

1. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ

1.1. ΣΥΝΟΛΟ ΧΑΡΑΚΤΗΡΩΝ

Το σύνολο των χαρακτήρων της QuickBASIC (QB) απαρτίζεται από τους αριθμητικούς χαρακτήρες (0-9), τους αλφαριθμητικούς, κεφαλαία (A-Z) και πεζά (a-z), καθώς και τους περισσότερους ειδικούς.

1.2. ΣΤΑΘΕΡΕΣ

1.2.1 Αριθμητικές σταθερές.

Υπάρχουν τέσσερις τύποι:

- ⇒ ακέραιοι σε 2 bytes με τιμές από -32768 μέχρι 32767.
- ⇒ “μακροί” ακέραιοι (long integers) σε 4 bytes με τιμές από -2.147.483.648 μέχρι 2.147.483.647.
- ⇒ πραγματικοί απλής ακρίβειας (real single precision) σε 4 bytes με τιμές τάξης μεγέθους 10^{38} και ακρίβεια 6-7 δεκαδικών ψηφίων.
- ⇒ πραγματικοί διπλής ακρίβειας (real double precision) σε 8 bytes με τιμές τάξης μεγέθους 10^{308} και ακρίβεια 15-16 δεκαδικών ψηφίων.

1.2.2 Αλφαριθμητικές σταθερές.

Μια αλφαριθμητική σταθερά είναι μια στοιχειοσειρά (string) με μήκος μέχρι 32767 χαρακτήρες περικλειόμενη σε εισαγωγικά.

1.2.3 Συμβολικές σταθερές.

Οι συμβολικές σταθερές ορίζονται με την εντολή CONST και μπορούν να χρησιμοποιηθούν αντί των αριθμητικών ή αλφαριθμητικών σταθερών. Π.χ.

```
CONST MaxElem% = 255
```

Ορίζεται η συμβολική σταθερά MaxElem ίση με 255.

Οι συμβολικές σταθερές μπορεί να είναι οποιουδήποτε τύπου και το όνομά τους σχηματίζεται με βάση τους κανόνες δημιουργίας μεταβλητών της QB. Αν χρησιμοποιηθούν οι χαρακτήρες %, &, !, # και \$ για τον προσδιορισμό του τύπου της σταθεράς, τότε οι χαρακτήρες αυτοί **δεν** αποτελούν μέρος του ονόματος. Οι συμβολικές σταθερές δεν επηρεάζονται από τις δηλωτικές εντολές DEF.

1.3. ΜΕΤΑΒΛΗΤΕΣ

Μια μεταβλητή είναι ένα όνομα το οποίο αναφέρεται σε ένα αντικείμενο (έναν αριθμό, μία στοιχειοσειρά ή μια εγγραφή). Το όνομα μιας μεταβλητής σχηματίζεται από 40 το πολύ αριθμητικούς και αλφαριθμητικούς χαρακτήρες με τον πρώτο υποχρεωτικά αλφαριθμητικό. Στους αλφαριθμητικούς χαρακτήρες των ονομάτων μεταβλητών δεν έχει σημασία, αν είναι κεφαλαία ή πεζά γράμματα. Η QB δεν ξεχωρίζει τα κεφαλαία από τα

πεζά γράμματα. Έτσι τα ονόματα NUMBER, number και NumBeR αναφέρονται στην ίδια θέση μνήμης, πρόκειται δηλ. για ίδιες μεταβλητές. Ο χρήστης μπορεί να χρησιμοποιεί κεφαλαία ή πεζά γράμματα κατά την κρίση του. Συνήθως οι μεταβλητές ονοματίζονται έτσι, ώστε το όνομα να προσδιορίζει και το περιεχόμενο. Στην περίπτωση αυτή χρησιμοποιούνται και κεφαλαία γράμματα για έμφαση π.χ. InventoryItem, StockElement, κ.λπ.

Μια μεταβλητή μπορεί να είναι αριθμητική, αλφαριθμητική ή τύπου εγγραφής. Ο προσδιορισμός του τύπου μιας μεταβλητής μπορεί να γίνει με έναν από τους επόμενους τρόπους.

1. Χρησιμοποιώντας τους παρακάτω ειδικούς χαρακτήρες στο τέλος του ονόματος της μεταβλητής.

% για ακέραιες π.χ. a%, Delta%

& για μακρές ακέραιες π.χ. a&, Value&

! για πραγματικές απλής ακρίβειας π.χ. a!, Sum!

για πραγματικές διπλής ακρίβειας π.χ. a#, Total#

\$ για αλφαριθμητικές π.χ. a\$, Nam\$

Οι χαρακτήρες αυτοί αποτελούν μέρος του ονόματος της μεταβλητής, γι' αυτό και οι μεταβλητές a%, a&, a!, a# και a\$ είναι όλες διαφορετικές μεταξύ τους. Είναι διαφορετικές επίσης και από την a η οποία ορίζεται κατ' αρχήν (by default) ως μεταβλητή απλής ακρίβειας.

2. Δηλώνοντας τον τύπο της μεταβλητής σε δηλωτικές εντολές του τύπου:

```
âïôïëß üíîîá-îäöääëçðëð AS ôýðîð
```

όπου η εντολή μπορεί να είναι μία από τις DIM, COMMON, REDIM, SHARED ή STATIC και ο τύπος ένας από τους INTEGER, LONG, SINGLE, DOUBLE, STRING ή τύπος δεδομένων από τον χρήστη.

Π.χ. με την εντολή

```
DIM x AS LONG
```

η μεταβλητή x ορίζεται μακρύς ακέραιος.

Στον ορισμό αλφαριθμητικών μεταβλητών υπάρχουν δύο δυνατότητες, μεταβλητές σταθερού ή μεταβλητού μήκους. Π.χ. με την εντολή

```
DIM Nam1 AS STRING
```

ορίζεται μια μεταβλητή με μήκος που μπορεί να εκτείνεται από 0 μέχρι 32767 χαρακτήρες.

Με την εντολή

```
DIM Nam2 AS STRING*20
```

ορίζεται μια μεταβλητή με μήκος ακριβώς 20 χαρακτήρες.

Με τον ίδιο τρόπο μπορεί να δηλωθεί μια μεταβλητή εγγραφής, αρκεί προηγούμενα να έχει ορισθεί η εγγραφή με την εντολή TYPE. Π.χ.

```

TYPE StockItem
    Code      AS STRING * 4
    Description AS STRING * 20
    UnitPrice AS SINGLE
    Quantity  AS LONG
END TYPE
DIM CurrentItem AS StockItem

```

Μετά τον ορισμό μιας εγγραφής η αναφορά σε ένα πεδίο της μπορεί να γίνει χρησιμοποιώντας τον τύπο *όνομα-εγγραφής.όνομα-πεδίου*, π.χ. η μεταβλητή `StockItem.Code` αναφέρεται στο πεδίο κωδικός της εγγραφής με τό όνομα `StockItem`.

Ας σημειωθεί ότι ο χαρακτήρας τελεία (.) μπορεί να χρησιμοποιηθεί μόνο σε μεταβλητές τύπου εγγραφής.

3. Με τη χρήση των δηλωτικών εντολών `DEFINT`, `DEFLNG`, `DEFSNG`, `DEFDBL` και `DEFSTR` για ακέραιους, μακρούς ακέραιους, πραγματικούς απλής ακρίβειας, πραγματικούς διπλής ακρίβειας και αλφαριθμητικές μεταβλητές αντίστοιχα. Με τις εντολές αυτές μεταβλητές που αρχίζουν από ένα γράμμα ή περιοχή γραμμάτων είναι συγκεκριμένου τύπου. Οι εντολές `DEF` επηρεάζουν μόνο μεταβλητές που ανήκουν στην ενότητα που αναφέρονται.

Π.χ. με την εντολή

```
DEFINT A-Z
```

ορίζονται όλες οι μεταβλητές ως ακέραιες.

1.4. ΠΙΝΑΚΕΣ

Ενας πίνακας είναι ένα σύνολο αντικειμένων επί των οποίων η αναφορά γίνεται με το ίδιο όνομα μεταβλητής. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες μέσα σε παρενθέσεις. Οι δείκτες προσδιορίζουν την τάξη του στοιχείου μέσα στον πίνακα. Π.χ.

```
A(3), Array(5,8), Nam$(20)
```

Σε ένα πίνακα ο αριθμός των δεικτών προσδιορίζει το πλήθος των διαστάσεων, ενώ η μέγιστη δυνατή τιμή κάθε δείκτη προσδιορίζει το πλήθος των στοιχείων του πίνακα ανά διάσταση. Πριν χρησιμοποιηθεί ένας πίνακας πρέπει να ορισθούν οι διαστάσεις του. Με τον όρο αυτό αναφερόμαστε τόσο στον ορισμό του πλήθους των διαστάσεων, όσο και των μέγιστων τιμών κάθε μιάς. Ο ορισμός των διαστάσεων ενός πίνακα επιτυγχάνεται με την εντολή `DIM`. Ο μέγιστος αριθμός των διαστάσεων ενός πίνακα είναι 60 και η μέγιστη τιμή μιας διάστασης 32767. Είναι δυνατόν ένας πίνακας να χρησιμοποιηθεί χωρίς να ορισθούν οι διαστάσεις με την `DIM`, αρκεί ο μέγιστος αριθμός στοιχείων ανά διάσταση να μην ξεπεράσει το 10. Με μία εντολή `DIM` δηλώνεται η μέγιστη τιμή κάθε δείκτη, οπότε ελάχιστη θεωρείται η μηδενική π.χ. `DIM A(100)`. Μπορεί να χρησιμοποιηθεί όμως και ο τύπος από-έως, οπότε προσδιορίζονται μαζί τα άνω και κάτω όρια π.χ. `DIM A(1 TO 100)`. Να σημειωθεί η πολύ σπουδαία δυνατότητα

να ορίζονται με τον τύπο αυτό και αρνητικές τιμές δεικτών π.χ. `DIM A(-5 TO 5)`. Έτσι μπορεί να ορισθεί οποιαδήποτε περιοχή τιμών των δεικτών από -32768 έως 32767.

Τέλος με την `DIM` όλα τα στοιχεία του πίνακα λαμβάνουν αρχική τιμή μηδέν ή `null`.

Για ονόματα πινάκων μπορούν να χρησιμοποιηθούν οποιοδήποτε μεταβλητές οποιουδήποτε τύπου. Ως δείκτες χρησιμοποιούνται ακέραιοι ή ακέραιες εκφράσεις (μπορούν να χρησιμοποιηθούν και μη ακέραιοι αλλά στρογγυλοποιούνται πριν από τη χρήση). Για το ορισμό ενός πίνακα με στοιχεία εγγραφές αρχειού πρέπει πρώτα να δηλωθεί ο τύπος της εγγραφής και μετά οι διαστάσεις του πίνακα. Π.χ.

```

TYPE QueueNode
    Data AS STRING * 20
    Pointer AS INTEGER
END TYPE
DIM Q(100) AS QueueNode

```

Κάθε στοιχείο του πίνακα `Q` είναι μία εγγραφή τύπου `QueueNode`. Η αναφορά σε ένα πεδίο της εγγραφής σαν στοιχείο πίνακα γίνεται με τη χρήση του τύπου *ονομα - πίνακα.πεδίο - εγγραφής*, π.χ.

```
PRINT Q(i).Data
```

1.5 ΕΚΦΡΑΣΕΙΣ

Για τη διαμόρφωση των εκφράσεων χρησιμοποιούνται σταθερές, μεταβλητές, τελεστές, συναρτήσεις και παρενθέσεις με τη συνήθη ιεραρχία. Υπάρχουν οι γνωστοί από τη ΓΛΩΣΣΑ αριθμητικοί και συγκριτικοί τελεστές με τη διαφορά ότι ως τελεστής ακέραιας διαίρεσης χρησιμοποιείται η ανάποδη κάθετος `\`.

Οι σπουδαιότεροι λογικοί τελεστές είναι οι `AND`, `OR` και `NOT`. Ακόμη ο τελεστής `+` χρησιμοποιείται και για τη συνένωση δύο στοιχειοσειρών (strings) σε μία.

Τέλος η `QB` διαθέτει ένα πλουσιότατο ρεπερτόριο ενσωματωμένων συναρτήσεων μαθηματικών, αλφαριθμητικών, κ.α. (βλ. βιβλιογραφία).

Κυριότερες μαθηματικές συναρτήσεις	
ABS	Απόλυτη τιμή
ATN	Τόξο εφαπτομένης
COS	Συνημίτονο
EXP	e^x
FIX	Ακέραιο μέρος
INT	Μικρότερος ακέραιος
LOG	Φυσικός λογάριθμος
RND	Τυχαίος αριθμός
SIN	Ημίτονο
SQR	Τετραγωνική ρίζα
TAN	Εφαπτομένη

γραμμα, δηλ. από την αρχή του κειμένου προς το τέλος και πάνω στην ίδια γραμμή από τ' αριστερά προς τα δεξιά. Μεταξύ δύο συνεχόμενων εντολών που βρίσκονται στην ίδια γραμμή απαιτείται η παρουσία ενός διαχωριστή εντολών, που εδώ είναι ο χαρακτήρας "άνω-κάτω τελεία" (:).

3.2 ΕΠΙΛΟΓΗ

Σε όλα τα επίπεδα του προγραμματισμού η απλούστερη απόκλιση από την ακολουθιακή δομή είναι το άλμα σε άλλο σημείο του προγράμματος με την εντολή GOTO. Όταν συναντάται η εντολή GOTO διακόπτεται η σειριακή εκτέλεση των εντολών και ο έλεγχος μεταφέρεται στο σημείο προορισμού της εντολής, γι' αυτό και το GOTO αναφέρεται πολλές φορές ως διακλάδωση άνευ όρων. Η εντολή GOTO ακολουθείται από μια διεύθυνση προορισμού που είναι μια αλφαριθμητική ετικέτα (label).

Η χρήση της GOTO στα προγράμματα παρουσιάζει αρκετά προβλήματα, γι' αυτό και οι πεπειραμένοι προγραμματιστές την αποφεύγουν. Αλλωστε η χρήση της δεν είναι απαραίτητη στον προγραμματισμό. Η QB διαθέτει την GOTO, κύρια για λόγους συμβατότητας με προηγούμενες εκδόσεις.

3.2.1. Σύνθετη επιλογή

Κατά την εκτέλεση ενός προγράμματος είναι δυνατό να επιλεγεί η εκτέλεση ορισμένων εντολών και να αποφευχθεί η εκτέλεση άλλων. Η πιο απλή δομή για κάτι τέτοιο είναι η IF..THEN..ELSE. Η σύνταξη της εντολής είναι:

```
IF B THEN E1 ELSE E2
```

Η B είναι μια λογική έκφραση (μια συνθήκη) και η οποία έχει τιμές: αληθής (true), αν η συνθήκη ισχύει και ψευδής (false), αν η συνθήκη δεν ισχύει. Οι E1 και E2 είναι εντολές ή ομάδες εντολών. Η λειτουργία της εντολής είναι η εξής: υπολογίζεται πρώτα η λογική τιμή της παράστασης B. Αν η τιμή της B είναι αληθής, τότε εκτελείται η εντολή E1, αλλιώς εκτελείται η E2.

Οι E1 και E2 μπορούν να είναι ατομικές εντολές ή σύνολο εντολών χωρισμένες με ":". Όλες σχεδόν οι εντολές της γλώσσας μπορούν να υπάρχουν μέσα στις E1 και E2, ακόμη και εντολές IF..THEN..ELSE, οπότε δημιουργούνται τα λεγόμενα εμφωλευμένα IF. Επίσης είναι δυνατόν το τμήμα ELSE να απουσιάζει, οπότε αν η συνθήκη δεν ισχύει, η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί την IF..THEN.

Μια εντολή IF..THEN..ELSE μπορεί να εκτείνεται σε περισσότερες από μία φυσικές γραμμές μέχρι ένα μέγιστο 255 χαρακτήρων.

Παράδειγμα. Ένας άνδρας χαρακτηρίζεται ελαφρύς (βαρύς) αν είναι κάτω (πάνω) από 80 κ. Επίσης χαρακτηρίζεται κοντός (ψηλός) αν είναι κάτω (πάνω) από 1.75. Στο επόμενο τμήμα προγράμματος εισάγο-

νται το βάρος και ύψος ενός ατόμου και εξάγεται ο χαρακτηρισμός του με ένα σύνθετο IF.

```
INPUT "Άρσενίδο =", A
INPUT "Θρσίδο =", O
IF B<80 THEN IF Y<1.75 THEN PRINT "êííðüð
êáé äëäðñýð" ELSE PRINT "øçëüð êáé äëäðñýð"
ELSE IF Y<1.75 THEN PRINT "êííðüð êáé ää-
ñýð" ELSE PRINT "øçëüð êáé ääñýð"
```

Για να αποφεύγονται αυτού του είδους πολύπλοκες εντολές, η QB διαθέτει έναν άλλο τύπο εντολής IF..THEN..ELSE, η οποία δεν περιορίζεται πλέον σε μία φυσική γραμμή προγράμματος. Η εντολή αυτή καλείται **ομαδοποιημένο** (block) IF, με την έννοια ότι οι ομάδες εντολών E1 και E2 μπορούν να εκτείνονται σε οποιονδήποτε αριθμό γραμμών προγράμματος. Επειδή τώρα το μήκος της εντολής δεν προκαθορίζεται, μία ειδική εντολή η END IF προσδιορίζει το τέλος της εντολής. Με τη νέα μορφή η εντολή IF στο προηγούμενο παράδειγμα γράφεται:

```
IF B<80 THEN
  IF Y<1.75 THEN
    PRINT "êííðüð êáé äëäðñýð"
  ELSE
    PRINT "øçëüð êáé äëäðñýð"
  END IF
ELSE
  IF Y<1.75 THEN
    PRINT "êííðüð êáé ääñýð"
  ELSE
    PRINT "øçëüð êáé ääñýð"
  END IF
END IF
```

Από το παράδειγμα αυτό γίνεται αμέσως φανερή η ευελιξία που προσφέρει στον προγραμματισμό αυτό ο τύπος της εντολής. Μάλιστα αυτό δεν επιδρά μόνο μόνο στη δημιουργία πολύπλοκων δομών ελέγχου, αλλά διευκολύνεται σημαντικά και η ανάγνωση και κατανόηση προγραμμάτων από κάθε τρίτον.

Ο τύπος αυτός της εντολής IF..THEN..ELSE διαθέτει και μια άλλη προαιρετική δήλωση την ELSEIF, με την οποία παρέχεται ακόμα μεγαλύτερη ευελιξία στη σύνταξη πολύπλοκων δομών. Π.χ.

```
INPUT "Άρσενίδο Ýíáí ÷äñäëðññä :", a$
IF a$>="0" AND a$<="9" THEN
  PRINT "øçðßí"
ELSEIF a$>="A" AND a$<="z" THEN
  PRINT "êáðéíéëü äñüííä"
ELSEIF a$>="A" AND a$<="ù" THEN
  PRINT "äëëçíéëü äñüííä"
ELSE
  PRINT "ìç äëäðñëëëçðëëüð ÷äñäëðññäð"
END IF
```

Κάθε μία από τις δηλώσεις IF, ELSEIF και ELSE ακολουθείται από μια ομάδα εντολών, οι οποίες δεν πρέπει να βρίσκονται στην ίδια γραμμή με τις IF, ELSEIF και ELSE. Αλλιώς η QB θεωρεί την εντολή αυτή σαν μιας γραμμής.

Η QB εξετάζει κάθε μία από τις συνθήκες της IF και των ELSEIF δηλώσεων από πάνω προς τα κάτω υπερ-πιδιώοντας τις ομάδες των εντολών που ακολουθούν μέχρι να βρεθεί η πρώτη αληθινή συνθήκη. Τότε εκτελούνται οι εντολές που αντιστοιχούν και μετά γίνεται διακλάδωση στην εντολή που ακολουθεί το END IF.

3.2.2 Πολιτική επιλογή

Με τον όρο αυτό αναφερόμαστε στη δυνατότητα μιας δομής να επιτυγχάνει την εκτέλεση μιας ομάδας εντολών ανάμεσα σε πολλές ανάλογα με την τιμή της εξεταζόμενης συνθήκης. Η σχετική εντολή είναι η SELECT..CASE. Πρόκειται για εντολή πολλαπλών επιλογών απόφασης ανάλογη, από πλευράς τελικού αποτελέσματος, με το ομαδοποιημένο IF..THEN..ELSE. Η βασική διαφορά της SELECT..CASE από το IF..THEN..ELSE είναι, ότι η πρώτη εξετάζει μια απλή έκφραση και ανάλογα με το αποτέλεσμα εκτελεί διαφορετικές εντολές, ενώ η δεύτερη εξετάζει απλές ή σύνθετες λογικές εκφράσεις. Αν και η λειτουργία της SELECT..CASE μπορεί να πραγματοποιηθεί και με την IF..THEN..ELSE, λόγω της συμπαγούς δομής της η χρήση της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματιστή.

Η γενική μορφή της εντολής είναι:

```
SELECT CASE ᚢᚳᚰᚱᚰᚰ
  CASE ᚳᚰᚰᚰᚰ-ᚳᚰᚰᚰ-1
    ᚰᚰᚰᚰᚰᚰ-1
  CASE ᚳᚰᚰᚰᚰ-ᚳᚰᚰᚰ-2
    ᚰᚰᚰᚰᚰᚰ-2
  . . . . .
CASE ELSE
  ᚰᚰᚰᚰᚰᚰ-ᚰ
END SELECT
```

Η έκφραση μπορεί να είναι αριθμητική ή αλφαριθμητική. Οι *λίστες-τιμών* μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές, περιοχή τιμών από-έως ή τιμές που υπακούουν σε μια λογική συνθήκη. Η εντολή *εξετάζει* την έκφραση και ανάλογα με την τιμή της εκτελεί τις εντολές μετά την CASE που αντιστοιχεί στην τιμή αυτή. Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμμία από τις λίστες τιμών, τότε εκτελούνται οι εντολές που ακολουθούν την CASE ELSE. Μετά την εκτέλεση των εντολών κάποιας CASE, το πρόγραμμα συνεχίζεται με την εντολή που ακολουθεί την END SELECT.

Η λειτουργία της εντολής καθώς και οι δυνατότητές της αναδεικνύονται με τα επόμενα παραδείγματα.

Παράδειγμα 1.

```
INPUT "Äþoä æéŷñáéí áðü 1 Ýùð 3 : " , i
SELECT CASE i
  CASE 1
    PRINT "Ðáñßððùðç 1"
  CASE 2
    PRINT "Ðáñßððùðç 2"
  CASE 3
    PRINT "Ðáñßððùðç 3"
```

```

CASE ELSE
    PRINT "EÜèiò"
END SELECT

```

Παράδειγμα 2.

```

INPUT "İpöa áēŷñáēī ìàôáŷ 0 êáē 9 : " , i
SELECT CASE i
    CASE 0
        PRINT "İçäŷı"
    CASE 1,3,5,7,9
        PRINT "İiüò"
    CASE 2,4,6,8
        PRINT "Æöäüò"
    CASE ELSE
        PRINT " İ 9 İ iç áēŷñáēiò"
END SELECT

```

Εδώ στις περιπτώσεις μονού ή ζυγού η *λίστα-τιμών* περιέχει όλες τις δυνατές τιμές χωρισμένες με κόμμα-τα.

Παράδειγμα 3.

```
INPUT "ᐱᐃᐅᐱ ᐱᓇᓇᓇᓇᓇᓇᓇᓇᓇ 1 ᓇᓇᓇ 20 : " , i
SELECT CASE i
CASE 1 TO 10
    PRINT "1ç ᐱᐱᓇᓇᓇᓇᓇᓇ"
CASE 11 TO 20
    PRINT "2ç ᐱᐱᓇᓇᓇᓇᓇᓇ"
CASE ELSE
    PRINT "ᐸᓇᓇᓇᓇᓇᓇᓇᓇ"
END SELECT
```

Εδώ στη *λίστα-τιμών* χρησιμοποιείται περιοχή τιμών από-έως, όπου μεταξύ αρχικής και τελικής τιμής τίθεται η λέξη ΤΟ.

Παράδειγμα 4.

```

INPUT "Óà ðíēŮ çēēēšá Ũñ+éóàð íá ìææšáíæō
                                     BASIC ; ",age
SELECT CASE age
  CASE IS<0
    PRINT "Ášðàìà çēēēšá"
  CASE IS<5
    PRINT "ÌŮēēíí ðá ðáñáēŸò! "
  CASE 5 TO 80
    PRINT "ÁíòŮìáē"
  CASE IS>80
    PRINT "ÊŮēēēí áñāŮ ðáñŮ ðìòŸ."
  CASE ELSE
    PRINT "ÁāŸíáðíí"
END SELECT

```

Εδώ χρησιμοποιείται η λέξη IS ακολουθούμενη από ένα συγκριτικό τελεστή και μια τιμή. Έτσι η περίπτωση CASE IS<5 πληρείται εφ' όσον age<5. Ως συγκριτικοί τελεστές μπορούν να χρησιμοποιηθούν όλοι οι γνωστοί τελεστές της BASIC δηλ. οι <, <=, >, >=, <> και =. Να σημειωθεί ότι αν μια τιμή της έκφρασης αντιστοιχεί σε περισσότερες από μία περιπτώσεις, τότε εκτελείται η πρώτη. Έτσι αν εισαχθεί αρνητικός αριθμός που είναι μικρότερος από το 0 και το 5, εκτελούνται οι εντολές που ακολουθούν την CASE IS<0.

Περισσότερες από μία συνθήκες ή περιοχές τιμών μπορούν να υπάρχουν σε μία CASE, όπως στα επόμενα παραδείγματα.

```
CASE -1, 0, 10 TO 20, 50 TO 90, IS>99
CASE i, j, k, -1 TO 1
CASE IS<"A", IS>"z"
CASE "á" òĭ "ù", "Û", "Ÿ", "Ɔ", "ß", "ü",
      "ȳ", "Ɔ"
```

3.3 ΕΠΑΝΑΛΗΨΗ Ή ΑΝΑΚΥΚΛΩΣΗ

Η δομή της επανάληψης ή ανακύκλωσης αναφέρεται στη επαναληπτική εκτέλεση μιας ομάδας εντολών προγράμματος κατά ένα προκαθορισμένο αριθμό φορών ή μέχρι να ισχύσει κάποια συνθήκη. Η δομή της ανακύκλωσης υλοποιείται με εντολές FOR-NEXT, WHILE-WEND και DO-LOOP.

3.3.1 FOR-NEXT

Η εντολή αυτή εκτελεί τις εντολές προγράμματος που βρίσκονται μεταξύ του FOR και του NEXT για όλες τις τιμές της μεταβλητής ελέγχου. Η μεταβλητή ελέγχου που αναφέρεται στη FOR, συνοδεύεται από την αρχική και τελική τιμή, καθώς και το βήμα μεταβολής της. Ως γνωστόν αν το βήμα είναι 1 τότε παραλείπεται. Στην υλοποίηση της εντολής από την QB έχει προστεθεί μια νέα δυνατότητα, ότι η έξοδος από το βρόχο μπορεί να γίνει σε οποιοδήποτε σημείο με τη χρήση της εντολής EXIT FOR. Στο επόμενο παράδειγμα χρησιμοποιείται η εντολή FOR-NEXT για να διατρέξει διαδοχικά όλα τα στοιχεία του πίνακα k\$(). Αν σημειωθεί ισοσύνη στον έλεγχο κάποιου στοιχείου του πίνακα με το ζητούμενο cle\$, τότε η διαδικασία διακόπτεται με την EXIT FOR.

```
INPUT "Εὔαέεῦδ : ",cle$
index = 0 : ex = 0
FOR i = 1 TO n
    IF cle$ = k$(i) THEN
        index = i : ex = 1 : EXIT FOR
    END IF
NEXT i
```

Στο τέλος λαμβάνεται η θέση του στοιχείου index, καθώς και μία μεταβλητή ex η οποία γίνεται 1 σε επιτυχημένη αναζήτηση.

3.3.2 WHILE-WEND

Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που υπάρχουν μεταξύ WHILE και WEND όσο ισχύει η συνθήκη που συνοδεύει το WHILE. Στο επόμενο τμήμα προγράμματος η WHILE ελέγχει τη συνθήκη "οχι τέλος αρχείου" και εφόσον ισχύει δηλ. δεν είναι στο τέλος του αρχείου, τότε γίνεται ανάγνωση μιας εγγραφής.

```
OPEN "I",#1,"SEQFILE.DAT"
WHILE NOT EOF(1)
    LINE INPUT#1, Rec$
    PRINT Rec$
```

```
WEND
CLOSE
```

3.3.3 DO-LOOP

Παρόμοια στη λογική με τη WHILE-WEND αλλά με περισσότερες δυνατότητες είναι και η εντολή DO-LOOP. Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που βρίσκονται μεταξύ του DO και του LOOP. Η εκτέλεση των εντολών επαναλαμβάνεται συνέχεια εφόσον μια συνθήκη ελέγχου παραμένει αληθής (WHILE) ή μέχρις ότου γίνει ψευδής (UNTIL). Ο έλεγχος της συνθήκης εξόδου μπορεί να γίνεται στην αρχή ή στο τέλος του βρόχου. Η έξοδος από το βρόχο μπορεί να γίνει σε οποιοδήποτε σημείο με τη χρήση της εντολής EXIT DO. Είναι δυνατόν επίσης να μην υπάρχει καθόλου συνθήκη ελέγχου, οπότε ο βρόχος εκτελείται επ' άπειρον. Στην περίπτωση αυτή η μοναδική δυνατότητα εξόδου παρέχεται με την EXIT DO.

Η DO-LOOP είναι μια εντολή ανακύκλωσης που μπορεί να χρησιμοποιηθεί σε κάθε περίπτωση. Η αξία της αναδεικνύεται με τα επόμενα παραδείγματα.

Παράδειγμα 1.

```
z$ = ""
DO UNTIL LEN(z$)<>0
    z$ = INKEY$
LOOP
```

Στο παράδειγμα αυτό ο βρόχος εκτελείται μέχρι να πληκτρολογηθεί κάποιος χαρακτήρας. Η συνθήκη ελέγχεται στην αρχή του βρόχου.

Παράδειγμα 2.

```
z$ = ""
DO WHILE LEN(z$) = 0
    z$ = INKEY$
LOOP
```

Ακριβώς η ίδια λειτουργία με το προηγούμενο γίνεται και στο παράδειγμα αυτό. Ο βρόχος εκτελείται εφόσον κανείς χαρακτήρας δεν εισάγεται. Και εδώ ο έλεγχος γίνεται στην αρχή του βρόχου.

Παράδειγμα 3.

```
DO
    INPUT "ΆíðŸîâé (í/ĭ) : "; z$
    z = INSTR("NOíi",z$)
LOOP UNTIL z<>0
```

Στο παράδειγμα αυτό ζητείται η εισαγωγή των χαρακτήρων N για NAI και O για OXI. Αν δεν δοθούν οι χαρακτήρες αυτοί και μόνο, καθώς και οι αντίστοιχοι πεζοί, τότε ο βρόχος επαναλαμβάνεται. Εδώ η συνθήκη εξετάζεται στο τέλος του βρόχου.

Παράδειγμα 4.

```
i% = INT(RND(1)*10)
j% = INT(RND(1)*10)
DO
```

```
PRINT i% ; "*" ; j% ; "=" ;
INPUT " ", k%
LOOP WHILE k% <> i% * j%
```

Στο παράδειγμα αυτό ζητείται η απάντηση του χρήστη στην ερώτηση ποιο είναι το γινόμενο των αριθμών $i\%$ επί $j\%$. Εφόσον η απάντηση δεν είναι σωστή, ο βρόχος επαναλαμβάνεται. Και εδώ ο έλεγχος της συνθήκης γίνεται στο τέλος.

Παράδειγμα 5.

```
s = 0
DO
  INPUT x
  IF x = 0 THEN EXIT DO
  s = s + x
LOOP
```

Στο παράδειγμα αυτό αθροίζονται τιμές που εισάγει ο χρήστης σε έναν αθροιστή. Το πλήθος των τιμών δεν είναι γνωστό εκ των προτέρων, γι' αυτό η έξοδος από το βρόχο γίνεται όταν πληκτρολογηθεί το μηδέν. Στην περίπτωση αυτή δεν υπάρχει καθόλου συνθήκη εξόδου στην αρχή ή το τέλος του βρόχου.

Από τα προηγούμενα γίνεται φανερό ότι ένας βρόχος μπορεί να υλοποιηθεί είτε με τη συνθήκη WHILE, είτε με την UNTIL. Να σημειωθεί τέλος ότι όταν ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου, αυτός εκτελείται τουλάχιστον μία φορά.

4. ΔΙΑΔΙΚΑΣΙΕΣ : ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Η κυριότερη ίσως φροντίδα του προγραμματιστή εφαρμογών στην ανάπτυξη των προγραμμάτων είναι να επιτύχει τον καλύτερο χωρισμό ενός προγράμματος σε μικρότερες ενότητες. Κάθε ενότητα αφού πρώτα ελεγχθεί χωριστά και έπειτα σε συνεργασία με άλλες, θεωρείται ότι έχει ολοκληρωθεί και δεν απομένει παρά η σύνδεσή της με τις υπόλοιπες προκειμένου να αποτελέσουν μαζί ένα ενιαίο λειτουργικό σύνολο, το ζητούμενο πρόγραμμα. Η QB παρέχει ένα μέσο κατάτμησης ενός μεγάλου προγράμματος σε μικρότερα, τις διαδικασίες (procedures). Ο όρος διαδικασία αναφέρεται στα **υποπρογράμματα** (subprograms) και τις **συναρτήσεις** (functions).

4.1 ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Υποπρόγραμμα είναι ένα αυτόνομο τμήμα προγράμματος το οποίο πραγματοποιεί μία ή περισσότερες ανεξάρτητες λειτουργίες. Ένα υποπρόγραμμα έχει ένα όνομα, μία αρχή και ένα τέλος και μπορεί να περιέχει οποιοδήποτε αριθμό εντολών. Ένα υποπρόγραμμα μπορεί να κληθεί από ένα πρόγραμμα, το κύριο πρόγραμμα ή άλλο υποπρόγραμμα προκειμένου να επιτελέσει τη λειτουργία για την οποία γράφτηκε. Σχετική με την έννοια του υποπρογράμματος είναι και η **υπορουτίνα** (subroutine). Υπορουτίνα είναι τμήμα προ-

γράμματος στο οποίο μπορεί να πραγματοποιηθεί διακλάδωση με τη χρήση της εντολής GOSUB. Η επιστροφή από την υπορουτίνα γίνεται με την εντολή RETURN. Παρόλο που τα υποπρογράμματα και οι υπορουτίνες επιτελούν ουσιαστικά παρόμοιες λειτουργίες, διαφέρουν σε πολλά σημεία και για να αποφευχθεί οποιαδήποτε σύγχυση στο κεφάλαιο αυτό ο όρος υποπρόγραμμα αναφέρεται στο τμήμα προγράμματος που περιλαμβάνεται ανάμεσα στις εντολές SUB και END SUB.

Οι κυριότερες διαφορές μεταξύ υποπρογραμμάτων και υπορουτινών είναι οι εξής:

1. Τα υποπρογράμματα μπορούν να χρησιμοποιούν **τοπικές** (local) ή **καθολικές** (global) μεταβλητές. Τοπικές μεταβλητές είναι αυτές που διατηρούν την τιμή τους μέσα στα όρια του υποπρογράμματος και δεν έχουν καμία σχέση με συνώνυμες μεταβλητές του κύριου προγράμματος ή άλλων υποπρογραμμάτων. Αντίθετα οι καθολικές μεταβλητές είναι κοινές για όλα τα υποπρογράμματα της ίδιας ενότητας. Σε ένα υποπρόγραμμα μία ή περισσότερες μεταβλητές αν δεν δηλωθούν ρητά σαν καθολικές, τότε είναι τοπικές (by default). Στις υπορουτίνες όλες οι μεταβλητές είναι καθολικές και δεν υπάρχει η δυνατότητα τοπικών μεταβλητών. Το γεγονός αυτό ενέχει σοβαρούς κινδύνους για τον προγραμματιστή και δεν είναι σπάνιες οι φορές που σε μια υπορουτίνα επηρεάζεται κατά λάθος μια μεταβλητή βασικής σημασίας άλλης υπορουτίνας.

2. Ένα υποπρόγραμμα είναι ένα ανεξάρτητο τμήμα προγράμματος, σε αντίθεση με την υπορουτίνα που είναι συνδεδεμένη με το πρόγραμμα από το οποίο χρησιμοποιείται. Το υποπρόγραμμα μπορεί να οριστεί σε μια ενότητα και να χρησιμοποιηθεί από άλλη. Αυτό επιφέρει σημαντική οικονομία στον απαιτούμενο κώδικα για την υλοποίηση εφαρμογών.

4.1.2. Ορισμός υποπρογραμμάτων

Ένα υποπρόγραμμα ορίζεται με την εντολή SUB. Η σύνταξη της εντολής είναι:

```
SUB ùíííá-ððíðñíãñÜííáôíð [(ëéððá
    ðáñáíáððñí)] STATIC
```

Όπου:

- ⇒ το *όνομα υποπρογράμματος* έχει μήκος έως 40 χαρακτήρες και δεν πρέπει να εμφανίζεται σε άλλη εντολή SUB στο ίδιο πρόγραμμα.
- ⇒ η *λίστα παραμέτρων* περιέχει σταθερές, μεταβλητές και πίνακες που περνούν από το κύριο πρόγραμμα στο υποπρόγραμμα. Οι παράμετροι χωρίζονται με κόμα.
- ⇒ η δήλωση STATIC είναι προαιρετική και προσδιορίζει τον τρόπο με τον οποίο χειρίζονται οι τοπικές μεταβλητές. Όταν υπάρχει, οι τοπικές μεταβλητές ορίζονται ως **στατικές** δηλ. διατηρούν την τιμή τους μεταξύ διαδοχικών κλήσεων του υποπρογράμματος. Αν παραληφθεί, όλες οι τοπικές μετα-

βλητές παίρνουν αρχική τιμή το μηδέν ή null κατά την κλήση του υποπρογράμματος. Οι τελευταίες καλούνται **αυτόματες** (automatic).

Το τέλος του υποπρογράμματος προσδιορίζεται με την εντολή:

```
END SUB
```

Μεταξύ των εντολών SUB και END SUB μπορεί να υπάρχει οποιοσδήποτε αριθμός εντολών. Όλες οι εκφράσεις της QuickBASIC επιτρέπονται μέσα σε ένα υποπρόγραμμα εκτός από:

- ⇒ συναρτήσεις ορισμένες από το χρήστη (user-defined functions) και οι εντολές ορισμού τους (DEF FN...END DEF, FUNCTION...END FUNCTION).
- ⇒ ένα μπλοκ SUB ... END SUB. Υποπρογράμματα δεν μπορούν να περιέχονται μέσα σε άλλα υποπρογράμματα (nested subprograms). Πάντως ένα υποπρόγραμμα μπορεί να καλεί ένα άλλο υποπρόγραμμα.

Παράδειγμα. Το επόμενο υποπρόγραμμα πραγματοποιεί τη λύση μιας πρωτοβάθμιας εξίσωσης $ax+b=0$. Δέχεται ως είσοδο τις τιμές των συντελεστών a και b και επιστρέφει ως αποτελέσματα την τιμή του x , εφόσον η εξίσωση έχει λύση, καθώς και την τιμή μιας μεταβλητής $flag$, η οποία είναι 1, αν υπάρχει λύση, αλλιώς 0.

```
SUB EqSolve1 (a,b,x,flag) STATIC
  x=0 : flag=1
  IF a=0 THEN flag=0 ELSE x=-b/a
END SUB
```

4.1.3. Κλήση υποπρογραμμάτων.

Η εκτέλεση μεταφέρεται από το κύριο πρόγραμμα στο υποπρόγραμμα με την εντολή CALL. Η εντολή CALL μπορεί να καλέσει υποπρογράμματα της QB, υποπρογράμματα σε άλλες γλώσσες ή υπορουτίνες σε γλώσσα ASSEMBLY.

Η σύνταξη της εντολής CALL είναι η εξής:

```
CALL üíííá-ððíðñüãñáííáðð [(ëßððá
                               íñέóíüðüí)]
```

Οπου:

- ⇒ **όνομα_υποπρογράμματος** είναι το όνομα του καλούμενου υποπρογράμματος.
- ⇒ η **λίστα_ορισμάτων** περιλαμβάνει μεταβλητές, πίνακες και στοιχεία πινάκων που περνούν στο καλούμενο υποπρόγραμμα.

Ο αριθμός και ο τύπος των ορισμάτων στη εντολή CALL πρέπει να είναι ίδια με τον αριθμό και τον τύπο των παραμέτρων στην εντολή SUB. Οι παράμετροι και τα ορίσματα πρέπει να δίνονται με την ίδια σειρά και χωρίζονται με κόμμα.

Με την CALL ο έλεγχος μεταβιβάζεται στο υποπρόγραμμα. Εκτελούνται όλες οι εντολές του και όταν συναντηθεί η END SUB, τότε γίνεται επιστροφή στο

κύριο πρόγραμμα στην εντολή που ακολουθεί την CALL. Επιστροφή στο κύριο πρόγραμμα μπορεί να γίνει και από οποιοδήποτε σημείο του υποπρογράμματος με την εντολή EXIT SUB.

Για παράδειγμα η κλήση του προηγούμενου υποπρογράμματος EqSolve1 για την εξίσωση $3X+2=0$ είναι:

```
CALL EqSolve1 (3,2,x,flag)
```

Ας σημειωθεί ότι η κλήση ενός υποπρογράμματος μπορεί να γίνει και χωρίς τη χρήση της CALL. Προς τούτο αρκεί να τεθεί μόνο το όνομα του υποπρογράμματος, αλλά τώρα τα ορίσματα δεν περικλείονται σε παρενθέσεις. Έτσι στο ίδιο παράδειγμα η κλήση γίνεται:

```
EqSolve1 3,2,x,flag
```

4.1.4. Μεταβίβαση τιμών με την εντολή CALL.

Σταθερές, μεταβλητές, πίνακες, εγγραφές και πεδία εγγραφών μπορούν να μεταβιβαστούν από το κύριο πρόγραμμα στο υποπρόγραμμα μέσω της λίστας ορισμάτων της εντολής CALL.

Απλές μεταβλητές. Το παράδειγμα που ακολουθεί δείχνει πως μια εντολή CALL καλεί ένα υποπρόγραμμα και περνά σ'αυτά τις μεταβλητές a και b .

```
´ Êýñέí ðñüãñáííá
a = 5 : b = 0
PRINT "Êýñέí ðñüãñáííá, ðñέí ðçí êëßðç:";
PRINT "a =" ; a, "b =" ; b
CALL Square (a,b)
PRINT "Êýñέí ðñüãñáííá, íáðü ðçí êëßðç:";
PRINT "a =" ; a, "b =" ; b
END
```

```
SUB Square (x,y) STATIC
y = x * x
PRINT "Óðí ððíðñüãñáííá, x =" ; x ; " êáέ y =" ; y
END SUB
```

Το παραπάνω πρόγραμμα εκτυπώνει τα εξής:

```
Êýñέí ðñüãñáííá, ðñέí ðçí êëßðç:
a = 5      b = 0
Óðí ððíðñüãñáííá, x = 5 êáέ y = 25
Êýñέí ðñüãñáííá, íáðü ðçí êëßðç:
a = 5      b = 25
```

Το υποπρόγραμμα αλλάζει την τιμή της μεταβλητής b δίνοντας μια νέα τιμή στην αντίστοιχη παράμετρο y της εντολής SUB. Όταν αρχίζει η εκτέλεση του υποπρογράμματος Square, η αρχική τιμή του y είναι 0. Το υποπρόγραμμα υπολογίζει μια νέα τιμή για το y (σ'αυτή την περίπτωση το 25) και την αποθηκεύει στην παράμετρο που πέρασε στο πρόγραμμα.

Οι τιμές των μεταβλητών που μεταβιβάζονται με τον παραπάνω τρόπο λέμε ότι περνούν **με αναφορά** (by reference). Αυτό σημαίνει ότι στο υποπρόγραμμα περνά η διεύθυνσή τους στη μνήμη. Το υποπρόγραμμα επενεργεί στην τιμή σ'αυτή τη διεύθυνση και όταν ο έ-

λεγχος επιστρέφει στο κύριο πρόγραμμα το περιεχόμενο της διεύθυνσης αυτής μπορεί να έχει αλλάξει. Αν δεν επιθυμούμε το υποπρόγραμμα να μεταβάλλει την τιμή της μεταβλητής, τότε πρέπει να μεταβιβάζεται μόνο η τιμή της μεταβλητής και όχι η διεύθυνσή της. Όταν περνά η πραγματική τιμή της μεταβλητής, το υποπρόγραμμα αντιγράφει την τιμή αυτή σε μια προσωρινή διεύθυνση για δική του χρήση, την οποία και ακυρώνει πριν ο έλεγχος γυρίσει στο κύριο πρόγραμμα. Η αρχική τιμή τότε δεν αλλάζει. Για να περάσουμε μια μεταβλητή **με τιμή** (by value) την περιβάλλουμε με παρενθέσεις. Για παράδειγμα, αν αλλάξουμε τη δήλωση CALL Square(a,b) στο προηγούμενο παράδειγμα σε CALL Square(a,(b)) το b περνά "με τιμή" η οποία είναι 0. Στο παράδειγμα το υποπρόγραμμα Square δεν μπορεί να αλλάξει την τιμή της μεταβλητής b. Το υποπρόγραμμα μπορεί ν'αλλάξει την τιμή του y αλλά αυτή η τιμή είναι τοπική στο υποπρόγραμμα και δεν μπορεί να γυρίσει πίσω στο κύριο πρόγραμμα μέσω της μεταβλητής b.

Το αναθεωρημένο παράδειγμα τυπώνει:

```
Εγρήϊ δñüāñāīā, δñēī δϑί ēēβδϑ:
a = 5   b = 0
Οδī δδīδñüāñāīā x = 5 ēāē y = 25
Εγρήϊ δñüāñāīā, īāδϑ δϑί ēēβδϑ:
a = 5   b = 5
```

Ας σημειωθεί ότι μπορούμε ακόμα να περάσουμε εκφράσεις ως παραμέτρους όπως παρακάτω:

```
CALL Prog2(x, (y+1))
```

Οι εκφράσεις περνούν "με τιμή".

Πίνακες. Με τον ίδιο τρόπο μπορούν να μεταβιβαστούν και πίνακες σε ένα υποπρόγραμμα.

Παράδειγμα. Το ακόλουθο υποπρόγραμμα υπολογίζει το άθροισμα των στοιχείων του μονοδιάστατου πίνακα X. Το πλήθος των στοιχείων του πίνακα μεταβιβάζεται στο υποπρόγραμμα με τη μεταβλητή N, ενώ το αποτέλεσμα επιστρέφεται στη μεταβλητή S.

```
SUB VectorSum (x(1),n,s) STATIC
s=0
FOR i=1 TO n
s=s+x(i)
NEXT i
END SUB
```

Η εντολή CALL με όρισμα έναν πίνακα έχει την ακόλουθη μορφή:

```
CALL üīīā-δδīδñīāñüīāδδīδ (üīīā δβīāēā())
```

π.χ. CALL VectorSum (x(),n,s)

Αν ένα υποπρόγραμμα δεν χρειάζεται έναν ολόκληρο πίνακα, μπορεί να μεταβιβαστούν μόνο όσα στοιχεία του πίνακα απαιτούνται. Για να μεταβιβαστεί ένα ατομικό στοιχείο ενός πίνακα αρκεί να προστεθούν και οι τιμές των δεικτών του στοιχείου μέσα στις παραμέτρους της CALL.

π.χ. CALL Prog1 (Array(3,9))

4.2 ΣΥΝΑΡΤΗΣΕΙΣ

Μία συνάρτηση είναι μια διαδικασία που επιστρέφει μια τιμή με το όνομά της. Η συνάρτηση διαφέρει από το υποπρόγραμμα, γιατί ενώ μπορεί να δεχτεί πολλές παραμέτρους, επιστρέφει πάντα μια μόνο τιμή. Οι συναρτήσεις μπορούν να εκτελούν οποιαδήποτε λειτουργία που επιθυμεί ο χρήστης, γι' αυτό και καλούνται συναρτήσεις ορισμένες από τον χρήστη (user defined functions) σε αντιδιαστολή με τις ενσωματωμένες συναρτήσεις της γλώσσας.

4.2.2. Ορισμός συναρτήσεων

Μία συνάρτηση ορίζεται με την εντολή FUNCTION. Η σύνταξη της εντολής είναι:

```
FUNCTION üīīā-όδīÜñδϑόϑδ (ēβδδā-δāñāīŸ-δñüī) STATIC
```

Το όνομα _συνάρτησης έχει μήκος έως 40 χαρακτήρες και δεν μπορεί να έχει αποδοθεί σε άλλη συνάρτηση ή/και υποπρόγραμμα. Για τη λίστα παραμέτρων και τη δήλωση STATIC ισχύουν όσα αναφέρθηκαν για τα υποπρογράμματα. Το τέλος τη συνάρτησης προσδιορίζεται από την εντολή END SUB. Μεταξύ της FUNCTION και END SUB μπορούν να υπάρχουν οποιοσδήποτε εντολές εκτός από αυτές που ορίζουν συναρτήσεις ή υποπρογράμματα. Η έξοδος από μία συνάρτηση επιτυγχάνεται με την εντολή EXIT SUB.

Παράδειγμα. Η επόμενη συνάρτηση ορίζει τη μαθηματική συνάρτηση $y = \eta\mu(x)/x$.

```
FUNCTION y (x) STATIC
IF x<>0 THEN
y=SIN(x)/x
ELSE
y=0
END IF
END FUNCTION
```

4.2.3 Κλήση συναρτήσεων και πέρασμα μεταβλητών

Η κλήση μιας FUNCTION γίνεται με τον ίδιο τρόπο που καλείται οποιαδήποτε ενσωματωμένη συνάρτηση της BASIC, δηλ. με απλή αναφορά του ονόματός της σε μία έκφραση. Έτσι η κλήση της συνάρτησης του προηγούμενου παραδείγματος μπορεί να γίνει ως εξής:

```
INPUT "X=",x
PRINT y(x)
```

Να σημειωθεί ότι επειδή μια συνάρτηση επιστρέφει μια τιμή με το όνομά της, αυτό πρέπει να συμφωνεί με τον τύπο του αποτελέσματος. Αν για παράδειγμα μια συνάρτηση επιστρέφει ως αποτέλεσμα μια αλφαριθμητική τιμή, τότε το όνομα της συνάρτησης πρέπει να είναι αλφαριθμητική μεταβλητή.

Η μεταβίβαση τιμών σε μία συνάρτηση γίνεται μέσω της λίστας ορισμάτων της καλούμενης συνάρτησης. Η σειρά των ορισμάτων και ο τύπος τους πρέπει να συμφωνεί με τις παραμέτρους στον ορισμό της συ-

νάρτησης. Σε μία συνάρτηση μπορούν να περνούν σταθερές, εκφράσεις, μεταβλητές, πίνακες και εγγραφές με το ίδιο τρόπο που γίνεται και στα υποπρογράμματα.

4.3 ΠΕΡΑΣΜΑ ΜΕΤΑΒΛΗΤΩΝ ΜΕ ΤΗ ΒΟΗΘΕΙΑ ΤΗΣ SHARED.

Πέρα από τη δυνατότητα να μεταβιβάζονται μεταβλητές μέσα από την εντολή CALL, η QuickBASIC παρέχει εναλλακτικούς τρόπους με τους οποίους περνούν μεταβλητές κατά την κλήση διαδικασιών:

1. Η ιδιότητα SHARED σε εντολές COMMON και DIM στο κύριο πρόγραμμα επιτρέπει τη χρήση των ορισμένων εκεί μεταβλητών από όλα τα υποπρογράμματα, δηλ. οι μεταβλητές αυτές ορίζονται ως καθολικές. Αυτή η μέθοδος χρησιμοποιείται όταν όλες οι μεταβλητές χρησιμοποιούνται από όλα τα υποπρογράμματα. Για να χρησιμοποιηθεί η ιδιότητα SHARED, πρέπει να τοποθετηθεί η λέξη SHARED αμέσως μετά τις εντολές COMMON ή DIM όπως στα παρακάτω παραδείγματα:

```
COMMON SHARED a,b,c
DIM SHARED Array(10,10)
```

Αν απαιτείται για μία ή περισσότερες μεταβλητές να δηλωθεί και ο τύπος τους, τότε οι προηγούμενες εντολές επεκτείνονται και με τη δήλωση AS πχ.

```
DIM SHARED i AS INTEGER
```

Δεν επιτρέπεται η διπλή δήλωση για μία μεταβλητή, δηλ αν η i έχει δηλωθεί ως ακέραια μεταβλητή με την DEFINT, τότε θα σημειωθεί λάθος στην εντολή COMMON [SHARED] i που τυχόν ακολουθεί.

2. Η εντολή SHARED επιτρέπει σ'ένα υποπρόγραμμα να χρησιμοποιήσει μεταβλητές οι οποίες έχουν δηλωθεί στο κύριο πρόγραμμα χωρίς την ιδιότητα SHARED. Αυτή η μέθοδος είναι πιο χρήσιμη όταν διαφορετικά υποπρογράμματα χρησιμοποιούν διαφορετικούς συνδυασμούς μεταβλητών του κύριου προγράμματος. Η εντολή SHARED έχει την ακόλουθη μορφή:

```
SHARED έβδδά_ιάδδääεçδρί
```

Μια απλή χρήση της SHARED φαίνεται στο παρακάτω παράδειγμα:

```
a = 1 : b = 2
CALL Prog2
```

```
SUB Prog2 STATIC
  SHARED a,b
  c = a + b
  PRINT c      ' άέδδδρίάδδää 3
END SUB
```

4.4 ΑΝΑΔΡΟΜΗ

Ο όρος αναδρομή αναφέρεται στη δυνατότητα μιας διαδικασίας να χρησιμοποιεί την ίδια τη διαδικασία στον ορισμό της δηλ. να μπορεί να καλεί τον εαυτό της. Οι διαδικασίες της QB μπορούν να είναι αναδρομικές. Κλασικό παράδειγμα αναδρομικής διαδικασίας είναι ο υπολογισμός του n! (n παραγοντικό). Το παραγοντικό ενός μη αρνητικού αριθμού n ορίζεται ως εξής:

$$n! = n \cdot (n-1)! \text{ και}$$

$$0! = 1$$

$$\text{Π.χ. διά } n=5 \quad \Rightarrow 5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$

Η αναδρομική συνάρτηση Fact υπολογισμού του n! υλοποιείται άμεσα στις επόμενες γραμμές.

```
DECLARE FUNCTION Fact# (n%)
INPUT "Άρδää äñέèìü äðü 1 Ýüð 20 :", n%
PRINT n%,Fact#(n%)
END
'
FUNCTION Fact#(n%) STATIC
  IF n%>0 THEN
    Fact# = n% * Fact#(n%-1)
  ELSE
    Fact# = 1
  END IF
END FUNCTION
```

Στην αναδρομική κλήση διαδικασιών πρέπει να λαμβάνονται ορισμένες προφυλάξεις. Πριν από κάθε κλήση μιας διαδικασίας φυλάσσονται σε μια στοίβα (stack) οι τιμές όλων των τοπικών μεταβλητών, ώστε μετά την επιστροφή να ανακληθούν και η διαδικασία να συνεχίσει από εκεί που σταμάτησε με τη σωστή "εικόνα". Το γεγονός αυτό μπορεί να εξαντλήσει τα περιθώρια της στοίβας, ιδιαίτερα αν γίνεται μεγάλος αριθμός κλήσεων ή χρησιμοποιούνται πολλές μεταβλητές. Στην περίπτωση αυτή προκαλείται μήνυμα Out of stack space. Το πρόβλημα αυτό μπορεί να διορθωθεί με την αύξηση του μεγέθους της στοίβας με τη χρήση της CLEAR,,μς. Οπου μς είναι το μέγεθος στοίβας, το οποίο προεκτιμάται ανάλογα με την πολυπλοκότητα της διαδικασίας.

4.5 ΕΛΕΓΧΟΣ ΜΕΤΑΒΛΗΤΩΝ ΜΕ ΤΗΝ ΕΝΤΟΛΗ DECLARE

Κατά την αποθήκευση επί του δίσκου προγραμμάτων που περιέχουν διαδικασίες, αυτόματα από το συντάκτη της QB παρεμβάλλονται στην αρχή ορισμένες εντολές DECLARE μία για κάθε χρησιμοποιούμενη διαδικασία. Κάθε τέτοια εντολή αποτελείται από τη λέξη DECLARE, ακολουθούμενη από τη SUB ή FUNCTION, το όνομα της διαδικασίας και ένα ζεύγος παρενθέσεων. Αν η διαδικασία έχει παραμέτρους, τότε η λίστα παραμέτρων παρεμβάλλεται μεταξύ των παρενθέσεων, αλλιώς παραμένουν κενές. Η λίστα παραμέτρων της DECLARE είναι ακριβώς ίδια με αυτή που χρησιμοποιείται στη γραμμή ορισμού της διαδικασίας SUB ή FUNCTION.