ΚΕΦΑΛΑΙΟ 6
ΧΡΗΣΙΜΕΣ ΕΝΤΟΛΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ ΤΟΥ CLIPPER

Μέθοδοι Εξόδου των Δεδομένων

Έχουμε ήδη δει τις εντολές @ SAY και ?, με τις οποίες μπορούμε να εμφανίσουμε αποτελέσματα στην οθόνη. Με τις εντολές αυτές μπορούμε να συνδυάζουμε παραστάσεις στην ίδια πρόταση με τον τελεστή +, ως εξής :

? "Γεια σου, "+ " " + "Γιώργο"

Ο τελεστής + συνενώνει τις διάφορες παραστάσεις και έτσι η παράσταση "Γεια σου" συνενώνεται μ' ένα κενό διάστημα και με την παράσταση "Γιώργο".

Στις εντολές @ SAY και ? μπορούμε να χρησιμοποιούμε και μεταβλητές ή συναρτήσεις του Clipper ή συναρτήσεις δικές μας, ως εξής :

? "Η ημερομηνία είναι :" + DTOC(DATE())

Επειδή η συνάρτηση DATE() επιστρέφει μια τιμή τύπου ημερομηνίας, θα πρέπει να την μετατρέψουμε σε αλφαριθμητικό. Αυτό γίνεται με τη συνάρτηση ***DTOC()*** (date to character - μετατρέπει ημερομηνία σε αλφαριθμητικό). Είναι, δηλ., η αντίθετη της συνάρτησης CTOD().

Μορφοποίηση Δεδομένων με Συναρτήσεις του Clipper

Η συνάρτηση ***LTRIM()*** αφαιρεί τα κενά διαστήματα από την αρχή ενός αλφαριθμητικού, η ***RTRIM()*** τα αφαιρεί από το τέλος και η ***ALLTRIM()*** τα αφαιρεί και από την αρχή και από το τέλος.

Π.χ.
NAME := LTRIM(" Γεωργιάδης ")
STORE ALLTRIM(" ΙΕΚ Φλώρινας ") TO ONOMA_IEK

Η συνάρτηση ***PADC()*** κεντράρει ένα αλφαριθμητικό ή έναν αριθμό μέσα σ' ένα δεδομένο πλάτος χαρακτήρων. Η συνάρτηση ***PADL()*** στοιχίζει ένα αλφαριθμητικό ή έναν αριθμό αριστερά μέσα σ' ένα δεδομένο πλάτος χαρακτήρων και η ***PADR()*** στοιχίζει δεξιά.

Π.χ.
? PADC(DATE(), 20)
? PADL(MISTHOS, 40)
? PADR("Clipper 5.01", 60)

Για να μετατρέψουμε αριθμητικά δεδομένα σε αλφαριθμητικά, χρησιμοποιούμε τη συνάρτηση ***STR()***.

Π.χ.

VAR1 := 1234.56
VAR2 := STR(VAR1, 7, 2) // επιστρέφει "1234.56"

Η δεύτερη παράμετρος της συνάρτησης STR(), που είναι προαιρετική, καθορίζει το μήκος που θέλουμε να έχει το αλφαριθμητικό. Δηλ. στο παραπάνω παράδειγμα, μήκος 7 χαρακτήρων. Η τρίτη παράμετρος, που είναι κι αυτή προαιρετική, δηλώνει τον αριθμό των δεκαδικών που θα συμπεριληφθούν, εδώ 2.

Δείτε κι ένα άλλο παράδειγμα :

VAR := STR(123.456, 10, 1) // επιστρέφει " 123.4"

Είσοδος Δεδομένων

Εκτός από τον συνδυασμό των εντολών @ SAY, GET και READ, υπάρχουν κι άλλες εντολές στον Clipper για την εισαγωγή δεδομένων.

Η εντολή ***ACCEPT*** εμφανίζει ένα μήνυμα και αποθηκεύει αυτά που γράφουμε σε μια μεταβλητή τύπου χαρακτήρα.

Π.χ.

ACCEPT "Τέλος Προγράμματος; (Ν/Ο)" TO ANSWER

Η εντολή ***INPUT*** εμφανίζει ένα μήνυμα και αποθηκεύει αυτά που γράφουμε σε μια μεταβλητή ανάλογου τύπου.

Π.χ.

INPUT "Τιμή προϊόντος : " TO TIMH

Υπάρχει και η εντολή ***WAIT***, η οποία εμφανίζει ένα μήνυμα στην οθόνη και περιμένει να πατήσουμε ένα πλήκτρο για να συνεχιστεί το πρόγραμμα. Αν την δώσουμε χωρίς παραμέτρους εμφανίζει το μήνυμα : "*Press any key to continue*" και όταν πατήσουμε ένα πλήκτρο, το πρόγραμμα συνεχίζει κανονικά. Η άλλη μορφή της εντολής WAIT είναι :

WAIT "Δώσε ημέρα της εβδομάδας : " TO MERA

Το πρόγραμμα εμφανίζει το παραπάνω μήνυμα στην οθόνη και μόλις πατήσουμε ένα πλήκτρο, αποθηκεύει την ASCII τιμή του πλήκτρου στη μεταβλητή MERA και συνεχίζει κανονικά.

Φιλτράρισμα Δεδομένων

Ένα ***φίλτρο*** *(filter)* επενεργεί σε μια βάση δεδομένων και εμφανίζει μόνο ορισμένες εγγραφές. Με την εντολή ***SET FILTER TO*** μπορούμε να δημιουργήσουμε φίλτρα σε μια βάση δεδομένων. Δείτε το παρακάτω παράδειγμα, όπου ένα φίλτρο επιδρά σ' ένα αρχείο πελατών με βάση τον κωδικό του νομού που είναι ο πελάτης.

```
USE  Customer
l_state := SPACE(2)
CLEAR  SCREEN
@ 12, 10  "Δώσε τον κωδικό του νομού : "  GET  l_state
READ

SET  FILTER  TO  Fstate = l_state
DO  WHILE  .NOT.  EOF()
        // επεξεργασία εγγραφών
        SKIP
ENDDO
SET  FILTER  TO
```

Ο χρήστης δίνει τον κωδικό του νομού, που καταχωρείται στη μεταβλητή l_state. Η εντολή SET FILTER TO Fstate = l_state εφαρμόζει ένα φίλτρο στη βάση δεδομένων Customer και εμφανίζει μόνο εκείνες τις εγγραφές που ο κωδικός του νομού (Fstate) είναι ίσος με την τιμή της l_state. Η εντολή SET FILTER TO χωρίς παραμέτρους καταργεί το φίλτρο και δεν θα πρέπει να ξεχνάμε να την δίνουμε όταν δεν χρειαζόμαστε άλλο το φίλτρο.

Επεξεργασία των Εγγραφών μιας Βάσης Δεδομένων

Ο βρόχος DO WHILE μπορεί να χρησιμοποιηθεί για την επεξεργασία των εγγραφών μιας βάσης δεδομένων. Το πρόγραμμα τοποθετεί τον δείκτη εγγραφής στην επιθυμητή θέση σε μια βάση δεδομένων και επεξεργάζεται τις εγγραφές μέχρι να φθάσει στο τέλος του αρχείου ή μέχρι να ικανοποιηθεί κάποια άλλη συνθήκη. Αν πρέπει να επεξεργαστούμε ολόκληρη τη βάση δεδομένων, από την πρώτη εγγραφή ως την τελευταία, θα κάνουμε τα εξής :

```
USE Testdb
DO WHILE .NOT. EOF()// μέχρι να φθάσουμε στο τέλος του αρχείου
        // επεξεργασία της εγγραφής
        SKIP
ENDDO
```

Ο βρόχος DO WHILE μπορεί να χρησιμοποιηθεί και για την επεξεργασία δευτερευουσών εγγραφών :

```
USE  Customer  NEW              // πρωτεύουσα βάση δεδομένων
USE  Trans  NEW                 // δευτερεύουσα βάση δεδομένων
SET  INDEX  TO  Transid         // άνοιγμα αρχείου ευρετηρίου
SEEK  Customer->Id              // αναζήτηση της πρώτης ταύτισης
IF  FOUND()
      DO WHILE .NOT. EOF() .AND. Trans->Id==Customer->Id
            // επεξεργασία της εγγραφής συναλλαγής
      ENNDO
ELSE
      // δεν υπάρχουν συναλλαγές
ENDIF
```

Ο βρόχος DO WHILE επεξεργάζεται κάθε εγγραφή της δευτερεύουσας βάσης δεδομένων μέχρι να βρεθεί η ένδειξη τέλους αρχείου ή μέχρι να αλλάξει η τιμή του πεδίου Id, που σημαίνει ότι δεν υπάρχουν άλλες ταυτίσεις.

Πράξεις με Αλφαριθμητικά

Η συνάρτηση ***AT()*** επιστρέφει τη θέση αρχής μιας αλληλουχίας χαρακτήρων μέσα σ' ένα αλφαριθμητικό. Η πρώτη παράμετρος είναι η αναζητούμενη αλληλουχία και η δεύτερη το αλφαριθμητικό μέσα στο οποίο θα αναζητηθεί.

Π.χ.

```
VAR := AT("c", "abcde")          // επιστρέφει 3
```

Η παραπάνω συνάρτηση επιστρέφει την τιμή 3, επειδή το c είναι ο τρίτος χαρακτήρας του αλφαριθμητικού. Αν δε βρεθεί η αλληλουχία, η συνάρτηση επιστρέφει την τιμή 0.

Η συνάρτηση ***RAT()*** κάνει την ίδια δουλειά, αλλά επιστρέφει τη θέση από το τέλος του αλφαριθμητικού αντί από την αρχή του.

Π.χ.

```
VAR := RAT("σου", "Γεια σου")          // επιστρέφει 3
```

Η συνάρτηση ***SUBSTR()*** εξάγει συγκεκριμένα τμήματα αλφαριθμητικών.

Π.χ.

```
VAR := SUBSTR("ΙΕΚ Φλώρινας", 1, 3)  // επιστρέφει "ΙΕΚ"
```

Η πρώτη παράμετρος δηλώνει το αλφαριθμητικό από το οποίο θέλουμε να εξάγουμε ένα τμήμα. Η δεύτερη δηλώνει τη θέση αρχής μέσα στο αλφαριθμητικό και η τρίτη δηλώνει το πλήθος των χαρακτήρων που θα εξαχθούν.

Αν δεν ορίζεται η τρίτη παράμετρος, τότε περιλαμβάνονται όλοι οι χαρακτήρες από τη θέση που ορίζει η δεύτερη παράμετρος και μετά. Αν η τρίτη παράμετρος έχει αρνητική τιμή, τότε η φορά λειτουργίας γίνεται από δεξιά προς τα αριστερά.

Π.χ.

VAR := SUBSTR("Αθήνα - 2004", 5, -2)

Η συνάρτηση ***LEFT()*** επιστρέφει ορισμένους χαρακτήρες από το αριστερό άκρο ενός αλφαριθμητικού και η συνάρτηση ***RIGHT()*** από το δεξιό άκρο.

Π.χ.

```
VAR := LEFT("Βασιλειάδης Γιώργος", 5)        // επιστρέφει "Βασιλ"
VAR := RIGHT("Βασιλειάδης Γιώργος ", 7)      // επιστρέφει "Γιώργος"
```

Η συνάρτηση ***ISALPHA()*** ελέγχει αν ο πρώτος από τα αριστερά χαρακτήρας ενός αλφαριθμητικού είναι αλφαβητικός (αγγλικό γράμμα) και τότε επιστρέφει λογική τιμή .T.

Π.χ.

```
ISALPHA("abc")          // επιστρέφει .T.
ISALPHA("123")          // επιστρέφει .F.
```

Η συνάρτηση ***ISDIGIT()*** ελέγχει αν ο πρώτος από τα αριστερά χαρακτήρας ενός αλφαριθμητικού είναι ψηφίο (0-9) και τότε επιστρέφει λογική τιμή .T.

Π.χ.

```
ISDIGIT("abc")          // επιστρέφει .F.
ISDIGIT("123")          // επιστρέφει .T.
```

Η συνάρτηση ***ISLOWER()*** ελέγχει αν ο πρώτος από τα αριστερά χαρακτήρας ενός αλφαριθμητικού είναι πεζό αγγλικό γράμμα και τότε επιστρέφει λογική τιμή .T.

Π.χ.

```
ISLOWER("abc")          // επιστρέφει .T.
ISLOWER("Abc")          // επιστρέφει .F.
```

Η συνάρτηση ***ISUPPER()*** ελέγχει αν ο πρώτος από τα αριστερά χαρακτήρας ενός αλφαριθμητικού είναι κεφαλαίο αγγλικό γράμμα και τότε επιστρέφει λογική τιμή .Τ.

Π.χ.

```
ISUPPER("abc")          // επιστρέφει .F.
ISUPPER("Abc")          // επιστρέφει .T.
```

Η συνάρτηση ***STRTRAN()*** αναζητά και αντικαθιστά χαρακτήρες μέσα σ' ένα αλφαριθμητικό.

Π.χ.

```
VAR := STRTRAN("MISSISSIPPI", "SS","CC")
       // επιστρέφει MICCICCIPPI
```

Η πρώτη παράμετρος είναι το αλφαριθμητικό στο οποίο θα κάνουμε την αναζήτηση, η δεύτερη παράμετρος είναι το αναζητούμενο αλφαριθμητικό και η τρίτη παράμετρος είναι το αλφαριθμητικό αντικατάστασης.

Στη συνάρτηση ***STUFF()***, η πρώτη παράμετρος είναι το αλφαριθμητικό που θα χειριστούμε, η δεύτερη είναι η θέση μέσα στο αλφαριθμητικό όπου θα γίνουν οι αλλαγές, η τρίτη είναι ο αριθμός των χαρακτήρων που θα επηρεαστούν από την αλλαγή και η τέταρτη είναι το νέο αλφαριθμητικό που θα παρεμβληθεί στο αρχικό αλφαριθμητικό.

Για να παρεμβάλουμε χαρακτήρες σ' ένα αλφαριθμητικό, χωρίς να διαγράψουμε κάποιον από τους υπάρχοντες χαρακτήρες, δίνουμε 0 στην τρίτη παράμετρο.

Π.χ.

```
VAR := STUFF("ABCDEF", 4, 0, "abc")    // επιστρέφει "ABCabcDEF"
VAR := STUFF("ABCDEF", 4, 3, "abc")    // επιστρέφει "ABCabc"
VAR := STUFF("ABCDEF", 4, 0, "")       // επιστρέφει "ABC"
```

Η συνάρτηση ***REPLICATE()*** επιστρέφει έναν χαρακτήρα όσες φορές θέλουμε.

Π.χ.

```
VAR := REPLICATE("A", 10)          // επιστρέφει "AAAAAAAAAA"
```

Η συνάρτηση ***LOWER()*** μετατρέπει τα γράμματα ενός αλφαριθμητικού σε πεζά.

Π.χ.

VAR := LOWER("PRESPES") *// επιστρέφει prespes*

Η συνάρτηση ***UPPER()*** μετατρέπει τα γράμματα ενός αλφαριθμητικού σε κεφαλαία.

Π.χ.

VAR := UPPER("radio") *// επιστρέφει RADIO*

Η συνάρτηση ***VAL()*** μετατρέπει ένα αλφαριθμητικό σε αριθμό. Η συνάρτηση αυτή υπολογίζει μια παράσταση χαρακτήρων μέχρι να συναντήσει μια δεύτερη υποδιαστολή ή έναν μη αριθμητικό χαρακτήρα.

Π.χ.

VAR := VAL("123.45") *// επιστρέφει 123.45*
VAR := VAL("A") *// επιστρέφει 0*
VAR := VAL("12B") *// επιστρέφει 12*

Η συνάρτηση ***ASC()*** επιστρέφει την ASCII τιμή του πρώτου από τα αριστερά χαρακτήρα ενός αλφαριθμητικού.

Π.χ.

VAR := ASC("abc") *// επιστρέφει 97*

Η συνάρτηση ***CHR()*** μετατρέπει μια τιμή ASCII στον αντίστοιχο χαρακτήρα.

Π.χ.

VAR := CHR(65) *// επιστρέφει A*

Αριθμητικές Πράξεις

Η συνάρτηση ***INT()*** μετατρέπει έναν πραγματικό (δεκαδικό) αριθμό σε ακέραιο, αποκόπτοντας τα δεκαδικά του ψηφία.

Π.χ.

VAR := INT(12.2) *// επιστρέφει 12*
VAR := INT(12.8) *// επιστρέφει 12*

Η συνάρτηση ***ROUND()*** στρογγυλοποιεί έναν αριθμό. Η πρώτη παράμετρος είναι ο αριθμός που θέλουμε να στρογγυλοποιήσουμε και η δεύτερη παράμετρος είναι ο αριθμός των δεκαδικών θέσεων που θα υπάρχουν στην επιστρεφόμενη τιμή.

Π.χ.

```
VAR := ROUND(12.567, 2)        // επιστρέφει 12.57
VAR := ROUND(12.564, 0)        // επιστρέφει 13
```

Η συνάρτηση ***MAX()*** δέχεται δύο αριθμούς σαν είσοδο και επιστρέφει τον μεγαλύτερο απ' αυτούς, ενώ η συνάρτηση ***MIN()*** δέχεται κι αυτή δύο αριθμούς σαν είσοδο και επιστρέφει τον μικρότερο απ' αυτούς.

Η εντολή ***SET DECIMALS TO*** καθορίζει τον αριθμό των δεκαδικών θέσεων που θα παρουσιάζονται στα αποτελέσματα των αριθμητικών πράξεων.

Π.χ.

```
VAR := 2/4
SET DECIMALS TO 2
? VAR                  // εμφανίζεται 0.50
SET DECIMALS TO 4
? VAR                  // εμφανίζεται 0.5000
```

Η εντολή ***SET FIXED ON*** ενεργοποιεί τη ρύθμιση SET DECIMALS, ενώ η εντολή SET FIXED OFF την απενεργοποιεί.

Αν σ' ένα αριθμητικό πεδίο εμφανιστούν αστεράκια (**) κατά τον χρόνο εκτέλεσης ενός προγράμματος, αυτό σημαίνει ότι έχουμε δηλώσει λιγότερες αριθμητικές θέσεις γι αυτό το πεδίο απ' ό,τι είναι η τιμή του (υπερχείλιση – overflow).

Επεξεργασία Ημερομηνιών

Οι συναρτήσεις ***DAY()***, ***MONTH()*** και ***YEAR()*** επιστρέφουν αντίστοιχα την αριθμητική τιμή της ημέρας, του μήνα και του έτους από μια μεταβλητή τύπου ημερομηνίας. Και στις τρεις συναρτήσεις, η παράμετρος είναι ημερομηνία και η επιστρεφόμενη τιμή είναι ακέραιος αριθμός.

Η συνάρτηση ***DOW()*** επιστρέφει έναν αριθμό που αντιστοιχεί στην ημέρα της εβδομάδας. Η παράμετρος της συνάρτησης είναι ημερομηνία και οι τιμές που επιστρέφει αρχίζουν από το 1 (Κυριακή) έως το 7 (Σάββατο).

Η συνάρτηση ***CDOW()*** μετατρέπει μια ημερομηνία σε αλφαριθμητικό που δείχνει την ημέρα της εβδομάδας, όπως "Sunday", "Monday" κ.ά.

Η συνάρτηση ***CMONTH()*** μετατρέπει μια ημερομηνία σε αλφαριθμητικό που δείχνει τον μήνα του έτους, όπως "January", "December" κ.ά.

Μπορούμε να κάνουμε και πράξεις με ημερομηνίες :

```
VAR := DATE() + 30      // 30 ημέρες από σήμερα
VAR := DATE() - 365     // ένα έτος πριν από σήμερα
```

Με την εντολή ***SET CENTURY ON*** ενεργοποιούμε τη χρήση του αιώνα στις ημερομηνίες, δηλ. θα πρέπει να δίνουμε το έτος ως 1997, ενώ με την εντολή SET CENTURY OFF δεν χρειάζεται να δίνουμε το 1997, αλλά μόνο το 97.

Η εντολή ***SET EPOCH TO*** ορίζει ένα σημείο έναρξης ημερομηνιών, από το οποίο υπολογίζονται όλες οι ημερομηνίες με διψήφιο αριθμό έτους. Το προεπιλεγμένο σημείο έναρξης ημερομηνιών είναι το έτος 1900. Δείτε το παρακάτω παράδειγμα για να καταλάβετε τι συμβαίνει :

```
SET EPOCH TO 1980       // υποστηρίζει διψήφια έτη από 1980-2079
? CTOD("11/25/92")      // θα εμφανίσει 11/25/1992
? CTOD("05/21/12")      // θα εμφανίσει 05/12/2012
```

Στο παραπάνω παράδειγμα ορίσαμε σαν αρχή ημερομηνιών το έτος 1980 και κάθε έτος από το 80 και μετά θα θεωρείται ότι ανήκει στον $20^{ο}$ αιώνα, ενώ κάθε έτος από το 80 και πριν θα θεωρείται ότι ανήκει στον $21^{ο}$ αιώνα.

Η εντολή ***SET DATE TO*** χρησιμοποιείται για τη μορφοποίηση της εξόδου ημερομηνιών, με προεπιλεγμένη μορφή την mm/dd/yy. Όλες οι μορφές είναι οι εξής :

```
AMERICAN mm/dd/yy
ANSI        yy.mm.dd
BRITISH     dd/mm/yy
FRENCH      dd/mm/yy
GERMAN      dd.mm.yy
ITALIAN     dd-mm-yy
JAPAN       yy/mm/dd
USA         mm-dd-yy
```

Π.χ., με την εντολή SET DATE TO FRENCH ορίζουμε τη μορφή ημερομηνίας σε dd/mm/yy. Μπορούμε να ορίσουμε και μια δική μας μορφή ημερομηνιών με την εντολή ***SET DATE FORMAT TO*** :

```
SET DATE FORMAT TO "yyyy-mm-dd"
```

Πράξεις με Πίνακες

Στον Clipper, κάθε στοιχείο ενός πίνακα μπορεί να είναι οποιουδήποτε τύπου. Για παράδειγμα, το πρώτο στοιχείο ενός πίνακα θα μπορούσε να είναι ένας άλλος πίνακας, το δεύτερο στοιχείο μια ημερομηνία και το τρίτο στοιχείο ένας αριθμός. Οι πίνακες στον Clipper ξεκινούν με δείκτη πρώτου στοιχείου το 1.

Οι πίνακες μπορούν να δηλωθούν με πολλούς τρόπους :

```
DECLARE var[10]
PUBLIC var[10]                  // καθολική μεταβλητή
LOCAL var[10]                   // τοπική μεταβλητή
LOCAL var := ARRAY(10, 20)      // πίνακας δύο διαοτάσεων
var := {1, 2, 3}                // δήλωση με απόδοση αρχικών τιμών
var := {1, 2, 3}, {4, 5, 6}, {"Γεια σας", .T., 0}     // πίνακας 3 X 3
```

Η εκχώρηση τιμών σ' ένα στοιχείο ενός μονοδιάστατου πίνακα γίνεται με την εισαγωγή της θέσης του στοιχείου μέσα σε αγκύλες.

Π.χ.

```
var[3] := "Γιάννης"          // τιμή στο 3ο στοιχείο του πίνακα
```

Η εκχώρηση τιμών σ' ένα στοιχείο πολυδιάστατου πίνακα γίνεται με την εισαγωγή των αριθμών των στοιχείων της γραμμής και της στήλης μέσα σε αγκύλες.

Π.χ.

```
var[1][1] := "Μαρία"         // 1η γραμμή, 1η στήλη
var[1][5] := 152             // 1η γραμμή, 5η στήλη
```

Η συνάρτηση ***AFILL()*** γεμίζει όλα ή κάποια στοιχεία ενός πίνακα με μία συγκεκριμένη τιμή :

```
var := ARRAY(10)                 // δήλωση πίνακα 10 θέσεων
var := AFILL(var, 0)             // όλα τα στοιχεία γίνονται 0
var := AFILL(var, "IEK", 5, 3)   // από το 5ο στοιχείο μέχρι και το 7ο
```

Ο Clipper έχει πολλές χρήσιμες συναρτήσεις για να εργαστούμε άνετα με τους πίνακες, οι οποίες φαίνονται στον παρακάτω πίνακα :

Συνάρτηση	Λειτουργία
AADD()	προσθέτει στοιχεία στον πίνακα
ACHOICE()	εκτελεί ένα κυλιόμενο μενού επιλογής
ACLONE()	αντιγράφει έναν πίνακα
ACOPY()	αντιγράφει στοιχεία ενός πίνακα
ADEL()	διαγράφει στοιχεία ενός πίνακα
ADIR()	καταχωρεί υποκατάλογο
AEVAL()	εκτελεί ένα κομμάτι κώδικα για κάθε στοιχείο
AFIELDS()	καταχωρεί πεδία της βάσης δεδομένων
AFILL()	δίνει μία σταθερή τιμή σ' όλα τα στοιχεία του πίνακα
AINS()	εισάγει ένα νέο στοιχείο με τιμή NIL
ARRAY()	δημιουργεί έναν πίνακα
ASCAN()	αναζητά ένα στοιχείο σ' έναν πίνακα
ASIZE()	αλλάζει τη διάσταση του πίνακα
ASORT()	ταξινομεί κάποια στοιχεία ενός πίνακα
ATAIL()	δίνει την τιμή του τελευταίου στοιχείου ενός πίνακα

Δείτε και τη χρήση των συναρτήσεων αυτών με παραδείγματα :

```
PIN := ARRAY(5)          // PIN := {NIL, NIL, NIL, NIL, NIL}
AFILL(PIN, 10)           // PIN := {10, 10, 10, 10, 10}
ADEL(PIN, 3)             // PIN := {10, 10, 10, 10}
AINS(PIN, 2)             // PIN := {10, NIL, 10, 10, 10}
ASIZE(PIN, 3)            // PIN := {10, NIL, 10}
AADD(PIN, 5)             // PIN := {10, NIL, 10, 5}
AFILL(PIN[2], 2)         // PIN := {10, 2, 10, 5}
ADEL(PIN, 1)             // PIN := {2, 10, 5}
```

Οι Μακροεντολές στον Clipper

Προσέξτε το παρακάτω παράδειγμα :

```
VAR := "DATE()"
? VAR                    // θα τυπωθεί η λέξη DATE()
? &VAR                   // θα τυπωθεί η σημερινή ημερομηνία
```

Η πρώτη γραμμή αποθηκεύει το όνομα DATE() στη μεταβλητή VAR, η δεύτερη γραμμή εμφανίζει τα περιεχόμενα του αλφαριθμητικού και η τρίτη χρησιμοποιεί τον τελεστή μακροεντολής & για να υπολογίσει τα περιεχόμενα της μεταβλητής VAR. Το αποτέλεσμα είναι η παρουσίαση της σημερινής ημερομηνίας.

΄Ελεγχος των Περιεχομένων των Μεταβλητών

Η συνάρτηση ***EMPTY()*** ελέγχει μια μεταβλητή οποιουδήποτε τύπου για να διαπιστώσει αν είναι κενή. Μια αλφαριθμητική μεταβλητή θεωρείται κενή αν περιέχει κενά διαστήματα, μια αριθμητική μεταβλητή θεωρείται κενή αν είναι ίση με 0 και μια μεταβλητή ημερομηνίας θεωρείται κενή αν δεν εισαχθεί σ' αυτήν κάποια τιμή ημερομηνίας.

Η συνάρτηση ***TYPE()*** ελέγχει τον τύπο δεδομένων μιας μεταβλητής και μπορεί να χρησιμοποιηθεί μόνο σε καθολικές (public) και αποκλειστικές (private) μεταβλητές. Η συνάρτηση ***VALTYPE()*** υπολογίζει τον τύπο των τοπικών (local) και των στατικών (static) μεταβλητών καθώς και των συναρτήσεων που ορίζονται από το χρήστη.

Η Συνάρτηση TRANSFORM()

Η συνάρτηση TRANSFORM() μοιάζει πολύ στη λειτουργία της με τη δήλωση PICTURE. Σαν πρώτη παράμετρο στη συνάρτηση TRANSFORM() μεταβιβάζουμε την τιμή ή την παράσταση που θα μετασχηματιστεί και σαν δεύτερη τον τρόπο μορφοποίησής της, δηλ. το πρότυπο εικόνας.

Π.χ.

```
? TRANSFORM("abc", "@!")          // εμφανίζει ABC
? TRANSFORM(71628, "$99,999")     // εμφανίζει $71,628
```

Χειρισμός του Πληκτρολογίου

Ό,τι πληκτρολογούμε εισάγεται σε μια περιοχή προσωρινής αποθήκευσης. Η περιοχή αυτή περιέχει ένα αλφαριθμητικό καθορισμένου μήκους από κωδικούς χαρακτήρων. Η εντολή ***SET TYPEAHEAD*** χρησιμοποιείται για τον καθορισμό του μεγέθους της περιοχής προσωρινής αποθήκευσης πληκτρολογήσεων, με ελάχιστη τιμή 16 και μέγιστη 4096 χαρακτήρες.

Οι χαρακτήρες απομακρύνονται από εκεί καθώς τους επεξεργάζεται ο Clipper. Η συνάρτηση ***LASTKEY()*** μάς πληροφορεί για την τιμή του πλήκτρου που πατήθηκε τελευταίο. Δείτε το επόμενο παράδειγμα :

```
@ 10, 10 SAY "Πόλη" GET City
READ
IF LASTKEY() == 27              // η ASCII τιμή του ESC
      <εντολές>
ENDIF
```

Η πληκτρολόγηση απομακρύνεται από την περιοχή προσωρινής αποθήκευσης πληκτρολογίου όταν κληθεί η συνάρτηση LASTKEY(). Υπάρχει και η συνάρτηση ***NEXTKEY()***, που μοιάζει με τη LASTKEY(), αλλά δεν απομακρύνει την πληκτρολόγηση. Η εντολή ***CLEAR TYPEAHEAD*** καθαρίζει την περιοχή προσωρινής αποθήκευσης πληκτρολογίου.

Η συνάρτηση ***INKEY()*** διαγράφει πληκτρολογήσεις από την περιοχή προσωρινής αποθήκευσης πληκτρολογίου, αλλά μπορεί να δεχθεί και μια πρόσθετη παράμετρο που προκαλεί αναμονή για ένα συγκεκριμένο χρονικό διάστημα ή μέχρι να πατηθεί κάποιο πλήκτρο.

Π.χ.

```
@ 24, 0  SAY "Γεια σου"
INKEY(0)            // περιμένει μέχρι να πατηθεί κάποιο πλήκτρο

@ 24, 0  SAY "Το κατάστημα κλείνει σε 15 δευτερόλεπτα"
INKEY(15)           // περιμένει 15 δευτερόλεπτα
```

Η συνάρτηση INKEY() χωρίς παραμέτρους ελέγχει την περιοχή προσωρινής αποθήκευσης πληκτρολογίου για τυχόν πληκτρολογήσεις. Αν υπάρχουν πληκτρολογήσεις, επιστρέφει τις αριθμητικές τιμές των πλήκτρων· διαφορετικά επιστρέφει μηδέν.

Αποθήκευση και Αποκατάσταση της Οθόνης

Στον Clipper μπορούμε να αποθηκεύουμε και να αποκαθιστούμε οθόνες με τις εντολές ***SAVE SCREEN*** και ***RESTORE SCREEN***. Μπορούμε να χρησιμοποιήσουμε μια μεταβλητή όπου θα αποθηκεύσουμε όλη την οθόνη και βέβαια μπορούμε να αποθηκεύσουμε σε ξεχωριστές μεταβλητές, περισσότερες από μία οθόνες. Δείτε τις παρακάτω εντολές :

```
SAVE SCREEN  TO  l_buffer
        <εντολές>
RESTORE SCREEN  FROM  l_buffer
```

Υπάρχουν και οι συναρτήσεις ***SAVESCREEN()*** και ***RESTSCREEN()*** που μπορούν να κάνουν περισσότερα. Μ' αυτές μπορούμε να αποθηκεύσουμε ένα τμήμα μόνο της οθόνης, δίνοντας τις συντεταγμένες των τεσσάρων κορυφών του. Δείτε τις παρακάτω εντολές :

```
l_buffer := SAVESCREEN(1, 1, 5, 10)
        <εντολές>
RESTSCREEN(1, 1, 5, 10, l_buffer)
```

Με τις παραπάνω εντολές, το τμήμα της οθόνης από τη γραμμή 1, στήλη 1 μέχρι τη γραμμή 5, στήλη 10 αποθηκεύεται στη μεταβλητή l_buffer και αργότερα το ίδιο τμήμα αποκαθίσταται. Πρέπει να προσέχουμε να αποκαθιστούμε μια οθόνη στο ίδιο μέγεθος μ' αυτήν που αποθηκεύσαμε.

Η συνάρτηση ***SCROLL()*** επιτρέπει την κύλιση ενός συγκεκριμένου τμήματος της οθόνης, δηλ. τη μετακίνηση των γραμμών προς τα πάνω ή προς τα κάτω ή ακόμη και το καθάρισμα του τμήματος, ανάλογα με τις παραμέτρους που θα μεταβιβαστούν στη συνάρτηση. Η ίδια συνάρτηση χωρίς παραμέτρους καθαρίζει ολόκληρη την οθόνη.

Παραγωγή ΄Ηχου

Ο χρήστης μπορεί να ειδοποιηθεί με δύο τρόπους για την ολοκλήρωση μιας διεργασίας ή για κάποιο σφάλμα. Ο ένας είναι ο εξής :

```
?? CHR(7)          // παράγει βόμβο
```

και ο άλλος είναι η χρήση της συνάρτησης ***TONE()***, με την οποία μπορούμε να καθορίσουμε τη συχνότητα και τη διάρκεια του ήχου.

Π.χ.

```
TONE(100, 3)       // παράγει έναν τόνο στη συχνότητα 100 Hz
                   // για 3/18 του δευτερολέπτου
```

Ρυθμίσεις Εισαγωγής Δεδομένων

Με τη συνάρτηση ***READEXIT()*** καθορίζουμε αν τα πλήκτρα ↑ και ↓ θα χρησιμοποιούνται για την έξοδο από τις εντολές READ. Για να ενεργοποιήσουμε τα πλήκτρα βελών δίνουμε READEXIT(.T.).

Η συνάρτηση ***READINSERT()*** χρησιμοποιείται για την ενεργοποίηση ή απενεργοποίηση του πλήκτρου INS. Για να ισχύει το πλήκτρο INS και να μπορούμε να κάνουμε παρεμβολή σε μια εντολή READ, δίνουμε READINSERT(.T.).

Με την εντολή ***SET CONFIRM ON*** πρέπει να πατάμε το πλήκτρο <enter> για κάθε πεδίο μιας εντολής READ, ενώ με την εντολή SET CONFIRM OFF δεν χρειάζεται να πατάμε το <enter> όταν έχει γεμίσει το πεδίο μιας εντολής GET. Αυτό είναι ιδιαίτερα χρήσιμο για πεδία ενός χαρακτήρα.

Η εντολή ***SET ESCAPE ON/FF*** ενεργοποιεί ή απενεργοποιεί τη δυνατότητα διαφυγής από μια εντολή READ με το πάτημα του πλήκτρου ESC.

Η εντολή ***SET BELL ON*** ενεργοποιεί τον βομβητή αν ο χρήστης βρίσκεται στον τελευταίο χαρακτήρα ενός πεδίου GET ή αν εισάγει έναν μη αποδεκτό χαρακτήρα, π.χ. εισάγει χαρακτήρα σε αριθμητικό πεδίο. Ο βομβητής απενεργοποιείται με την εντολή SET BELL OFF.

Με την εντολή ***SET DELIMITERS TO*** **":."** αναγκάζουμε την εντολή GET να εμφανίζει μία άνω και κάτω τελεία πριν από κάθε πεδίο και μία τελεία μετά.

Η εντολή ***SET INTENSITY ON/OFF*** ενεργοποιεί και απενεργοποιεί αντίστοιχα τον φωτισμό των πεδίων GET.

Η εντολή ***SET SCOREBOARD ON/OFF*** ενεργοποιεί και απενεργοποιεί αντίστοιχα την παρουσίαση πληροφοριών σχετικά με τις READ και MEMOEDIT στην πρώτη γραμμή της οθόνης. Όταν είναι ON, στην πάνω

δεξιά γωνία της οθόνης εμφανίζονται διάφορα μηνύματα σφαλμάτων, μηνύματα απόρριψης καθώς αν το πλήκτρο INS είναι ενεργό.

Το Χρώμα της Οθόνης

Ξέρουμε ότι με την εντολή SET COLOR TO ή με τη συνάρτηση SETCOLOR() μπορούμε να αλλάξουμε τα χρώματα της οθόνης. Μπορούμε, όμως, να αποθηκεύσουμε έναν συνδυασμό χρωμάτων σε μια μεταβλητή, να κάνουμε αλλαγές στα χρώματα και να αποκαταστήσουμε τα αρχικά χρώματα ως εξής :

```
l_holdcol := SETCOLOR("W/N, BG+/B, , ,W/N")
        <εντολές>
SETCOLOR(l_holdcol)
```

Η πρώτη γραμμή του παραπάνω παραδείγματος καλεί τη συνάρτηση SETCOLOR() για να αλλάξει το τρέχον χρώμα. Η SETCOLOR() αλλάζει το χρώμα και επιστρέφει το αρχικό χρώμα στη μεταβλητή l_holdcol. Μετά από μερικές εντολές, το πρόγραμμα καλεί και πάλι τη συνάρτηση SETCOLOR() με παράμετρο τη μεταβλητή l_holdcol για να επαναφέρει το αρχικό χρώμα.

΄Ελεγχος του Δρομέα

Η συνάρτηση ***SETCURSOR()*** μάς επιτρέπει να αλλάζουμε το σχήμα του δρομέα ως εξής :

0	Αόρατος
1	Υπογράμμιση
2	Κάτω μισό ορθογώνιο
3	Ορθογώνιο
4	Πάνω μισό ορθογώνιο

Τα Πεδία Σημειώσεων (MEMO)

Ο Clipper έχει και έναν ειδικό τύπο πεδίων δεδομένων, τα πεδία σημειώσεων (memo). Όπως ήδη ξέρουμε, όλα τα πεδία μιας βάσης δεδομένων αποθηκεύονται σ' ένα αρχείο με επέκταση .DBF. Όταν προσθέτουμε ένα πεδίο σημειώσεων σε μια βάση δεδομένων, δημιουργείται ένα ξεχωριστό αρχείο με το ίδιο όνομα, αλλά με επέκταση ***.DBT*** *(Data Base Text)*.

Η κάθε εγγραφή τύπου memo στο αρχείο .DBF περιέχει την αντίστοιχη διεύθυνση του αρχείου .DBT όπου είναι αποθηκευμένες οι σημειώσεις για τη συγκεκριμένη εγγραφή.

Σε μια μεγάλη βάση δεδομένων, το μέγεθος του αρχείου .DBT μπορεί να γίνει τεράστιο και αν αντιγράψουμε το αρχείο .DBF, δεν πρέπει να ξεχάσουμε να αντιγράψουμε και το αρχείο .DBT.

Επειδή τα πεδία σημειώσεων είναι συνήθως κενά στο μεγαλύτερο μέρος τους και καταλαμβάνουν μεγάλο χώρο στον δίσκο, θα πρέπει να προσέξουμε πολύ πριν τα δηλώσουμε στον σχεδιασμό μιας βάσης δεδομένων και να σκεφτούμε τη χρησιμοποίηση ενός αλφαριθμητικού πεδίου της βάσης δεδομένων για να κρατάμε σημειώσεις μικρού μεγέθους και αν παρουσιαστεί ανάγκη για μεγαλύτερες σημειώσεις, τότε να χρησιμοποιήσουμε το πεδίο memo.

Ο Clipper διαθέτει έναν μικρό επεξεργαστή κειμένου με όνομα ***MEMOEDIT()*** για να διορθώνουμε τα πεδία σημειώσεων. Για ένα πεδίο σημειώσεων με όνομα NOTES, η εντολή αυτή μπορεί να κληθεί ως εξής :

MEMOEDIT(NOTES, 2, 2, 20, 77)

΄Ετσι δημιουργείται ένα παράθυρο στην οθόνη στις θέσεις 2, 2 και 20, 77, όπου μπορούμε να κινηθούμε με τα βελάκια και να κάνουμε ό,τι καταχωρήσεις και διορθώσεις θέλουμε.

Με τη συνάρτηση MEMOREAD() μπορούμε να αποθηκεύσουμε τα περιεχόμενα ενός αρχείου σ' ένα string, με τη MEMOWRIT() να γράψουμε ένα πεδίο memo στον δίσκο, με τη συνάρτηση MEMOLINE() μπορούμε να παρουσιάζουμε μια γραμμή κειμένου ενός πεδίου σημειώσεων και με την MLCOUNT() να μετρήσουμε τον αριθμό γραμμών σ' ένα πεδίο memo.

Τα Τμήματα Κώδικα

΄Ενα ***τμήμα κώδικα*** *(code block)* μπορεί να θεωρηθεί σαν ένας τύπος δεδομένων που μοιάζει με τους άλλους τύπους δεδομένων. Ο Clipper αντιμετωπίζει τα τμήματα κώδικα όπως τους άλλους τύπους δεδομένων, αλλά τα εκτελεί διαφορετικά. Τα τμήματα κώδικα μπορούν να αποθηκεύονται σε μεταβλητές και να αντιγράφονται από τη μια μεταβλητή στην άλλη και επίσης μπορούν να παίρνουν όνομα και να μεταβιβάζονται σαν παράμετροι.

Το τμήμα κώδικα μοιάζει πολύ μ' ένα αλφαριθμητικό που μεταγλωττίζεται και εκτελείται με τον τελεστή μακροεντολής &. Η διαφορά είναι ότι το τμήμα κώδικα μεταγλωττίζεται κατά τη διάρκεια της μεταγλώττισης, ενώ οι μακροεντολές μεταγλωττίζονται κάθε φορά που συναντιούνται μέσα στο πρόγραμμα.

Τα τμήματα κώδικα υπολογίζονται με τη βοήθεια των συναρτήσεων EVAL(), AEVAL() ή DBEVAL(), περίπου με τον ίδιο τρόπο που χρησιμοποιείται ο τελεστής μακροεντολής & για τον υπολογισμό αλφαριθμητικών μακροεντολών.

Δείτε τις παρακάτω εντολές :

```
VAR := "DATE() - 27"
? &VAR        // εμφανίζει την ημερομηνία πριν από 27 ημέρες
```

Η παράσταση αποθηκεύεται στη μεταβλητή VAR και μεταγλωττίζεται και εκτελείται κατά το χρόνο εκτέλεσης με τον τελεστή μακροεντολής &. Το ίδιο αποτέλεσμα θα μπορούσε να προκύψει και με τη χρήση ενός τμήματος κώδικα, ως εξής :

```
VAR := {|| DATE() - 27}
? EVAL(VAR)        // εμφανίζει την ημερομηνία πριν από 27 ημέρες
```

Κάθε τμήμα κώδικα πρέπει να βρίσκεται μέσα σε άγκιστρα { }. Τα σύμβολα || δείχνουν σ' ένα τμήμα κώδικα και περικλείουν τις παραμέτρους που χρησιμοποιούνται σ' αυτά. Στην πρώτη γραμμή του προηγούμενου παραδείγματος, εκχωρείται το τμήμα κώδικα (DATE() - 27) στη μεταβλητή VAR. Η συνάρτηση EVAL() εκτελεί το τμήμα κώδικα με όνομα VAR.

Δείτε και το παρακάτω παράδειγμα :

```
VAR := {|x| DATE() - x}
? EVAL(VAR, 27)   // εμφανίζει την ημερομηνία πριν από 27 ημέρες
```

Η παράμετρος x τοποθετείται ανάμεσα στα σύμβολα || και η τιμή που θα εκχωρηθεί στην x μεταβιβάζεται σαν δεύτερη παράμετρος στη συνάρτηση EVAL(). Η συνάρτηση AEVAL() επεξεργάζεται τα περιεχόμενα όλων των στοιχείων ενός πίνακα χρησιμοποιώντας ένα τμήμα κώδικα. Η συνάρτηση DBEVAL() επεξεργάζεται τα περιεχόμενα όλων των εγγραφών μιας βάσης δεδομένων χρησιμοποιώντας ένα τμήμα κώδικα.

Οι Δηλώσεις MEMVAR και FIELD

Η πρόταση εντολών MEMVAR εφαρμόζεται στον Clipper 5.01 για να αναλύσει αναφορές χωρίς ψευδώνυμο, θεωρώντας ότι η αναφορά είναι μεταβλητή μνήμης. Για να δημιουργήσουμε μια λίστα μεταβλητών μνήμης στον Clipper 5.01, θα πρέπει να χρησιμοποιήσουμε τον εξής κώδικα :

MEMVAR *<μεταβλητήΜνήμης> [, <μεταβλητήΜνήμης>...]*

Στην πραγματικότητα δεν δημιουργείται καμία μεταβλητή μνήμης, απλώς ο Clipper λαμβάνει υπόψη του ότι κάθε αναφορά στο παραπάνω όνομα είναι αναφορά σε μια μεταβλητή μνήμης. Ακόμη, κάθε δήλωση MEMVAR πρέπει να εμφανίζεται πριν από οποιονδήποτε άλλο εκτελέσιμο κώδικα του Clipper· διαφορετικά θα εμφανιστεί ένα μήνυμα σφάλματος κατά τη μεταγλώττιση.

Δείτε το παρακάτω παράδειγμα :

```
...
MEMVAR  account
account := 1
USE  Clients  NEW

DO  WHILE .NOT. EOF()
        ? Clients->account, account++
        SKIP
ENDDO
...
```

Σ' αυτό το πρόγραμμα είπαμε στον μεταγλωττιστή ότι θα υπάρξει τουλάχιστον μια σαφής μεταβλητή μνήμης με το όνομα account. Πεδίο με το όνομα account υπάρχει, όμως, και στη βάση δεδομένων Clients. Το πεδίο account, όμως, έχει έναν σαφή δείκτη ψευδωνύμου στην περιοχή εργασίας Clients.

Η μεταβλητή μνήμης account αυξάνεται μετά τη χρήση της και ο μεταγλωττιστής, εξαιτίας της δήλωσης MEMVAR, αντιμετωπίζει αυτό το σύμβολο σαν να ήταν M->account++ ή MEMVAR->account++.

Αν χρησιμοποιούμε την εντολή MEMVAR για να δηλώσουμε στον μεταγλωττιστή τα ονόματα των μεταβλητών μνήμης προκαταβολικά, (και αν χρησιμοποιούμε τον διακόπτη μεταγλώττισης /W) τα πιθανά προειδοποιητικά μηνύματα και οι εσωτερικές ασάφειες θα περιοριστούν.

Η πρόταση εντολών FIELD χρησιμοποιείται για να δηλώσει στην εφαρμογή τα ονόματα των πεδίων της βάσης δεδομένων. Η σύνταξή της είναι παρόμοια μ' αυτήν της εντολής MEMVAR :

FIELD *<όνομα πεδίου> [, <όνομα πεδίου....]*
*[**IN**<βάση δεδομένων>]*

Μ' αυτόν τον τρόπο απλώς γίνεται υπόθεση ότι οι αναφορές χωρίς ψευδώνυμο στα ονόματα που περιλαμβάνονται στη λίστα είναι αναφορές σε πεδία της βάσης δεδομένων. Η εντολή FIELD πρέπει να εμφανιστεί σε μια υπορουτίνα, διαδικασία ή συνάρτηση πριν από οποιονδήποτε εκτελέσιμο κώδικα.

Δείτε το παρακάτω παράδειγμα :

```
...
FIELD files, sizes, subdrive, per_used IN Notap
CLS
USE  Notap NEW
USE  Clients NEW
DO  WHILE  Clients->(!EOF())
        ? Clients->status, Clients->account
        IF Notap->(!EOF())
                ? "   ", files, sizes, LEFT(subdrive, 25), per_used
                SKIP ALIAS Notap
        ENDIF
        SKIP
        IF  EMPTY(ROW() % 22)
                WAIT
                CLS
        ENDIF
ENDDO
...
```

Αυτό το παράδειγμα είναι παρόμοιο με το προηγούμενο που είδαμε με τη δήλωση MEMVAR, εκτός από το ότι εδώ δουλεύουμε με πεδία σε βάσεις δεδομένων και όχι με μεταβλητές μνήμης.

Δηλώνοντας στον μεταγλωττιστή προκαταβολικά ποια σύμβολα χωρίς ψευδώνυμο σε μια εφαρμογή είναι πεδία και ποια είναι μεταβλητές μνήμης, αποτρέπουμε τις πιθανές ασάφειες. Στην περίπτωση της δήλωσης FIELDS, τα πεδία δεν δηλώνονται απλώς, αλλά συνδέονται με το ψευδώνυμο της περιοχής εργασίας.

Για τη χρήση της εντολής FIELD και οι τρεις εκδοχές των παρακάτω κανόνων είναι πανομοιότυπες :

Με δήλωση FIELD	Με δείκτη ψευδωνύμου FIELD	Με δείκτη περιοχής ALIAS()
FIELD files, sizes; subdrive, per_used	USE Notap NEW; SELECT Notap	USE Notap NEW
files	Field->files	Notap->files
sizes	Field->sizes	Notap->sizes
subdrive	Field->subdrive	Notap->subdrive
per_used	Field->per_used	Notap->per_used

Η Δήλωση EXTERNAL

Η εντολή αυτή χρησιμοποιείται για να πληροφορεί τον μεταγλωττιστή προκαταβολικά για κάθε εξωτερική υπορουτίνα που μπορεί να χρειαστεί να συνδεθεί μέσα στην εφαρμογή. Για παράδειγμα, μερικές φορές το όνομα της συνάρτησης ή της διαδικασίας μπορεί να αποτελεί μέρος μιας επέκτασης μακροεντολής ή αλφαριθμητικού, όπως φαίνεται παρακάτω :

```
FOR x := 1 TO 3
        ext := ALTRIM(STR(x))
        DO Proc&ext.
NEXT x
```

ή

```
branch := "Listchoice()"
Op (&branch.)
```

όπου Op() είναι μια ψευδο-διαδικασία.

Και στις δύο περιπτώσεις, οι διαδικασίες ή συναρτήσεις που μπορεί να κληθούν κατά την εκτέλεση αυτών των παραδειγμάτων, μπορεί να έχουν σαν αποτέλεσμα ένα σφάλμα χρόνου εκτέλεσης που θα δηλώνει ότι «λείπει η εξωτερική αναφορά» («missing external»).

Στην ουσία, ο συνδετικός διορθωτής δεν γνωρίζει ποιες υπορουτίνες μπορεί να εκτελεσθούν - δηλαδή οι Proc1, Proc2, Proc3 και Listchoice(). Όπως βλέπουμε στα παραπάνω παραδείγματα, όλες αυτές οι υπορουτίνες είναι μέσα σε επεκτάσεις μακροεντολών ή ορισμούς αλφαριθμητικών.

Και αφού, όπως είναι φυσικό, ούτε ο μεταγλωττιστής ούτε ο συνδετικός διορθωτής αντιλαμβάνονται τι μπορεί να εκτελεστεί, θα πρέπει να τους πληροφορήσουμε προκαταβολικά ότι είναι πιθανό να χρειαστούμε αυτές τις ρουτίνες και ότι θα πρέπει να τις έχουν υπόψη τους.

Για να γίνει αυτό, χρειάζεται να προσθέσουμε την πρόταση εντολών EXTERNAL με το όνομα κάθε ρουτίνας που πρόκειται να χρησιμοποιηθεί. Με βάση τα παραπάνω παραδείγματα, π.χ., θα πρέπει να προσθέσουμε τα εξής :

```
EXTERNAL Proc1, Proc2, Proc3
FOR x := 1 TO 3
        ext := ALTRIM(STR(x))
        DO Proc&ext.
NEXT
```

ή

```
EXTERNAL Listchoice
branch := "Listchoice"
Op(&branch.)
```

όπου Op() είναι μια λειτουργική ψευδο-διαδικασία.

Η πρόταση EXTERNAL πρέπει και αυτή να εμφανιστεί πριν από κάθε εκτελέσιμη πρόταση εντολών και πριν από κάθε εντολή PARAMETERS.

Η Συνάρτηση ACHOICE()

Μπορούμε να χρησιμοποιούμε τη συνάρτηση ACHOICE() για να δημιουργούμε ένα μενού φωτεινής μπάρας με πολλές δυνατότητες. Το πλεονέκτημα της συνάρτησης ACHOICE() σε σχέση με την εναλλακτική λύση PROMPT/MENU TO, είναι ότι μπορεί να υπάρχουν περισσότερες επιλογές απ' αυτές που εμφανίζονται κάθε φορά στην οθόνη. Η ACHOICE() κυλά τις επιλογές στο παράθυρο ώστε να εμφανίζονται όλες.

Η γενική σύνταξη της συνάρτησης είναι ως εξής :

ACHOICE(*<γραμμή1>, <στήλη1>, <γραμμή2>, <στήλη2>, <πίνακας1> [, <πίνακας2> [, <συνάρτηση> [, <αρχή> [, <σχετ>]]]]*)

Το μενού φωτεινής μπάρας θα εμφανιστεί στο παράθυρο που ορίζεται από τις συντεταγμένες <γραμμή1>, <στήλη1> έως και <γραμμή2>, <στήλη2>. Όπως είπαμε πρωτύτερα, αν υπάρχουν περισσότερες επιλογές απ' όσες χωρούν στην περιοχή, γίνεται κύλιση (scrolling) των επιλογών. Η επιλογή που κάνει ο χρήστης πατώντας το πλήκτρο <enter> είναι η τιμή επιστροφής της συνάρτησης, όπως συμβαίνει και με τις εντολές PROMPT/MENU TO.

Όλες οι επιλογές τοποθετούνται ως στοιχεία στον πίνακα <πίνακας1>. Μπορούμε παράλληλα να προσθέσουμε κι έναν δεύτερο πίνακα <πίνακας2>, ο οποίος θα περιέχει λογικές παραστάσεις. Ανάλογα με τα περιεχόμενα του δεύτερου πίνακα, η αντίστοιχη επιλογή θα είναι ή όχι διαθέσιμη. Με τιμή .T., η επιλογή θα είναι διαθέσιμη, ενώ με τιμή .F. η επιλογή θα εμφανίζεται, αλλά δεν θα μπορούμε να την επιλέξουμε.

Κάθε λειτουργία του μενού μπορεί να μεταβιβάζει μια παράμετρο στη <συνάρτηση>. Η <συνάρτηση> μπορεί να επιστρέφει μια τιμή, η οποία μπορεί να χρησιμοποιηθεί για τη συνέχιση της διαδικασίας των επιλογών.

Με την επιλογή <αρχή>, η φωτεινή μπάρα μεταφέρεται αμέσως στην αντίστοιχη επιλογή του μενού. Με την προαιρετική παράμετρο <σχετ> μπορούμε να καθορίσουμε τη θέση της πρώτης γραμμής του μενού σε σχέση με την πάνω γραμμή του παραθύρου.

Δείτε ένα παράδειγμα χρήσης της συνάρτησης ACHOICE() :

```
fields := {"Είσοδος", "Ανάκληση Βάσης Δεδομένων", "Αποθήκευση;
      Δεδομένων", "Έξοδος"}
choice := {.T., .T., .F., .T.}
position := ACHOICE(5, 18, 22, 50, fields, choice)
```

Με τις παραπάνω εντολές, δημιουργείται ένα μενού από τη θέση 5, 18 έως τη θέση 22, 50 στο οποίο εμφανίζονται οι επιλογές του πίνακα fields και από τις οποίες δεν μπορούμε να επιλέξουμε την "Αποθήκευση Δεδομένων".