ΚΕΦΑΛΑΙΟ 3
ΣΧΕΔΙΑΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΤΟΝ CLIPPER

Προγραμματισμός Κατά Τμήματα (Modular)

Ένα από τα βασικότερα σημεία του προγραμματισμού είναι η οργάνωση. Πρέπει να ομαδοποιούμε τις λειτουργίες της εφαρμογής μας σε επιμέρους τμήματα. Στον προγραμματισμό, ο κώδικας που διασπάται σε ενότητες ονομάζεται *modular (τμηματοποιημένος)*.

Κάθε *ενότητα*, ή *module*, εκτελεί μια συγκεκριμένη ενέργεια και είναι ανεξάρτητη από τις υπόλοιπες. Έτσι, το γράψιμο μιας εφαρμογής είναι στην πραγματικότητα η δημιουργία μιας συλλογής από μικρότερα προγράμματα, όπου το καθένα μπορεί να ενεργοποιήσει το άλλο, να περάσει πληροφορίες από και προς τα άλλα και να χρησιμοποιήσει από κοινού μια βάση δεδομένων.

Η Αξία των Modules

Με τα modules, ο εντοπισμός των σφαλμάτων του προγράμματος είναι πολύ ευκολότερος και αυτό γιατί ελέγχουμε μικρές ενότητες κάθε φορά. Ακόμη, είναι πιο εύκολη η κατανόησή τους. Η κατασκευή μακροσκελών τμημάτων κώδικα είναι συνήθως πολύ χρονοβόρα και δύσκολα κατανοητή.

Τα Ονόματα των Μεταβλητών

Για τις μεταβλητές μας θα πρέπει να χρησιμοποιούμε ονόματα που να δηλώνουν την τιμή στην οποία αναφέρονται. Για παράδειγμα, για το επώνυμο ενός πελάτη, θα μπορούσαμε να έχουμε τη μεταβλητή lastname_client και για τη διεύθυνσή του τη μεταβλητή address_client και όχι ονόματα όπως a, b, sd κοκ. Έτσι, είναι πολύ εύκολο και για μας, αλλά και για κάποιον τρίτο, να καταλάβει αμέσως πού αναφέρεται η κάθε μεταβλητή.

Ακόμη, θα πρέπει να ξεχωρίζουμε τα ονόματα των μεταβλητών που αναφέρονται στα πεδία των αρχείων μιας βάσης δεδομένων από τα ονόματα των μεταβλητών του κυρίου προγράμματος. Θα μπορούσαμε, για παράδειγμα, να χρησιμοποιούμε έναν πρόθεμα F_ για τα ονόματα των πεδίων των αρχείων και ένα πρόθεμα M_ για τα ονόματα των κανονικών μεταβλητών.

Τα Σχόλια στον Clipper

Όπως ήδη ξέρουμε, τα σχόλια είναι μη-εκτελέσιμες γραμμές αλφαριθμητικών, που χρησιμεύουν στην περιγραφή και στην επεξήγηση του προγράμματος. Τα σχόλια είναι απαραίτητα για τη σωστή τεκμηρίωση του προγράμματος.

Ο Clipper δέχεται τρεις τύπους σχολίων :

/* ... */

Ολόκληρος ο κώδικας που ακολουθεί το /* αγνοείται από τον μεταγλωττιστή έως ότου συναντηθεί το αντίστοιχο */. Είναι χρήσιμο για τον σχολιασμό πολλών συνεχόμενων γραμμών.

*****

Οι γραμμές που έχουν έναν αστερίσκο στην πρώτη στήλη τους θεωρούνται σχόλια.

// ή **&&**

Τα σύμβολα // ή && μπορούν να εμφανιστούν οπουδήποτε σε μια γραμμή και όλο το κείμενο μέχρι το τέλος της γραμμής θεωρείται σχόλιο.

Οι Μεταβλητές στον Clipper

Υπάρχουν τέσσερις βασικοί τύποι μεταβλητών στον Clipper : Public, Private, Local και Static. Όλες οι δηλώσεις των μεταβλητών πρέπει να γίνονται στην αρχή μιας υπορουτίνας (procedure ή function), δηλ. πριν από κάθε εκτελέσιμη εντολή. Οι μη δηλωμένες μεταβλητές που εμφανίζονται στο εσωτερικό μιας υπορουτίνας, χαρακτηρίζονται αυτόματα ως Private.

Οι Μεταβλητές Local

Οι μεταβλητές *local (τοπικές)* θα πρέπει να δηλώνονται ρητά στο πρόγραμμα με τη δήλωση LOCAL. Δημιουργούνται εκ νέου κάθε φορά που καλούμε την υπορουτίνα που τις περιέχει και εξακολουθούν να υφίστανται και να είναι προσπελάσιμες μέχρι την ολοκλήρωση της υπορουτίνας, οπότε και καταστρέφονται (αποδεσμεύεται ο αντίστοιχος χώρος στη μνήμη).

Οι μεταβλητές local μπορούν να προσπελαστούν μόνο από την υπορουτίνα στην οποία δημιουργήθηκαν, ενώ οι υπορουτίνες που καλούνται απ' αυτήν δεν μπορούν να τις προσπελάσουν.

Δείτε το επόμενο παράδειγμα :

```
PROCEDURE My_Proc
        LOCAL  Sum,  Total := 5
        /* εντολές */
RETURN
```

Οι μεταβλητές Sum και Total ορίζονται ως local και η Total παίρνει αρχική τιμή ίση με 5. Στη Sum εκχωρείται εξ ορισμού η τιμή NIL. Επειδή οι Sum και Total είναι τοπικές μεταβλητές, οι υπορουτίνες που καλούνται μέσα από την My_Proc δεν θα μπορούν να τις προσπελάσουν.

Οι Μεταβλητές Static

Οι μεταβλητές *static (στατικές)* είναι παρόμοιες με τις μεταβλητές local, με τη διαφορά ότι δεν χάνουν την τιμή τους μετά την ολοκλήρωση της υπορουτίνας στην οποία δηλώθηκαν. Οι στατικές μεταβλητές μπορούν να χρη-

σιμοποιηθούν μόνο από την υπορουτίνα στην οποία δηλώθηκαν και δεν είναι προσπελάσιμες από άλλες procedures ή functions. Οι στατικές μεταβλητές που δεν διαθέτουν αρχική τιμή παίρνουν αυτόματα την τιμή NIL.

Δείτε το επόμενο παράδειγμα :

```
PROCEDURE My_Proc
        STATIC  Counter := 0, Print_Status := 'ON'
        /* εντολές */
RETURN
```

Οι Μεταβλητές Private

Αν και οι μεταβλητές μπορούν να δηλωθούν ρητά ως private με τη δήλωση PRIVATE, κάθε μη δηλωμένη μεταβλητή, στην οποία αποδόθηκε μια τιμή, χαρακτηρίζεται αυτόματα ως PRIVATE. Όπως και με τις τοπικές μεταβλητές (local), οι μεταβλητές *private (ιδιωτικές ή αποκλειστικές)* εξακολουθούν να ισχύουν, έως ότου η υπορουτίνα στην οποία δηλώθηκαν ολοκληρωθεί.

Ωστόσο, σε αντίθεση με τις μεταβλητές local, οι μεταβλητές private είναι προσπελάσιμες απ' όλες τις υπορουτίνες οι οποίες καλούνται από την υπορουτίνα που τις δημιούργησε.

Δείτε το επόμενο παράδειγμα :

```
PROCEDURE My_Proc
        PRIVATE  Menu := 'FIRST'
        iLoop := 1
        /* εντολές */
RETURN
```

Η μεταβλητή Menu παίρνει την τιμή FIRST και αν η iLoop δεν έχει οριστεί προηγουμένως, τότε θα δημιουργηθεί ως private και θα της δοθεί η τιμή 1. Οι procedures ή functions που καλούνται από την My_Proc μπορούν να χρησιμοποιήσουν τις μεταβλητές Menu και iLoop. Στις μεταβλητές private που δηλώνονται χωρίς να παίρνουν αρχική τιμή, δίνεται αυτόματα η τιμή NIL.

Οι Μεταβλητές Public

Οι μεταβλητές *public (καθολικές ή κοινόχρηστες)* είναι προσπελάσιμες απ' όλες τις υπορουτίνες της εφαρμογής και ποτέ δεν χάνουν τα περιεχόμενά τους από τη μνήμη. Στις μεταβλητές public που δεν διαθέτουν αρχική τιμή, δίνεται η τιμή .F.

Δείτε το επόμενο παράδειγμα :

```
PROCEDURE My_Proc
        PUBLIC  Designer := 'Frankel'
        /* εντολές */
RETURN
```

Τύποι Δεδομένων

Το είδος δεδομένων που αποθηκεύεται σε μια μεταβλητή ονομάζεται τύπος δεδομένων (data type). Για παράδειγμα, η μεταβλητή που έχει την τιμή 'ΙΕΚ Φλώρινας' ανήκει στον τύπο δεδομένων Character (Χαρακτήρας) και η μεταβλητή που περιέχει την τιμή 5 ανήκει στον τύπο δεδομένων Numeric (Αριθμητικός).

Μια μεταβλητή μπορεί να ανήκει μόνο σ' έναν τύπο δεδομένων κάθε φορά. Γενικά, ο τύπος δεδομένων μιας μεταβλητής καθορίζεται την πρώτη φορά που εκχωρείται μια τιμή σ' αυτήν, αλλά μπορεί να αλλάξει στη διάρκεια του προγράμματος αν της εκχωρηθεί μια τιμή διαφορετικού τύπου.

Ο Clipper χρησιμοποιεί πέντε τύπους δεδομένων :

Character, Numeric, Logical, Date και Nil.

Ο τύπος δεδομένων *Character (χαρακτήρα)* χρησιμοποιείται για τον χειρισμό αλφαριθμητικών, δηλ. σειρών χαρακτήρων (strings) που περικλείονται από ένα ζευγάρι διαχωριστικών συμβόλων (delimiters). Ο Clipper δέχεται τρία είδη διαχωριστικών συμβόλων :

- δύο απλά εισαγωγικά : 'Florina'
- δύο διπλά εισαγωγικά : "Florina"
- αριστερή και δεξιά αγκύλη : [Florina]

Τα μηδενικά ή κενά αλφαριθμητικά γράφονται μ' ένα ζευγάρι διαχωριστικών συμβόλων, χωρίς ενδιάμεσο χαρακτήρα : "".

Ο τύπος δεδομένων *Numeric (αριθμητικός)* χρησιμοποιείται για τη δήλωση δεδομένων με τα οποία θέλουμε να κάνουμε μαθηματικές πράξεις.

Ο τύπος δεδομένων *Logical (λογικός)* χρησιμοποιείται για τον προσδιορισμό δεδομένων τύπου Boolean, δηλ. αυτών που μπορεί να έχουν δύο μόνο πιθανές τιμές, True ή False, Yes ή No. Οι αποδεκτές τιμές των δεδομένων τύπου logical είναι : y, Y, t, T, n, N, f, και F.

Οι τιμές y, Y, t, T (αληθές) είναι ισοδύναμες μεταξύ τους όπως και οι n, N, f, F (ψευδές). Οι μεταβλητές logical ορίζονται με την εκχώρηση σ' αυτές μιας από τις παραπάνω τιμές, κλεισμένες ανάμεσα σε δύο τελείες.

Δείτε τα παρακάτω παραδείγματα :

Married := .Y.
Print_On := .f.
Fylo := .T.

Ο τύπος δεδομένων *Date (ημερομηνίας)* χρησιμοποιείται για τον χειρισμό ημερομηνιών. Ο Clipper περιέχει πολλούς τελεστές και πολλές συναρ-

τήσεις που κυριολεκτικά μάς λύνουν τα χέρια στην επεξεργασία των ημερομηνιών, όπως το ότι μπορούμε να βρούμε πόσες ημέρες μεσολαβούν ανάμεσα σε δύο ημερομηνίες, ακόμη κάνει αυτόματο έλεγχο για δίσεκτα έτη, για σωστή καταχώρηση ημερών και μηνών (π.χ. όχι 32 Δεκ) κ.ά.

Η συνάρτηση DATE() επιστρέφει την τρέχουσα ημερομηνία από το ρολόι του υπολογιστή μας και η συνάρτηση TIME() την ώρα.

Οι ημερομηνίες ορίζονται μέσω της συνάρτησης CTOD(), η οποία μετατρέπει ένα αλφαριθμητικό σε ημερομηνία. Έτσι, για να δημιουργήσουμε μια ημερομηνία, την γράφουμε με τη μορφή αλφαριθμητικού ('ΜΜ/ΗΗ/ΕΕ') και με τη συνάρτηση CTOD() την μετατρέπουμε σε τιμή ημερομηνίας.

Δείτε τα παρακάτω παραδείγματα :

```
Birth_Date := CTOD('02/12/91')
Lib_Florina := CTOD('11/07/12')
Date_Timol = CTOD(' / / ')  // μηδενική ημερομηνία για αρχική τιμή
```

Υπάρχουν και οι μεταβλητές τύπου *NIL* που δηλώνονται χωρίς ταυτόχρονα να τους εκχωρείται αρχική τιμή. Μπορούμε, όμως, να δηλώσουμε ρητά μια μεταβλητή ως τύπου NIL, δίνοντάς της απευθείας τιμή :

```
Test_Value := NIL
```

Ο τύπος NIL πληροφορεί την εφαρμογή ότι η συγκεκριμένη μεταβλητή δεν διαθέτει καμία τιμή.

Υπάρχει και ο τύπος δεδομένων *πίνακα (array)*. Ένας πίνακας είναι ένα σύνολο μεταβλητών με το ίδιο όνομα. Για να δημιουργήσουμε έναν πίνακα, δίνουμε την εξής εντολή :

```
DECLARE PIN[100]
```

Έτσι, ορίζουμε μια μεταβλητή με όνομα PIN, η οποία έχει 100 στοιχεία στα οποία μπορούν να αποθηκευθούν δεδομένα. Σε καθένα απ' αυτά τα στοιχεία μπορεί να εκχωρηθεί μια τιμή αλφαριθμητική, αριθμητική, λογική ή ημερομηνίας.

Π.χ.

```
PIN[1] := 'Νίκος'
PIN[2] := 15
PIN[3] := .T.
PIN[4] := CTOD('10/13/97')
```

Ο Clipper υποστηρίζει και πίνακες δύο διαστάσεων :

```
DECLARE A[5][5]
```

Στον Clipper, εκτός από τους γνωστούς μας τύπους δεδομένων, έχει προστεθεί και ένας επιπλέον τύπος δεδομένων, το *πεδίο σημειώσεων (memo)*. Αυτό στην πραγματικότητα είναι ένα συνεχές τμήμα πληροφοριών και βρίσκεται αποθηκευμένο σ' ένα βοηθητικό αρχείο (γνωστό σαν αρχείο .DBT), το οποίο έχει έναν δείκτη αρχής τμήματος στην αρχική βάση δεδομένων.

Η συνάρτηση *TYPE()* μπορεί να ελέγξει τους τύπους δεδομένων και από την τιμή που μας επιστρέφει μπορούμε να δούμε ποιος είναι ο τύπος δεδομένων που υπάρχει σαν όρισμα.

Οι τιμές που μπορεί να επιστρέψει η TYPE() είναι :

Κώδικας	**Επιστροφή**	**΄Εννοια**
TYPE("array")	A	Τύπος δεδομένων πίνακα
TYPE("block")	B	Τύπος δεδομένων τμήματος κώδικα
TYPE("string")	C	Τύπος δεδομένων χαρακτήρα
TYPE("date")	D	Τύπος δεδομένων ημερομηνίας
TYPE("logical")	L	Τύπος λογικών δεδομένων
TYPE("Clients->memo")	M	Πεδίο σημειώσεων
TYPE("number")	N	Τύπος αριθμητικών δεδομένων
TYPE("object")	O	Τύπος δεδομένων αντικειμένου
TYPE("unknown")	U	Τύπος δεδομένων NIL ή αόριστος

Τελεστές και Εκφράσεις

Τελεστές του Clipper

Οι *τελεστές* είναι σύμβολα που χρησιμοποιούνται σε συνδυασμό με τις μεταβλητές, με σκοπό τη δημιουργία εκφράσεων. Για παράδειγμα, η έκφραση 5+6 περιέχει τις αριθμητικές σταθερές 5 και 6, καθώς και τον τελεστή της πρόσθεσης +. Οι αριθμοί αποτελούν τα δεδομένα εισόδου του τελεστή ή τους *τελεστέους*.

Ορισμένοι τελεστές απαιτούν την ύπαρξη ενός τελεστέου *(τελεστές unary)*, ενώ άλλοι χρειάζονται δύο *(τελεστές binary)*.

Αλφαριθμητικοί Τελεστές

Οι δύο βασικοί αλφαριθμητικοί τελεστές είναι οι τελεστές συνένωσης (concatenators), οι οποίοι προσθέτουν ή συνενώνουν αλφαριθμητικά. Και οι δύο είναι τύπου binary, δηλ. απαιτούν δύο αλφαριθμητικά ορίσματα και επιστρέφουν ένα αλφαριθμητικό.

Οι τελεστές αυτοί είναι :

+ Βασική συνένωση
- Συνένωση χωρίς τα τυχόν ενδιάμεσα κενά διαστήματα

Ο τελεστής - μετακινεί τα τυχόν κενά διαστήματα που υπάρχουν στο τέλος του πρώτου αλφαριθμητικού προς το τέλος του τελικού αλφαριθμητικού. Αντίθετα, ο τελεστής + δεν μετακινεί τα κενά διαστήματα.

Δείτε τα παρακάτω παραδείγματα :

```
LOCAL  Name1,  Name2
Name1 = 'Thomas  ' + 'Smith'
Name2 = 'Thomas  ' -  'Smith'
```

Η μεταβλητή Name1 θα περιέχει τα κενά διαστήματα που είναι μετά το Thomas, επειδή χρησιμοποιήθηκε ο τελεστής + και θα εμφανίζεται σαν 'Thomas Smith', ενώ η μεταβλητή Name2 θα εμφανίζεται σαν 'Thomas Smith', επειδή ο τελεστής - διέγραψε όλα τα κενά διαστήματα των αλφαριθμητικών.

Και οι δύο προηγούμενες εμφανίσεις δεν στέκονται καλά αισθητικά και έτσι, σαν τρίτη λύση, θα μπορούσαμε να κάνουμε το εξής :

```
LOCAL Name
Name = 'Thomas          ' - ' ' + 'Smith'
```

για να παρεμβάλουμε ένα κενό διάστημα μεταξύ των δύο ονομάτων και το αποτέλεσμα θα είναι : 'Thomas Smith'.

΄Ενας άλλος αλφαριθμητικός τελεστής είναι ο $, που ονομάζεται τελεστής υποσυνόλου. Ελέγχει αν η έκφραση που βρίσκεται στα αριστερά του αποτελεί υποσύνολο της έκφρασης που βρίσκεται στα δεξιά του (ή περιέχεται σ' αυτήν). Ο τελεστής αυτός είναι binary και παράγει έξοδο τύπου logical.

Για παράδειγμα, η έκφραση : 'Hog' $ 'WartHog' θα επιστρέψει την τιμή .T. και η έκφραση 'hog' $ 'WartHog' την τιμή .F. Δηλαδή, ο τελεστής υποσυνόλου διακρίνει τα κεφαλαία από τα μικρά γράμματα.

Μαθηματικοί Τελεστές

Οι μαθηματικοί τελεστές απαιτούν την ύπαρξη αριθμητικών τύπων δεδομένων στη θέση των τελεστέων, είναι τύπου binary και επιστρέφουν αριθμητικές τιμές.

Οι μαθηματικοί τελεστές είναι οι εξής :

+	Πρόσθεση ή συμβολισμός θετικού αριθμού
-	Αφαίρεση ή συμβολισμός αρνητικού αριθμού
*	Πολλαπλασιασμός
/	Διαίρεση και πηλίκο μιας ακέραιας διαίρεσης
%	Υπόλοιπο μιας ακέραιας διαίρεσης
** ή ^	΄Υψωση σε δύναμη

Τελεστές Ημερομηνιών

Οι τελεστές ημερομηνιών επιτρέπουν την ανάμιξη διαφορετικών τύπων δεδομένων σε εκφράσεις του Clipper και είναι οι + και -. Οι τελεστές αυτοί επιτρέπουν την πρόσθεση ή την αφαίρεση ημερών, στη φόρμα αριθμητικών τιμών, σε ή από μεταβλητές ημερομηνίας. Με τη χρήση του τελεστή -, μπορούμε να βρούμε πόσες ημέρες μεσολαβούν ανάμεσα σε δύο ημερομηνίες.

Δείτε τα παρακάτω παραδείγματα :

```
LOCAL  Entry_date, Due_date, Billing_date
Entry_date = CTOD('11/27/91')
Due_date = Entry_date + 30        // μεταγενέστερη κατά 30 ημέρες
Billing_date = Due_date - 21      // προγενέστερη κατά 21 ημέρες
```

Λογικοί Τελεστές

Οι λογικοί τελεστές είναι οι εξής :

.AND.	Λογικό ΚΑΙ
.OR.	Λογικό Ή
.NOT. ή !	Λογική άρνηση

Το αποτέλεσμα μιας λογικής έκφρασης είναι πάντα μια τιμή τύπου logical, δηλ. .T. ή .F. Ο τελεστής .AND. επιστρέφει αληθές αποτέλεσμα, μόνο αν και οι δύο τελεστέοι του είναι αληθείς, ο τελεστής .OR. επιστρέφει αληθές αποτέλεσμα όταν ο ένας τουλάχιστον τελεστέος του είναι αληθής και ο τελεστής .NOT. ή ! επιστρέφει αληθές αποτέλεσμα, αν ο τελεστέος του είναι ψευδής.

Δείτε τα παρακάτω παραδείγματα :

```
LOCAL  First_pass := .T., Printer_on := .F., Menu_on := .T.
LOCAL Test1, Test2, Test3
Test1 = First_pass .AND. Printer_on      // αποτέλεσμα .F.
Test2 = Menu_on .OR. Printer_on // αποτέλεσμα .T.
Test3 = !Printer_on                      // αποτέλεσμα .T.
```

Σχεσιακοί Τελεστές

Επιτρέπουν τη σύγκριση μεταβλητών του ίδιου τύπου, είναι όλοι δυαδικοί, επιστρέφουν τιμές τύπου logical και μπορούν να χρησιμοποιηθούν με κάθε τύπο δεδομένων.

Οι σχεσιακοί τελεστές είναι οι εξής :

<	Μικρότερο
>	Μεγαλύτερο
=	Ίσο
≠, <>, !=	Άνισο
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο

Ο Clipper 5.01 έχει έναν κανόνα για τα αλφαριθμητικά που συγκρίνονται μ' άλλα αλφαριθμητικά : θεωρώντας ότι η EXACT είναι OFF, ο Clipper θα συγκρίνει το αλφαριθμητικό στα δεξιά της σύγκρισης με το αλφαριθμητικό αριστερά, μέχρι να εξαντληθούν οι χαρακτήρες.

Αν τα δύο αλφαριθμητικά είναι όμοια μέχρι αυτό το σημείο, ακόμη κι αν το αλφαριθμητικό στα αριστερά είναι μεγαλύτερο, ο Clipper θα θεωρήσει ότι τα αλφαριθμητικά είναι ισοδύναμα.

Δηλαδή, η παρακάτω παράσταση θα δώσει αληθή (.T.) τιμή :

```
x = " "
y = " Γεια σου κόσμε"
? y = x
```

Στο παραπάνω παράδειγμα, αν αντιστρέψουμε τη σειρά της σύγκρισης (? x=y) δεν θα ισχύει πια το ίδιο και η σύγκριση θα δώσει ψευδή τιμή.

Υπάρχει, όμως, ένας καινούργιος τελεστής, ο ==, που κάνει ακριβή σύγκριση :

```
x = " "
y = " Γεια σου κόσμε"
? x == y
```

Η παραπάνω εντολή λέει στον Clipper να αξιολογήσει το αλφαριθμητικό από αριστερά προς τα δεξιά και το αντίθετο.

Ουσιαστικά, ο τελεστής == κάνει το εξής :

```
? ((x = y) .AND. (y = x))
```

Ο τελεστής διπλής ισότητας (==) λειτουργεί μ' όλους τους τύπους δεδομένων.

Δείτε και τα παρακάτω παραδείγματα :

```
? "AMERICA" = "A"
? "AMERICA" = "AM"
? "AMERICA" = "AME"
```

Όλες οι παραπάνω παραστάσεις δίνουν αληθείς τιμές, ενώ όλες οι παρακάτω παραστάσεις δίνουν ψευδείς τιμές.

```
? "A" = "AMERICA"
? " " = "AMERICA"
? "AMERICA = "A "
? "AM" = "AMERICA"
```

Τελεστές Εκχώρησης

Χρησιμοποιούνται για την εκχώρηση τιμών σε μεταβλητές και είναι όλοι τύπου binary. Ο καθένας απαιτεί μια μοναδική μεταβλητή στην αριστερή του πλευρά και μια έκφραση ή μια μεταβλητή ή μια σταθερά στη δεξιά του πλευρά.

Ο *τελεστής απλής εκχώρησης*, =, χρησιμοποιείται για την εκχώρηση μιας τιμής σε μια συγκεκριμένη μεταβλητή. Το σύμβολο = χρησιμοποιείται επίσης και για το συμβολισμό της ισότητας.

Δείτε το παρακάτω παράδειγμα :

```
LOCAL Test1, Test2
Test1 = 7              // απλή εκχώρηση
? (Test2 = Test2)      // έλεγχος ισότητας
```

Ο *τελεστής εκχώρησης γραμμής*, := (in-line) μπορεί να χρησιμοποιηθεί στη θέση μιας απλής εκχώρησης, αλλά έχει και μια επιπλέον λειτουργία : πρέπει να χρησιμοποιείται κατά την εκχώρηση αρχικής τιμής σε μια μεταβλητή.

Τιμές σε μεταβλητές μπορούμε να δώσουμε και με την εντολή STORE ως εξής :

```
STORE  10  TO A
STORE  'Bill'  TO NAME
STORE  CTOD('05/25/68')  TO  BIRTH_DATE
```

Προτεραιότητα Τελεστών

Σε περίπλοκες δηλώσεις που περιλαμβάνουν πολλούς τελεστές, ισχύει η εξής σειρά προτεραιότητας στους υπολογισμούς :

- Αλφαριθμητικοί
- Ημερομηνιών
- Μαθηματικοί
- Σχεσιακοί
- Λογικοί
- Εκχώρησης

Πρώτα εκτελούνται οι πράξεις μέσα στις παρενθέσεις. Οι αλφαριθμητικοί, οι σχεσιακοί και οι τελεστές ημερομηνιών εκτελούνται από αριστερά προς τα δεξιά. Οι λογικοί τελεστές εκτελούνται με την εξής σειρά : λογική άρνηση, λογικό ΚΑΙ και λογικό ΄Η. Για τους μαθηματικούς τελεστές ισχύει η γνωστή σειρά εκτέλεσης των πράξεων (πρόσημο, δύναμη, πολλαπλασιασμός και διαίρεση, πρόσθεση και αφαίρεση).

Πίνακας Τελεστών του Clipper

Οι τελεστές του Clipper είναι οι εξής :

&	Αντικατάσταση μακροεντολής
&&	Σημείωση ή εντολή in-line
@	Μεταβλητή που μεταβιβάζεται κατ' αναφορά (by reference)
/*	Αρχή σχολίων
*/	Τέλος σχολίων
.	Τερματισμός μακροεντολής
;	Συνέχιση εντολής στην επόμενη γραμμή ή πολλές εντολές στην ίδια γραμμή
;;	Προτάσεις πολλαπλών εντολών προεπεξεργαστή (ισχύει μόνο για τις εντολές #translate και #command)
( )	Οριοθέτης παράστασης
{}	Οριοθέτης πίνακα
[ ]	Οριοθέτης δείκτη πίνακα ή προαιρετικός οριοθέτης αλφαριθμητικών προεπεξεργαστή
,,	Οριοθέτης αλφαριθμητικών
""	Οριοθέτης αλφαριθμητικών
{¦¦}	Οριοθέτης τμήματος κώδικα
:	Τελεστής αποστολής
$	Τελεστής υποαλφαριθμητικού
=	Ισότητα ή σύγκριση
==	Ακριβής σύγκριση
:=	Ισότητα in-line
+	Πρόσθεση, θετική παράσταση ή συνένωση
-	Αφαίρεση, αρνητική παράσταση ή συνένωση με αφαίρεση των κενών που μεσολαβούν
/	Διαίρεση και πηλίκο ακέραιας διαίρεσης
*	Σημείο πολλαπλασιασμού ή δημιουργία αλφαριθμητικού για τον προεπεξεργαστή
**	΄Υψωση σε δύναμη
^	΄Υψωση σε δύναμη
%	Υπόλοιπο ακέραιας διαίρεσης
!	Άρνηση παράστασης (Σημ.: Αν ακολουθεί κενό, ισοδυναμεί με την εντολή RUN)
!=	Σύγκριση ανισότητας
<>	Σύγκριση ανισότητας
#	Σύγκριση ανισότητας ή πρόταση προεπεξεργαστή
<	Μικρότερο από ή σημειωτής μέρους (προεπεξεργαστή)
>	Μεγαλύτερο από ή σημειωτής μέρους (προεπεξεργαστή)
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο
=>	Οδηγία προεπεξεργαστή (μόνο στις εντολές #translate και #command)
->	Δείκτης ψευδωνύμου πεδίου
+=	Πρόσθεση και εκχώρηση in-line
-=	Αφαίρεση και εκχώρηση in-line

*=	Πολλαπλασιασμός και εκχώρηση in-line
/=	Διαίρεση και εκχώρηση in-line
**= ή ^=	Ύψωση σε δύναμη και εκχώρηση in-line
%=	Υπόλοιπο και εκχώρηση in-line
++	Αύξηση κατά 1 πριν ή μετά
- -	Μείωση κατά 1 πριν ή μετά
.OR.	Λογικό Ή
.NOT.	Λογικό ΟΧΙ
.AND.	Λογικό ΚΑΙ