Χρήση Δομών Ελέγχου

Η Ανακύκλωση DO WHILE ... ENDDO

Εκτελεί την ομάδα εντολών που βρίσκεται μεταξύ του DO WHILE και του ENDDO έως ότου πάψει να ισχύει μια συνθήκη. Ο έλεγχος της συνθήκης γίνεται πάντα πριν μπούμε στο βρόχο και έτσι πρέπει να υπάρχει κάποια αρχική τιμή για τη μεταβλητή ή τις μεταβλητές που παίρνουν μέρος στη συνθήκη.

Υπάρχει, βέβαια, περίπτωση να μη μπούμε καθόλου στον βρόχο, όπως υπάρχει και περίπτωση να μην μπορούμε να βγούμε από τον βρόχο από δικό μας λογικό λάθος στο πρόγραμμα. Ο Clipper δεν διαθέτει δομή ανάλογη της δομής *Repeat ... Until* της Pascal.

Στο παρακάτω παράδειγμα, οι εντολές θα εκτελούνται όσο το I είναι μικρότερο από το 10. Μόλις γίνει ίσο ή μεγαλύτερο του 10, θα βγούμε από το βρόχο.

```
...
NAME := SPACE(20)
I := 1
DO  WHILE  I<10
        @ 3, 5  SAY  'Πώς λέγεσαι; '  GET  NAME  PICTURE  '@A'
        READ
        I := I + 1
ENDDO
...
```

Ανάμεσα στις εντολές DO WHILE και ENDDO θα πρέπει να υπάρχει μια εντολή που να αλλάζει την τιμή της μεταβλητής που υπάρχει στη συνθήκη ανακύκλωσης· στο παράδειγμά μας δηλ., την τιμή του I. Αν δεν υπάρχει καμία τέτοια εντολή, τότε η ανακύκλωση θα εκτελείται επ' άπειρον *(ατέρμων βρόχος)* και για να διακόψουμε το πρόγραμμα θα πρέπει να πατήσουμε Ctrl-C.

Μπορούμε, όμως, να βγούμε από μια ανακύκλωση ή να επιστρέψουμε στην αρχή της πριν ακόμα αυτή ολοκληρωθεί κανονικά. Οι εντολές του Clipper που το κάνουν αυτό είναι οι EXIT και LOOP. Με την EXIT πηγαίνουμε στο τέλος της ανακύκλωσης και μετά βγαίνουμε εκτός. Με τη LOOP πηγαίνουμε στο τέλος της ανακύκλωσης και αμέσως μετά στην αρχή της, για να συνεχιστεί η επαναληπτική διαδικασία.

Δείτε το παρακάτω παράδειγμα :

```
...
DO WHILE .T.
  @ 2, 6  SAY  'Δώσε έναν θετικό αριθμό έως 999 - Έξοδος με 999'
  @ 3, 6  SAY  'Δώσε τον αριθμό : '  GET NUM  PICTURE  '999'
  READ
  IF NUM = 999
        EXIT
  ENDIF
  IF NUM <= 0
        LOOP
  ENDIF
  /* εντολές */
ENDDO
...
```

Η παραπάνω δομή βρόχου είναι φαινομενικά ατελείωτη, γιατί η συνθήκη εισόδου στον βρόχο είναι πάντα αληθής (DO WHILE .T.). Μπορούμε, όμως, να βγούμε από τον βρόχο με την εντολή EXIT αν δώσουμε αριθμό ίσο με 999 και να επιστρέψουμε στην αρχή του βρόχου με την εντολή LOOP αν δώσουμε έναν αριθμό μικρότερο ή ίσο του μηδενός.

Η Δομή IF ... ENDIF

Εκτελεί κάποιες εντολές αν ικανοποιείται μια συνθήκη, ενώ αν η συνθήκη δεν ικανοποιείται, μπορεί να εκτελεστεί μια εναλλακτική ομάδα εντολών.

Η γενική σύνταξή της είναι :

IF *<συνθήκη1>*
        <εντολές>
[ELSEIF *<συνθήκη2>]***
        <εντολές>
[ELSE]
        <εντολές>
ENDIF

Κάθε δομή IF πρέπει να τελειώνει με μια αντίστοιχη δήλωση ENDIF. Οι δηλώσεις ELSEIF και ELSE είναι προαιρετικές. Αν η <συνθήκη1> είναι αληθής, τότε θα εκτελεστούν οι εντολές που την ακολουθούν και οι οποίες θα πρέπει να είναι γραμμένες στην επόμενη γραμμή.

Αν η <συνθήκη1> είναι ψευδής, τότε θα ελεγχθεί η <συνθήκη2>, αν υπάρχει βέβαια, και αν η <συνθήκη2> είναι αληθής, τότε θα εκτελεστούν οι εντολές που την ακολουθούν. Αν η <συνθήκη2> είναι ψευδής, τότε θα εκτελεστούν οι εντολές που ακολουθούν τη δήλωση ELSE.

Δείτε τα παρακάτω παραδείγματα :

```
...
IF NUM % 2 = 0
        ? 'Ο αριθμός είναι άρτιος'
ELSE
        ? 'Ο αριθμός είναι περιττός'
ENDIF
...
```

```
...
IF BATHMOS < 10
        ? 'Απορρίπτεσαι'
ELSEIF BATHMOS >= 10 .AND. BATHMOS < 15
        ? 'Περνάς, αλλά να διαβάζεις'
ELSEIF BATHMOS >= 15 .AND. BATHMOS < 18
        ? 'Μπορείς να βελτιωθείς κι άλλο'
ELSE
        ? 'Περνάς με άριστα'
ENDIF
...
```

Υπάρχει και η συνάρτηση IIF(<συνθήκη>, <εντολή1>, <εντολή2>), η οποία εκτελεί την <εντολή1> αν η <συνθήκη> είναι αληθής, αλλιώς εκτελεί την <εντολή2>.

Δείτε το παρακάτω παράδειγμα :

```
? IIF(A<B, 'το Β είναι μεγαλύτερο', 'το Α είναι μεγαλύτερο')
```

Η Δομή DO CASE ... ENDCASE

Η δομή αυτή είναι ένας άλλος τρόπος για να γράψουμε μια δομή IF που έχει πολλές ELSEIF περιπτώσεις.

Η γενική σύνταξή της είναι :

```
DO CASE
        CASE <συνθήκη1>
                <εντολές>
        [CASE <συνθήκη2>]
                <εντολές>
        [CASE <συνθήκη3>]
                <εντολές>
        ...
        [OTHERWISE]
                <εντολές>
ENDCASE
```

Η δομή αυτή εκτελεί μόνο τις εντολές που ακολουθούν εκείνη τη δήλωση CASE της οποίας η συνθήκη αληθεύει. Αφού εκτελεστούν αυτές οι εντολές, το πρόγραμμα θα συνεχίσει με τις εντολές που βρίσκονται κάτω από τη δήλωση ENDCASE.

Η δήλωση OTHERWISE είναι προαιρετική και αναλαμβάνει την εκτέλεση των εντολών που την ακολουθούν, αν καμία από τις προηγούμενες CASE δεν αληθεύει. Είναι καλό να έχουμε πάντα μια δήλωση OTHERWISE σε μια δομή CASE, ακόμα κι αν είμαστε σίγουροι ότι αυτή δεν πρόκειται να εκτελεστεί.

Δείτε πώς γράφεται το προηγούμενο παράδειγμα με χρήση της δομής DO CASE ... ENDCASE :

```
...
DO CASE
        CASE BATHMOS < 10
                ? 'Απορρίπτεσαι'
        CASE BATHMOS >= 10 .AND. BATHMOS < 15
                ? 'Περνάς, αλλά να διαβάζεις'
        CASE BATHMOS >= 15 .AND. BATHMOS < 18
                ? 'Μπορείς να βελτιωθείς κι άλλο'
        OTHERWISE
                ? 'Περνάς με άριστα'
ENDCASE
...
```

Η δομή CASE είναι πολύ χρήσιμη για τον έλεγχο των διακλαδώσεων ενός μενού. Όπως θα δούμε παρακάτω, μια δομή μενού επιστρέφει μια ακέραια τιμή, η οποία δείχνει την επιλογή του χρήστη. Οι τιμές που μπορεί να πάρει αυτή η ακέραια τιμή είναι από 0 έως τον αριθμό των επιλογών του μενού.

Δείτε το παρακάτω παράδειγμα, όπου η επιλογή του χρήστη από το μενού έχει καταχωρηθεί στη μεταβλητή Menu_Choice :

```
...
DO CASE
        CASE Menu_Choice = 0 .OR. Menu_Choice = 5
                CLEAR
                QUIT
        CASE Menu_Choice = 1
                Add_Customer()
        CASE Menu_Choice = 2
                Edit_Customer()
        CASE Menu_Choice = 3
                Report_Customer()
        CASE Menu_Choice = 4
                Utilities()
ENDCASE
```

Κάθε επιλογή που κάνει ο χρήστης στο μενού, ενεργοποιεί μια διαφορετική CASE, η οποία με τη σειρά της εκτελεί και μια διαφορετική διαδικασία (procedure) για να γίνουν οι αντίστοιχες ενέργειες. Η επιλογή 0 σημαίνει ότι πατήθηκε το πλήκτρο ESC και η επιλογή 5 είναι η τελευταία επιλογή του μενού, που συνήθως είναι η κανονική έξοδος από το μενού. Έτσι, μπορούμε να βγούμε από ένα μενού πατώντας 0 ή την τελευταία επιλογή του μενού.

Η εντολή CLEAR καθαρίζει την οθόνη, διαγράφει τα περιεχόμενα των εντολών GET και τοποθετεί τον δρομέα στην πάνω αριστερή γωνία της οθόνης. Η εντολή QUIT κλείνει όλα τα ανοικτά αρχεία βάσεων δεδομένων, τερματίζει την εκτέλεση του προγράμματος και επιστρέφει τον έλεγχο στο DOS.

Η Δομή FOR ... NEXT

Επαναλαμβάνει τις περικλειόμενες εντολές για έναν συγκεκριμένο αριθμό φορών.

Η σύνταξη της εντολής είναι :

FOR *I := <αρχική τιμή>* ***TO*** *<τελική τιμή>* ***STEP*** *<βήμα>*
? I
NEXT *I*

Δείτε κι ένα παράδειγμα :

FOR I := 1 TO 10 STEP 2
? 'Γεια σου για ' + I + 'η φορά'
NEXT I

Η δήλωση STEP είναι προαιρετική και αυξάνει ή μειώνει ανάλογα τον μετρητή I.

Οι Δομές Μενού

Θα δούμε πρώτα μερικές χρήσιμες εντολές του Clipper.

Η Εντολή @ ... SAY

Με την εντολή αυτή μπορούμε να εμφανίσουμε μηνύματα σε καθορισμένες συντεταγμένες της οθόνης.

Η βασική σύνταξη της εντολής είναι :

@ <γραμμή>, <στήλη> ***SAY*** *<έκφραση>*

Οι τιμές<γραμμή> και <στήλη> είναι οι συντεταγμένες της οθόνης απ' όπου θα αρχίζει να εμφανίζεται η <έκφραση> και μπορούν να έχουν τιμές από 0-23 για τη <γραμμή> και από 0-79 για τη <στήλη>. Τα όρια γραμμής και στήλης της οθόνης μπορούν να προσδιοριστούν από τις συναρτήσεις MAXROW() και MAXCOL() αντίστοιχα.

Δείτε παραδείγματα της εντολής @ ... SAY :

@ 2, 3 SAY DATE()
@ 4, 15 SAY 'Το αρχαίο όνομα της Φλώρινας είναι Λυγκηστίς'

Η Εντολή @ ... CLEAR

Καθαρίζει ένα συγκεκριμένο κομμάτι της οθόνης. Τη χρησιμοποιούμε συνήθως για να καθαρίσουμε τα μενού μας από υπολείμματα προηγούμενων εντολών.

Η βασική σύνταξη της εντολής είναι :

@ <πάνω>, <αριστερά> ***CLEAR TO*** *<κάτω>, <δεξιά>*

Καθαρίζει την περιοχή της οθόνης (ορθογώνιο) που ορίζεται από τις τέσσερις συντεταγμένες, όπως στο επόμενο παράδειγμα :

@ 3, 5 CLEAR TO 6, 70

Αν χρησιμοποιηθεί χωρίς τις δύο τελευταίες συντεταγμένες και χωρίς τις λέξεις CLEAT TO, τότε καθαρίζει την περιοχή της οθόνης από το σημείο που ορίζουν οι δύο πρώτες συντεταγμένες και μέχρι τις τιμές των MAXROW() και MAXCOL(), όπως στο επόμενο παράδειγμα :

@ 4, 8

Η Εντολή @ ... TO

Σχεδιάζει ένα πλαίσιο στην οθόνη.

Η βασική σύνταξη της εντολής είναι :

@ <πάνω>, <αριστερά> ***TO*** *<κάτω>, <δεξιά>* ***[DOUBLE]***

Αν χρησιμοποιηθεί η επιλογή [DOUBLE], τότε θα εμφανιστεί ένα πλαίσιο διπλής γραμμής. Αν οι συντεταγμένες <πάνω> και <κάτω> είναι ίδιες, θα σχεδιαστεί μια οριζόντια γραμμή, ενώ αν είναι ίδιες οι συντεταγμένες <αριστερά> και <δεξιά>, θα σχεδιαστεί μια κατακόρυφη γραμμή.

Δείτε τα παρακάτω παραδείγματα :

@ 5, 10 CLEAR TO 10, 40
@ 5, 10 TO 10, 40
@ 8, 15 TO 8, 35
@ 12, 14 TO 25, 14

Η Εντολή @ ... PROMPT

Είναι η βασικότερη εντολή για τον σχεδιασμό ενός μενού.

Η βασική σύνταξη της εντολής είναι :

SET WRAP ON
SET MESSAGE TO 20 [CENTER]
@ <γραμμή1>, <στήλη1> PROMPT <στοιχείο μενού1>;
[MESSAGE <μήνυμα1>]
@ <γραμμή2>, <στήλη2> PROMPT <στοιχείο μενού2>;
[MESSAGE <μήνυμα2>]
...
MENU TO Menu_Choice

Η εντολή PROMPT λειτουργεί ως εξής. Εμφανίζει στις καθορισμένες γραμμές και στήλες τα <στοιχεία μενού> και μας επιτρέπει να κινηθούμε ανάμεσα στις επιλογές με τα βελάκια κίνησης και να πατήσουμε το πλήκτρο <enter> για να επιλέξουμε το στοιχείο που θέλουμε. Ανάλογα με την επιλογή μας, η μεταβλητή Menu_Choice παίρνει μια τιμή από 1 έως την τελευταία επιλογή που υπάρχει στο μενού.

Αν πατήσουμε το πλήκτρο ESC, η τιμή της Menu_Choice θα γίνει ίση με 0. Αν η μεταβλητή αυτή είχε κάποια τιμή πριν από την εμφάνιση του μενού, τότε θα φωτιστεί το αντίστοιχο στοιχείο του μενού, αλλιώς φωτίζεται έντονα η πρώτη επιλογή του μενού.

Ένα μενού μπορεί να είναι οριζόντιο ή κατακόρυφο, ανάλογα με την επιλογή των συντεταγμένων που έχουμε κάνει. Τα <μηνύματα> που ακολουθούν τις δηλώσεις MESSAGE είναι επεξηγηματικά των επιλογών του μενού και εμφανίζονται σε μια συγκεκριμένη γραμμή της οθόνης.

Η δήλωση SET WRAP ON κάνει το μενού αναδιπλούμενο, δηλ. μας επιτρέπει να πάμε από τη βάση του μενού στην κορυφή του και ανάποδα. Η δήλωση SET MESSAGE TO 20 [CENTER] τοποθετεί το διευκρινιστικό <μήνυμα> για κάθε επιλογή του μενού στη γραμμή 20 της οθόνης και αν χρησιμοποιήσουμε και την επιλογή [CENTER], το μηνύματα θα είναι κεντραρισμένα. Η δήλωση SET MESSAGE TO, χωρίς παραμέτρους, απενεργοποιεί την εμφάνιση μηνυμάτων.

Η επιλογή που κάνουμε σ' ένα μενού μπορεί να ενεργοποιεί ένα άλλο μενού (υπομενού) και η εντολή DO CASE ... ENDCASE είναι η πλέον κατάλληλη για να χειριστούμε την επιλογή ενός μενού.

Δείτε το παρακάτω παράδειγμα :

```
/* εμφανίζει ένα οριζόντιο μενού */
LOCAL  Menu_Choice := 0

CLEAR SCREEN
@ 0, 65  SAY  DATE()       /* εμφανίζει την ημερομηνία του συστήματος */
@ 1, 1  TO  4, 70  DOUBLE
/* σχεδιάζει ένα οριζόντιο πλαίσιο διπλής γραμμής στην οθόνη
για να περικλείσει το μενού */

SET  WRAP  ON              /* αναδίπλωση του μενού */
SET  MESSAGE  TO  20  CENTER
@ 2, 01  PROMPT  'Πελάτες'  MESSAGE  'Στοιχεία των Πελατών'
@ 2, 15  PROMPT  'Προμηθευτές'  MESSAGE  'Στοιχεία των Προμηθευτών'
@ 2, 35  PROMPT  'Αποθήκη'  MESSAGE  'Στοιχεία της Αποθήκης'
@ 2, 55  PROMPT  'Έξοδος'  MESSAGE  'Έξοδος από το Πρόγραμμα'
MENU  TO  Menu_Choice
/* η επιλογή του μενού καταχωρείται στη μεταβλητή Menu_Choice */

/* χρήση μιας δομής DO CASE ... ENDCASE */
DO CASE
        CASE Menu_Choice = 0 .OR. Menu_Choice = 4
                CLEAR
                QUIT
        CASE Menu_Choice = 1
                Customers()
                /* καλεί τη ρουτίνα των πελατών */
        CASE Menu_Choice = 2
                Suppliers()
                /* καλεί τη ρουτίνα των προμηθευτών */
        CASE Menu_Choice = 3
                Inventory()
                /* καλεί τη ρουτίνα της αποθήκης */
        OTHERWISE
                ?  'κάποιος λάθος πρέπει να συμβαίνει'
                QUIT
ENDCASE
```

Καθεμία από τις διαδικασίες CUSTOMERS(), SUPPLIERS() και INVENTORY(), μπορεί μετά να δημιουργεί ένα δικό της υπομενού για να χειριστεί την κάθε περίπτωση, το οποίο θα πρέπει να είναι κατακόρυφο.

Στο επόμενο παράδειγμα, εμφανίζεται ένα κατακόρυφο μενού για τους πελάτες, το οποίο ενεργοποιείται όταν έχουμε επιλέξει την πρώτη επιλογή του βασικού μενού.

```
/* εμφανίζει ένα κατακόρυφο υπομενού πελατών */
PROCEDURE CUSTOMERS()
LOCAL  Menu_Choice := 0

@ 2, 1  TO  14, 20  DOUBLE
/* σχεδιάζει ένα κατακόρυφο πλαίσιο διπλής γραμμής στην οθόνη
για να περικλείσει το μενού */

SET  WRAP  ON              /* αναδίπλωση του μενού */
SET  MESSAGE  TO  20  CENTER
@ 3, 2  PROMPT  'Προσθήκη Πελατών';
            MESSAGE  'Προσθήκη Νέων Πελατών'
@ 4, 2  PROMPT  'Εμφάνιση Πελατών';
            MESSAGE  'Επεξεργασία των Στοιχείων των Πελατών'
@ 5, 2  PROMPT  'Διαγραφή Πελατών';
            MESSAGE  'Οριστική Διαγραφή Πελατών'
@ 6, 2  PROMPT  'Έξοδος';
            MESSAGE  'Επιστροφή στο Βασικό Μενού'
MENU  TO  Menu_Choice
/* η επιλογή του μενού καταχωρείται στη μεταβλητή Menu_Choice */

/* χρήση μιας δομής DO CASE ... ENDCASE */
DO CASE
        CASE Menu_Choice = 0 .OR. Menu_Choice = 4
                @ 2, 1  CLEAR  TO  14, 20
                RETURN
        CASE Menu_Choice = 1
                Add_Customers()
                /* ρουτίνα καταχώρησης στοιχείων νέων πελατών */
        CASE Menu_Choice = 2
                Edit_Customers()
                /* ρουτίνα εμφάνισης στοιχείων πελατών */
        CASE Menu_Choice = 3
                Del_Customers()
                /* ρουτίνα διαγραφής πελατών */
        OTHERWISE
                ?  'κάποιος λάθος πρέπει να συμβαίνει'
                RETURN
ENDCASE
```

Αποθήκευση και Αποκατάσταση Οθονών

Ο Clipper επιτρέπει την αποθήκευση και αποκατάσταση των οθονών μας, στην ουσία ενός τμήματος της οθόνης, έτσι ώστε να μπορούμε να τις επικαλύψουμε με νέα μενού και άλλες πληροφορίες και στη συνέχεια να τις αποκαταστήσουμε στην αρχική τους μορφή. Αυτό επιτυγχάνεται με τις συναρτήσεις SAVESCREEN() και RESTSCREEN().

Η σύνταξή τους είναι :

SAVESCREEN(<πάνω>, <αριστερά>, <κάτω>, <δεξιά>)
και
RESTSCREEN(<πάνω>, <αριστερά>, <κάτω>, <δεξιά>,;
<μεταβλητή>)

Δείτε το παρακάτω παράδειγμα :

```
PROCEDURE S_Saver
        LOCAL cScreen, cColor
        cScreen = SAVESCREEN(0, 0, MAXROW(), MAXCOL())
        cColor = SETCOLOR()
        Menu_Proc()
        SETCOLOR(cColor)
        RESTSCREEN(0, 0, MAXROW(), MAXCOL(), cScreen)
RETURN
```

Στο παραπάνω πρόγραμμα, ολόκληρη η οθόνη αποθηκεύεται στη μεταβλητή cScreen και μετά καλείται η διαδικασία Menu_Proc(). Μετά την επιστροφή από την Menu_Proc(), αποκαθίσταται η αρχική οθόνη και τα χρώματά της.

Λειτουργίες Εισόδου/Εξόδου

Στον Clipper η εμφάνιση και η καταχώρηση δεδομένων γίνεται πολύ απλά και αποδοτικά με τη χρήση των εντολών @ ... SAY ... GET. Η εντολή @ ... SAY, που είδαμε πρωτύτερα, εμφανίζει ένα μήνυμα σε μια καθορισμένη θέση στην οθόνη και η εντολή GET μάς επιτρέπει να καταχωρούμε δεδομένα σε κάποια μεταβλητή ή απλά να εμφανίζουμε τα περιεχόμενά της.

Αν τις εντολές αυτές τις ακολουθεί μια εντολή READ, τότε μπορούμε να κάνουμε καταχωρήσεις τιμών στις μεταβλητές της GET και μάλιστα μπορούμε να μετακινούμαστε με τα βελάκια πάνω-κάτω και δεξιά-αριστερά ανάμεσα στις μεταβλητές. Με τη δήλωση PICTURE μπορούμε να καθορίσουμε τη φόρμα εμφάνισης ή καταχώρησης των δεδομένων. Για να φύγουμε, χωρίς να αποθηκευθούν τα δεδομένα, πατάμε το πλήκτρο ESC.

Δείτε τις παρακάτω εντολές :

```
@ 2, 3  SAY  'Δώσε επώνυμο      : '  GET  EPONYMO  PICTURE  '@!A'
@ 3, 3  SAY  'Δώσε όνομα        : '  GET  NAME  PICTURE  '@A'
@ 4, 3  SAY  'Δώσε διεύθυνση    : '  GET  ADDRESS  PICTURE  '@X'
@ 5, 3  SAY  'Δώσε ταχ. κώδ.    : '  GET  TK  PICTURE  '999 99'
@ 6, 3  SAY  'Δώσε βαθμό        : '  GET  BATHMOS  PICTURE  '99'
READ
```

Η πρώτη γραμμή δέχεται μόνο αλφαβητικούς χαρακτήρες για το επώνυμο και ό,τι γράφουμε το μετατρέπει αυτόματα σε κεφαλαία, η δεύτερη δέχεται μόνο αλφαβητικούς χαρακτήρες για το όνομα, η τρίτη δέχεται οποιονδήποτε χαρακτήρα, η τέταρτη δέχεται μόνο νούμερα μ' ένα κενό ανάμεσά τους και η πέμπτη δέχεται έναν αριθμό με το πολύ δύο νούμερα.

Η εντολή :

```
@ 12, 10  SAY  'Αριθμός : '  GET  number  PICTURE  '99';
        RANGE 10, 40
READ
```

δέχεται μόνο αριθμούς με τιμές από 10 έως και 40.

Η εντολή :

```
@ 20, 25  SAY  'Να ξαναπροσπαθήσω ; (N/O)'  GET  keystroke;
        PICTURE  'X'  VALID keystroke  $  'νNoO'
READ
```

δέχεται τιμές μόνο από το σύνολο 'νNoO'.

Η εντολή SET BELL ON ενεργοποιεί το «καμπανάκι» όταν δώσουμε μια μη αποδεκτή τιμή σε μια GET ή όταν γεμίσει το πεδίο μιας GET.

Με την εντολή SET CONFIRM OFF λέμε στον υπολογιστή ότι δεν χρειάζεται να πατάμε το <enter> για να βγούμε από ένα GET, όταν αυτό έχει γεμίσει από χαρακτήρες.

Για την εμφάνιση δεδομένων στην οθόνη μπορούμε να χρησιμοποιήσουμε και τις εντολές ? και ??, όπου η πρώτη αλλάζει γραμμή μετά την εκτύπωση, ενώ η δεύτερη όχι. Την ίδια δουλειά με τις εντολές αυτές κάνουν και οι συναρτήσεις QOUT() και QQOUT() αντίστοιχα.

Δείτε τα παρακάτω παραδείγματα :

```
?  'Το Παγκόσμιο Κύπελλο Ποδοσφαίρου'        /* αλλαγή γραμμής */
??  'θα γίνει το καλοκαίρι του 1998 '        /* μένει στην ίδια γραμμή */
QOUT('στη Γαλλία')                           /* αλλαγή γραμμής */
QQOUT("Ενα παράδειγμα με τη συνάρτηση QQOUT()')
```