών επεξεργάζεται.

syntax: **SELect ON** *select_variable* = *select_list* : *statement* * [: *statement*] *

example:

i.
```
SELect ON test_data = 1 TO 10 :
PRINT "Answer within range"
```

ii.
```
SELect ON answer = 0.00001 TO 0.00005 :
PRINT "Accuracy OK"
```

iii.
```
SELect ON a = 1 TO 10 : PRINT a ! "in range"
```

Παρατήρηση: Η σύντομη μορφή της εντολής **SELect** επιτρέπει την εξέταση των κλιμάκων πιο εύκολα παρά ότι με την εντολή **IF.** Συγκρίνετε το παραπάνω παράδειγμα ii με την ανάλογη εντολή **IF.**

# SEXEC (Qdos)

Θα φυλάξει μία περιοχή της μνήμης σε μία μορφή που είναι κατάλληλη για φόρτωμα και εκτέλεση με την εντολή **EXEC.**

Τα δεδομένα που φυλάσσονται θα πρέπει να αποτελούν ένα πρόγραμμα γλώσσας μηχανής.

The data saved should constitute a machine code program.

syntax:
*start_address*:= *numeric_expression*
*length*:= *numeric_expression*
*data_space*:= *numeric_expression*

Αρχή περιοχής

Μήκος περιοχής

Μήκος περιοχής δεδομένων στο πρόγραμμα

SEXEC *device*, *start_address*, *length*, *data_space*

example:
```
SEXEC mdv1_program, 262144,3000,500
```

Παρατήρηση: Το εγχειρίδιο του Qdos θα πρέπει να διαβαστεί προτού προσπαθήσετε να χρησιμοποιήσετε αυτήν τη εντολή.

# SIN (Μαθηματικές συναρτήσεις)

Η **SIN** θα υπολογίσει το ημίτονο καθορισμένης παραμέτρου

syntax: *angle*:= *numeric_expression* {range −10000..10000 in radians}

SIN(*angle*)

example:
i. `PRINT SIN(3)`
ii. `PRINT SIN(3.141592654/2)`

# SQRT (Μαθηματικές συναρτήσεις)

θα υπολογίσει την τετραγωνική ρίζα του καθορισμένου ορίσματος. Το όρισμα θα πρέπει να είναι μεγαλύτερο ή ίσο με το μηδέν.

syntax: SQRT (*numeric_expression*) {range >= 0}

example:
i. `PRINT SQRT(3)`
ii. `LET C = SQRT(a^2 + b^2)`

# STOP (BASIC)

Η **STOP** θα τερματίσει την εκτέλεση ενός προγράμματος και θα επιστρέψει την SuperBASIC στις εντολές του interpreter.

syntax: STOP

example:
i. `STOP`
ii. `IF n = 100 THEN STOP`

Μπορείτε να συνεχίσετε με το **CONTINUE** μετά το **STOP.**

Παρατήρηση: Η τελευταία εκτελέσιμη σειρά ενός προγράμματος θα ενεργήσει σαν ένα αυτόματο STOP.

# STRIP (Παράθυρα)

Η **STRIP** θα βάλει το τρέχον χρώμα λωρίδας στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι. Το χρώμα λωρίδας είναι το χρώμα βάθους που χρησιμοποιείται όταν το **OVER 1** επιλέγεται. θέτοντας το **PAPER** θα τοποθετήσει αυτόματα το χρώμα λωρίδας στο νέο χρώμα **PAPER.**

syntax: STRIP [*channel*,] *colour*

example:
i. STRIP 7
ii. STRIP 0,4,2

i Βάζει μια λωρίδα με άσπρο χρώμα
ii Βάζει μια λωρίδα με μαύρα και πράσινα τετράγωνα

Παρατήρηση: Το αποτέλεσμα του **STRIP** είναι σαν να χρησιμοποιείται μία πένα τονισμού.

# TAN (Μαθηματικές συναρτήσεις)

Η **TAN** θα υπολογίσει την εφαπτομένη του καθορισμένου ορίσματος. Το όρισμα θα πρέπει να είναι μεσα στην κλίμακα από -30000 ως 30000 και θα πρέπει να ορίζεται σε ακτίνια.

syntax: TAN (*numeric__expression*)

example:
i. TAN(3)
ii. TAN(3.141592654/2)

# TURN
# TURNTO (γραφικά χελώνας)

Η **TURN** επιτρέπει στην κεφαλή της χελώνας να στραφεί μέσα από μια συγκεκριμένη γωνία ενώ το **TURNTO** επιτρέπει στη χελώνα να στραφεί προς μια συγκεκριμένη κεφαλή.

Η χελώνα στρέφεται στο παράθυρο που είναι προσαρτιμένο στο καθορισμένο ή default κανάλι.

Η γωνία καθορίζεται σε μοίρες. 'Ενας θετικός αριθμός μπορεί να γυρίσει τη χελώνα αντίθετα με τη φορά του ρολογιού και ένας αρνητικός αριθμός θα το γυρίσει με τη φορά του.

Αρχικά η χελώνα δείχνει 0, δηλαδή στη δεξιά πλευρά του παραθύρου.

syntax: *angle*:= *numeric_expression* {angle in degrees}

**TURN** [*channel*,] *angle*
**TURNTO** [*channel*,] *angle*

example:
i. `TURN 90` `i Στροφή 90 μοιρών`
ii. `TURNTO 0` `ii Στροφή 0 μοιρών`

# UNDER (Παράθυρα)

Ανοίγει ή κλείνει την υπογράμμιση για τις επόμενες σειρές αποτελεσμάτων. Η υπογράμμιση είναι στο τρέχον χρώμα **INK** στο προσαρτημένο παράθυρο στο συγκεκριμένο ή default κανάλι.

syntax: *switch*:= *numeric_expression* {range 0..1}

**UNDER** [*channel*,] *switch*

example:
i. `UNDER 1` `i Ανοίγει την υπογράμμιση`
ii. `UNDER 0` `ii Κλείνει την υπογράμμιση`

# WIDTH (Συσκευές)

Η **WIDTH** επιτρέπει να καθοριστεί το default πλάτος για συσκευές που δεν είναι κονσόλες, για παράδειγμα εκτυπωτές.

syntax: *line__width*:= *numeric__expression*
**WIDTH** [*channel*,] *line__width*

example:
i. `WIDTH 80`
ii. `WIDTH #6, 72`

i θέτει το πλάτος σε 80
ii θέτει το πλάτος στη συσκευή στο κανάλι 6, σε 72

# WINDOW (Παράθυρα)

Επιτρέπει στο χρήστη να αλλάξει τη θέση και το μέγεθος του προσαρτημένου παράθυρου στο συγκεκριμένο ή default κανάλι. Αν υπάρχουν περιθώρια, μετακινούνται όταν επανακαθοριστεί το παράθυρο.

Οι συντεταγμένες καθορίζονται χρησιμοποιώντας το σύστημα pixels που είναι σχετικό με την αρχή της οθόνης.

syntax: *width*:= *numeric__expression*
*depth*:= *numeric__expression*
*x*:= *numeric__expression*
*y*:= *numeric__expression*

**WINDOW** [*channel*,] *width*, *depth*, *x*, *y*

example: `WINDOW 30, 40, 10, 10`
**Παράθυρο 30 X 40 pixels στη θέση 10,10**

UNIVERSITY STUDIO PRESS
Εκδόσεις Επιστημονικών Βιβλίων & Περιοδικών
Κων. Μελενίκου 15 - τηλ. 209 637 & 209 837 - Θεσ/νίκη