

# Σύντομη περιγραφή της γλώσσας προγραμματισμού ABC

Απόστολος Συρόπουλος  
E-mail: apostolo@obelix.ee.duth.gr ή  
apostolo@ocean1.ee.duth.gr

Νοέμβριος 1998

## 1 Τι είναι η ABC;

Η ABC είναι μια *διερμηνευόμενη* γλώσσα προγραμματισμού αλλά και ένα *προγραμματιστικό περιβάλλον*. Όταν λέμε *διερμηνευόμενη* εννοούμε ότι δεν χρειάζεται μετάφραση του προγράμματος μας σε γλώσσα μηχανής, αλλά αυτό εκτελείται απ' ευθείας από το προγραμματιστικό περιβάλλον. Η γλώσσα σχεδιάστηκε από μία ομάδα ερευνητών του *Κέντρου Μαθηματικών και της Επιστήμης των Ηλεκτρονικών Υπολογιστών* του Amsterdam, ως μια εναλλακτική λύση στο πρόβλημα της διδασκαλίας του προγραμματισμού σε νέους μαθητές, οι οποίοι μέχρι τότε χρησιμοποιούσαν την BASIC.

Τα κύρια χαρακτηριστικά της ABC είναι ότι είναι απλούστατη στη χρήση και εύκολη στην εκμάθηση. Έχει δε μεγαλύτερη εκφραστική δύναμη από την Pascal και την C! Επιπλέον, η γλώσσα διατίθεται δωρεάν μέσω του Internet. Περισσότερες πληροφορίες μπορείτε να βρείτε στο παρακάτω URL:

<http://www.cwi.nl/~steven/abc/>

## 2 Χρησιμοποιώντας την ABC

Όπως αναφέρθηκε η ABC είναι μια γλώσσα προγραμματισμού αλλά και ένα απλό σχετικά περιβάλλον. Για να ξεκινήσετε το περιβάλλον δεν έχετε παρά να πληκτρολογήσετε την λέξη abc, δηλ.:

```
C:\>abc
```

και το σύστημα ξεκινάει απαντώντας:

```
ABC Release 1.04.01.  
Copyright (c) Stichting Mathematisch Centrum, Amsterdam, 1991.  
Type '?' for help.  
>first  
>>> ?
```

Στην προτροπή (?) του συστήματος μπορούμε να γράψουμε κάποια εντολή. Το ενδιαφέρον με την ABC είναι ότι όταν ξενιθήσουμε να γράψουμε μια εντολή, τότε το σύστημα μας προτείνει κάποια εντολή: αν είναι η εντολή που θέλουμε δεν έχουμε παρά να πατήσουμε το πλήκτρο tab για να συμπληρωθεί η λέξη:

```
>>> W?RITE ? (πατάμε tab)
>>> WRITE ?
```

Προσέξτε ότι το πρώτο ? μπαίνει επειδή το σύστημα ουσιαστικά σας ρωτάει για το αν η επιλογή του είναι σωστή. Με τον τρόπο αυτό εύκολα και γρήγορα μπορούμε να πλητρολογήσουμε το πρόγραμμα. Ενώ και με το δεδομένο ότι η γλώσσα είναι πολύ εύκολη στην εκμάθηση της, η ζωή γίνεται πραγματικά... εύκολη!

### 3 Αριθμοί και λέξεις

Με την γλώσσα ABC μπορεί κανείς να επεξεργαστεί αριθμούς και λέξεις αλλά και πίνακες. Ενώ στις περισσότερες γλώσσες είναι αυστηρά περιορισμένο το εύρος των αριθμών που ένας κάποιος μπορεί να χρησιμοποιήσει (π.χ. από  $-2^{31}$  ως  $2^{32}$ ), αντιθέτως για τον χρήστη της ABC δεν υπάρχουν τέτοια τεχνητά όρια. Έτσι για παράδειγμα μπορούμε πολύ εύκολα να υπολογίσουμε πόσο κάνει  $2^{100}$ :

```
>>> WRITE 2**100
1267650600228229401496703205376
>>> ?
```

Η εντολή WRITE τυπώνει στην οθόνη του υπολογιστή το αποτέλεσμα μια αριθμητικής παράστασης ή ένα μήνυμα, όπως θα δούμε παρακάτω. Ας δούμε όμως πώς μπορούμε να υπολογίσουμε ορισμένες απλές αριθμητικές παραστάσεις. Έστω ότι θέλουμε να εκτελέσουμε τις βασικές αριθμητικές πράξεις για δύο αριθμούς (ή και άλλες αριθμητικές παραστάσεις) a και b, τότε:

Πράξη	Τρόπος γραφής
Πρόσθεση	a+b
Αφαίρεση	a-b
Πολλαπλασιασμός	a*b
Διαίρεση	a/b
Ύψωση σε δύναμη	a**b
Αντίθετος	-a
Τετραγωνική ρίζα	root a
Νιοστή ρίζα	n root a ( $\sqrt[n]{a}$ )
Υπόλοιπο ακέραιας διαίρεσης	a mod b

Φυσικά υπάρχουν και άλλες δυνατότητες αλλά αυτές θα τις δούμε στην συνέχεια. Για να εξασκηθούμε λίγο στην γραφή αριθμητικών παραστάσεων ας πούμε στην ABC να υπολογίσει τις εξής παραστάσεις:

(i)  $1 + 3 + 5 + 7 + 9 + 11$



Μια ενδιαφέρουσα προκαθορισμένη συνάρτηση είναι η `random` η οποία επιστρέφει ένα τυχαίο αριθμό μεταξύ του μηδενός και του ένα:

```
>>> WRITE random
0.4981999659931342
>>> WRITE random
0.5564988299241083
>>> WRITE random
0.5421311503130343
>>> WRITE random
0.7672725901223743
```

Σημειώστε ότι άμα θέλετε να μάθεται ποιές άλλες συναρτήσεις παρέχει η ABC, δεν έχετε παρά να πληκτρολογήσετε το σύμβολο `?` και να διαβάσετε τη βοήθεια που παρέχει το σύστημα.

Μέχρι τώρα είδαμε την χρήση αριθμών. Όμως η ABC μπορεί και διαχειρίζεται και *κορδόνια*, δηλ. κείμενα τα οποία *μπαίνουν* μεταξύ δύο " π.χ., "Hello World!". Τα κείμενα αυτά μπορούν να τυπωθούν, ενώ μπορούμε να εκτελέσουμε και κάποιες πράξεις. Ορίστε μερικά παραδείγματα:

```
>>> WRITE "hello mary!"
hello mary!
>>> WRITE "hello "^"mary!"
hello mary!
>>> WRITE "Wow! "^^5
Wow! Wow! Wow! Wow! Wow!
>>> WRITE upper("Wow! "^^5)
WOW! WOW! WOW! WOW! WOW!
>>> WRITE lower "THIS IS WONDERFUL DAY!"
this is wonderful day!
>>> ?
```

Στο πρώτο παράδειγμα απλά τυπώνουμε τη κορδόνι `hello marry`. Στο δεύτερο παράδειγμα δείχνουμε πώς είναι δυνατό να δημιουργήσει κανείς ένα νέο κορδόνι από δύο άλλα χρησιμοποιώντας τον τελεστή `^` (θεωρήστε ότι το σύμβολο αυτό αντιστοιχεί σε πρόσθεση κορδονιών). Στο τρίτο παράδειγμα φαίνεται η χρήση του τελεστή `^^`, ο οποίος δημιουργεί ένα νέο κορδόνι το οποίο αποτελείται από τόσα αντίγραφα του κορδονιού που βρίσκεται στα αριστερά του, όσα καθορίζει η παράσταση ή ο αριθμός που βρίσκεται στα δεξιά του. Τέλος, οι συναρτήσεις `upper` και `lower` μετατρέπουν το κορδόνι-όρισμα σε κεφαλαία ή πεζή μορφή αντίστοιχα.

Εκτός όμως από τις παραπάνω συναρτήσεις υπάρχουν ακόμη μερικές, οι οποίες όμως δεν προορίζονται αποκλειστικά για χρήση σε κορδόνια, αλλά και σε λίστες (βλ. ενότητα 5.1). Παρακάτω δίνουμε τυπικά παραδείγματα χρήσης των και επεξήγηση της λειτουργίας των.

```
>>> WRITE "nowhere"@3
where
>>> WRITE "somewhere"|4
some
>>> WRITE #"how many characters?"
20
```

```

>>> WRITE "a#" "how many characters?"
3
>>> WRITE min "abcdefg"
a
>>> WRITE max "abcdefg"
g
>>> WRITE "abcdefg" item 3
c
>>> WRITE choice "abcdefg"
d
>>> WRITE choice "abcdefg"
c

```

Η παράσταση  $t @ n$  επιστρέφει ό,τι απομένει από το κορδόνι  $t$  αν διαγράψουμε  $n - 1$  χαρακτήρες από την αρχή του. Ενώ η παράσταση  $t | n$  επιστρέφει τους πρώτους  $n$  χαρακτήρες του κορδονιού  $t$ . Αν θέλουμε να υπολογίσουμε τον αριθμό των χαρακτήρων που συναποτελούν ένα κορδόνι  $t$ , μπορούμε να χρησιμοποιήσουμε την παράσταση  $\#t$ . Αν τώρα θέλουμε να μάθουμε πόσες φορές υπάρχει ο χαρακτήρας  $c$  στο κορδόνι  $t$ , χρησιμοποιούμε την παράσταση  $c \# t$ . Είναι δυνατό να βρούμε είτε το μεγαλύτερο είτε το μικρότερο χαρακτήρα ενός κορδονιού  $t$  με τις παραστάσεις:  $\max t$  και  $\min t$  αντίστοιχα. Ενώ οι εντολές  $i \max t$  και  $i \min t$  επιστρέφουν το χαρακτήρα  $j$  που είναι ο αμέσως μικρότερος ή μεγαλύτερος αντίστοιχα από τον  $i$ , π.χ.,

```

>>> WRITE 'b' min 'abcdefg'
c
>>> WRITE 'f' max 'abcdefg'
e

```

Προσέξτε ότι μπορούμε να χρησιμοποιούμε και τα μονά εισαγωγικά. Στην περίπτωση που θέλουμε να εξάγουμε το υπ' αριθμόν  $i$  στοιχείο ενός κορδονιού  $t$ , χρησιμοποιούμε την εντολή  $t \text{ item } n$ . Τέλος, η παράσταση  $\text{choice } t$  επιστρέφει ένα τυχαίο χαρακτήρα του κορδονιού  $t$ .

## 4 Εντολή ανάθεσης

Υπάρχει μια εντολή με την οποία μπορεί κανείς να αποθηκεύσει σε μια προσωρινή θέση μνήμης μια πληροφορία. Στην απλούστερη περίπτωση η πληροφορία αυτή είναι είτε ένας αριθμός είτε ένα κορδόνι. Η θέση μνήμης στην οποία αποθηκεύουμε την πληροφορία αυτή ονομάζεται *μεταβλητή*. Ο δε μηχανισμός με τον οποίο αποθηκεύουμε τις πληροφορίες ονομάζεται *εντολή ανάθεσης*. Για να μπορούμε να ξεχωρίζουμε τις διάφορες μεταβλητές, τις δίνουμε συμβολικά ονόματα τα οποία έχουν ως πρώτο γράμμα πάντα ένα πεζό γράμμα της αγγλικής αλφαβήτας, ενώ οι υπόλοιποι χαρακτήρες μπορούν να είναι είτε πεζά γράμματα της αγγλικής, είτε ψηφία είτε μια τελεία. Έτσι για παράδειγμα το όνομα `s300` είναι επιτρεπτό όνομα μεταβλητής, ενώ το `s300?` όχι.

**Άσκηση 4.1** Από τα παρακάτω ονόματα ποια είναι επιτρεπτά ονόματα μεταβλητών και ποια όχι;

```
hello.world 1greece Oleg.1 maria
world!      life+love john's
```

Η εντολή ανάθεσης της ABC είναι πολύ απλή:

```
PUT τιμή IN μεταβλητή
```

όπου τιμή ένας αριθμός, ένα κορδόνι, κ.λπ., και μεταβλητή το όνομα μιας μεταβλητής, π.χ.:

```
>>> PUT 3 IN x
>>> WRITE x
3
>>> PUT x**2 IN x
>>> WRITE x
9
```

Παρατηρήστε ότι μπορούμε να χρησιμοποιούμε τις μεταβλητές και ως τιμές. Αν θέλουμε να αναθέσουμε σε πολλές μεταβλητές ταυτόχρονα κάποιες τιμές, μπορούμε να χρησιμοποιήσουμε την γενικότερη μορφή της εντολής ανάθεσης:

```
PUT τιμή1, τιμή2, ..., τιμήN IN μεταβλητή1, μεταβλητή2, ..., μεταβλητήN
```

Έτσι μπορούμε να ανταλλάξουμε τις τιμές δύο μεταβλητών με την εντολή

```
PUT b,a IN a,b
```

Αν διακόψουμε την χρήση του συστήματος της ABC, όλες οι μεταβλητές που έχουμε ορίσει αυτομάτως αποθηκεύονται όπως επίσης και η τιμή της κάθε μιας. Για να δούμε ποιες μεταβλητές υπάρχουν αποθηκευμένες στο χώρο εργασία μας, πληκτρολογούμε το σύμβολο ==:

```
>>> ==
x
```

Αν θέλουμε να διαγράψουμε οριστικά μια μεταβλητή, χρησιμοποιούμε την εντολή DELETE και γράφουμε δίπλα της χωριζόμενα με κόμμα τα ονόματα των μεταβλητών που θέλουμε να διαγράψουμε:

```
>>> DELETE x
>>> ==
>>> ?
```

## 5 Άλλες δομές δεδομένων

Στην επιστήμη των υπολογιστών όταν λέμε *δομή δεδομένων* αναφερόμαστε σε μηχανισμούς με τους οποίους μπορούμε να οργανώσουμε δεδομένα κατά τέτοιο τρόπο ώστε και εύχρηστα να είναι αλλά και φανερά να είναι η *φυσική* δομή που αντιπροσωπεύουν. Η γλώσσα ABC παρέχει τις λίστες, τους πίνακες και τις σύνθετες δομές ως μηχανισμούς οργάνωσης δεδομένων. Ας δούμε όμως τι είναι οι λίστες, τι οι πίνακες και τι οι σύνθετες δομές.

## 5.1 Λίστες

Μια λίστα χονδρικά μιλώντας είναι κάτι σε σύνολο με την διαφορά ότι μπορεί ένα στοιχείο να εμφανίζεται πολλές φορές. Όλα τα στοιχεία μιας λίστας θα πρέπει να είναι της ίδιας μορφής, π.χ., αριθμοί, κορδόνια, ή και λίστες οι οποίες με την σειρά τους θα πρέπει να περιέχουν στοιχεία της ίδιας μορφής κ.ο.κ. Τα στοιχεία μιας λίστας γράφονται μέσα σε άγκιστρα και χωρίζονται μεταξύ των με το ελληνικό ερωτηματικό, π.χ., {1; 2; 3; 4}. Οι τυπικές διαδικασίες που μπορούμε να επιτελέσουμε σε μία λίστα είναι οι ακόλουθες:

Διαδικασία	Παράδειγμα
Δημιουργία	PUT {'a'; 'b'; 'c'} IN letters
Προσθήκη στοιχείου	INSERT 'd' IN letters
Διαγραφή στοιχείου	REMOVE 'b' FROM letters
Πλήθος στοιχείων	#letters
Πλήθος εμφανίσεων στοιχείου	'd'#letters

Είναι ενδιαφέρον να πούμε ότι γλώσσα ABC αυτόματα ταξινομεί τα στοιχεία που την αποτελούν:

```
>>> WRITE {3; 1; 2}
{1; 2; 3}
```

Αυτό είναι πάρα πολύ χρήσιμο αφού δεν υπάρχει έτσι λόγος να ταξινομούμε δεδομένα. Επιπλέον, αν οι τιμές μιας λίστας είναι διαδοχικοί αριθμοί ή χαρακτήρες, μπορούμε να μην γράψουμε όλους τους αριθμούς ή χαρακτήρες χρησιμοποιώντας εύρη, δηλ. γράφουμε το μικρότερο αριθμό, δύο τελείες και μετά τον μεγάλο αριθμό:

```
>>> PUT {1; 2; 3; 4; 5} IN x
>>> PUT {1 .. 5} IN y
>>> WRITE x
{1; 2; 3; 4; 5}
>>> WRITE y
{1; 2; 3; 4; 5}
```

## 5.2 Πίνακες

Ενώ σε μία λίστα δεν υπάρχει κάποιος ιδιαίτερος μηχανισμός για να χρησιμοποιήσουμε κάποιο στοιχείο της, οι πίνακες αριθμούν όλα τα στοιχεία και έτσι κανείς μπορεί εύκολα να χρησιμοποιήσει οποιοδήποτε στοιχείο του. Αν ένας πίνακας έχει, π.χ., δέκα στοιχεία και ονομάζεται count, τότε αν θέλουμε να τυπώσουμε στην οθόνη το έβδομο στοιχείο του γράφουμε WRITE count[7]. Οι πιο συνηθισμένες διαδικασίες που μπορούμε να επιτελέσουμε σε ένα πίνακα είναι οι εξής:

Διαδικασία	Παράδειγμα
Δημιουργία	PUT {} IN count
Προσθήκη στοιχείων	FOR l IN letters: PUT 0 IN count[l]
Μετατροπή στοιχείου	FOR l IN letters: PUT count[l]+1 IN count[l]

Προσέξτε ότι στα παραδείγματα μας τα στοιχεία του πίνακα αριθμούνται με αγγλικά γράμματα. Επιπλέον είναι δυνατό να αριθμούνται και με λέξεις:

```
PUT {} IN price
PUT 3.50 IN price['jam']
PUT 1.87 IN price['bread']
PUT 2.45 IN price['butter']
```

Δώστε τώρα την εντολή `WRITE price` για να δείτε πως αντιλαμβάνεται η ABC τους πίνακες. Τέλος, αξίζει να προσθέσουμε πως άμα θέλουμε να σπάσουμε ένα μεγάλο κορδόνι στις λέξεις που το αποτελούν, τότε μπορούμε να χρησιμοποιήσουμε την προκαθορισμένη συνάρτηση `split` όπως δείχνει το παρακάτω παράδειγμα:

```
>>> PUT split "now here there" IN words
>>> WRITE words
{[1]: "now"; [2]: "here"; [3]: "there"}
>>> WRITE words[1]
now
```

### 5.3 Σύνθετες δομές

Οι σύνθετες δομές μας επιτρέπουν να αποθηκεύουμε σε μία μεταβλητή δεδομένα διαφορετικής μορφής. Τα δεδομένα αυτά θα πρέπει να γράφονται μεταξύ παρενθέσεων και να χωρίζονται με κόμμα. Αν θέλουμε να πάρουμε τα στοιχεία που αποτελούν μια σύνθετη δομή, χρησιμοποιούμε μια εντολή ανάθεσης σε τόσες μεταβλητές όσα είναι τα στοιχεία του πίνακα. Όλες αυτές οι διαδικασίες φαίνονται στο παράδειγμα που ακολουθεί:

```
>>> PUT (root 2, "help") IN mimis
>>> WRITE mimis
(1.414213562373095, "help")
>>> PUT mimis IN n, ch
>>> WRITE n
1.414213562373095
>>> WRITE ch
help
```

## 6 Εντολές επανάληψης

Μια εντολή επανάληψης είναι ικανή να επαναλαμβάνει κάποιες εντολές ή μία μόνο εντολή ένα αριθμό φορές. Γενικά υπάρχουν εντολές επανάληψης στις οποίες ο αριθμός εξαρτάται από μία λογική συνθήκη, ή είναι απλά προκαθορισμένος. Η ABC παρέχει την εντολή `WHILE` ως εντολή εξαρτούμενη από λογική συνθήκη και την εντολή `FOR` ως εντολή επανάληψης με προκαθορισμένο αριθμό επαναλήψεων. Πριν πούμε πως λειτουργούν οι εντολές αυτές ας δούμε τι σημαίνει λογική συνθήκη.

Είναι γνωστό από τα μαθηματικά ότι μπορούμε να συγκρίνουμε αριθμούς. Έτσι ο αριθμός 3 είναι μεγαλύτερος από τον αριθμό 1, ενώ ο αριθμός  $-1$  είναι μικρότερος από τον αριθμό 1, κ.λπ.

Αν γράψουμε  $3 > 2$ , τότε *ισχυριζόμαστε* ότι ο αριθμός 3 είναι μεγαλύτερος από τον αριθμό 2. Στην περίπτωση αυτή *ισχυριζόμαστε* κάτι *αληθές*. Όμως υπάρχουν περιπτώσεις που αυτό δεν ισχύει, π.χ., ο ισχυρισμός  $-1 > 1$  είναι σαφώς *ψευδής*. Έτσι λοιπόν ένα ισχυρισμός είναι είτε αληθής είτε ψευδής<sup>1</sup>. Με την ABC μπορούμε να εκφράσουμε κύρια αριθμητικούς ισχυρισμούς, των οποίων την αλήθεια μπορεί να ελέγξει η γλώσσα. Οι ισχυρισμοί αυτοί είναι:

τιμήΑ < τιμήΒ	η τιμήΑ είναι μικρότερη από την τιμήΒ
τιμήΑ = τιμήΒ	η τιμήΑ είναι ίση με την τιμήΒ
τιμήΑ > τιμήΒ	η τιμήΑ είναι μεγαλύτερη από την τιμήΒ
τιμήΑ <= τιμήΒ	η τιμήΑ είναι μικρότερη ή ίση από την τιμήΒ
τιμήΑ >= τιμήΒ	η τιμήΑ είναι μεγαλύτερη ή ίση από την τιμήΒ
τιμήΑ <> τιμήΒ	η τιμήΑ δεν είναι ίση με την τιμήΒ

Προσέξτε ότι λέμε τιμήΑ και τιμήΒ γιατί δεν είναι απαραίτητο να είναι απλοί αριθμοί, αλλά γενικότερα αλγεβρικές παραστάσεις, κορδόνια, σύνθετες δομές, κ.ο.κ. Επιπλέον είναι δυνατό να έχουμε και συνδυασμό των παραπάνω ισχυρισμών, π.χ., ο ισχυρισμός  $0 < 1 < 2$  είναι απολύτα κατανοητός από την ABC. Ο προηγούμενος ισχυρισμός δηλώνει ότι  $0 < 1$  και  $1 < 2$ . Η δυνατότητα δημιουργίας ισχυρισμών με την χρήση της λέξης και ονομάζεται *σύζευξη*. Αν το και αντικατασταθεί με την λέξη ή, τότε δημιουργούμε *διάζευξη*. Ενώ, τέλος, μπορούμε να δημιουργήσουμε την *άρνηση* ενός ισχυρισμού, δηλ. η άρνηση μιας πρότασης, με την χρήση της λέξης *δεν*. Οι τρεις αυτές λογικές πράξεις, όπως ονομάζονται, είναι κατανοητές από την ABC μέσω ειδικών λέξεων:

**Σύζευξη** Αν έχουμε δύο ισχυρισμούς  $i_1$  και  $i_2$ , τότε η σύζευξη των σημειώνεται με τον εξής τρόπο:  $i_1$  AND  $i_2$ , π.χ.,  $(x > 1)$  AND  $(x < 5)$ .

**Διάζευξη** Αν έχουμε δύο ισχυρισμούς  $i_1$  και  $i_2$ , τότε η διάζευξη των σημειώνεται με τον εξής τρόπο:  $i_1$  OR  $i_2$ , π.χ.,  $(x > 1)$  OR  $(x < 5)$ .

**Άρνηση** Αν έχουμε ένα ισχυρισμό  $i$ , τότε η άρνηση του σημειώνεται με τον εξής τρόπο: NOT  $i$ , π.χ., NOT  $(x > 7)$ .

Η σύζευξη δύο ισχυρισμών είναι αληθής αν και μόνο αν και οι δύο ισχυρισμοί που την απαρτίζουν είναι αληθείς. Σε κάθε άλλη περίπτωση είναι ψευδής. Η διάζευξη δύο ισχυρισμών είναι αληθής αν και μόνο αν τουλάχιστον ένας ισχυρισμός από τους δύο είναι αληθής. Προφανώς, είναι ψευδής μόνο όταν και οι δύο ισχυρισμοί είναι ψευδής. Τέλος, άμα ένας ισχυρισμός είναι αληθής, τότε η άρνηση του είναι ψευδής και τουνάπαλιν. Ας επανέλθουμε όμως στο θέμα των εντολών επανάληψης.

Η εντολή WHILE συντάσσεται με πολύ απλό τρόπο, γράφουμε την λέξη WHILE, βάζουμε το λιγότερο ένα κενό και έπειτα γράφουμε την λογική συνθήκη. Στη συνέχεια πληκτρολογούμε τις εντολές που θέλουμε να εκτελεστούν, πατώντας απλά δύο φορές το enter μετά την τελευταία εντολή. Η WHILE εκτελεί τις εντολές και συνεχίζει μέχρι η συνθήκη να γίνει ψευδής. Όμως ας δούμε ένα παράδειγμα ότι να γίνουμε κατανοητοί:

<sup>1</sup> Φυσικά υπάρχουν περιπτώσεις που η σχετικά ακραία αυτή θέση δεν έχει νόημα, όπως είναι για παράδειγμα η περίπτωση του ισχυρισμού ότι *ένας άνθρωπος ύψους 1,80 m είναι ψηλός...*

```

>>> PUT 0 IN x
>>> WHILE x<10:           WRITE x
        PUT x+1 IN x

0 1 2 3 4 5 6 7 8 9

```

Αρχικά δίνουμε την τιμή 0 στη μεταβλητή x. Οι δύο εντολές που καλείται να εκτελέσει η εντολή WHILE είναι αυτές που έχουν γραφτεί λίγο δεξιότερα από τις άλλες. Η εντολή WHILE κάθε φορά ελέγχει η τιμή της μεταβλητής x είναι μικρότερη από τον αριθμό 10. Μόλις η τιμή της μεταβλητής, η οποία αλλάζει συνέχεια ως συνέπεια της εντολής PUT, πάρει την τιμή 10, η επανάληψη σταματά.

**Άσκηση 6.1** Πόσες φορές θα εκτελεστεί η εντολή WHILE αν η συνθήκη είναι: (α') η  $x \leq 13$  και (β') η  $1 < x < 10$ ;

Η εντολή FOR εκτελείται πάντα ένα προκαθορισμένο φοράν. Αμέσως μετά την λέξη FOR βάζουμε το όνομα μιας μεταβλητής και έπειτα μια λίστα. Αμέσως μετά γράφουμε τις εντολές που θέλουμε να εκτελέσει κατ' επανάληψη η εντολή. Ο τρόπος που δηλώνουμε το τέλος των εντολών είναι να πατήσουμε δύο φορές το πλήκτρο enter. Ας δούμε όμως πως μπορούμε να ξαναγράψουμε την προηγούμενη επανάληψη χρησιμοποιώντας την εντολή FOR:

```

>>> FOR x IN {1; 2; 3; 4; 5; 6; 7; 8; 9}:
        WRITE x

1 2 3 4 5 6 7 8 9

```

Όμως παρατηρούμε ότι οι αριθμοί γράφονται όλοι σε μία γραμμή, αν θέλαμε να εμφανίζονται ο καθένας σε μία γραμμή μόνος του, τότε θα έπρεπε να γράφουμε την εντολή εκτύπωσης ως εξής: WRITE x /. Η πλάγια γραμμή στο τέλος δηλώνει ότι επιθυμούμε να γίνει αλλαγή γραμμής μετά το πέρας της εκτύπωσης των πληροφοριών που πρέπει να τυπωθούν.

## 7 Εντολές ελέγχου

Ο όρος *εντολές ελέγχου* αναφέρεται σ' εντολές οι οποίες εκτελούν κάποιες άλλες εντολές ανάλογα με το αν κάποια συνθήκη είναι αληθής ή ψευδής. Φυσικά το ίδιο πράγμα κάνει και η εντολή WHILE, μόνο που οι εντολές που μας ενδιαφέρουν εδώ εκτελούν τις εντολές μία μόνο φορά. Η πιο γνωστή εντολή ελέγχου είναι η εντολή IF, η οποία αποτελείται από δύο μέρη. Το πρώτο είναι ένας λογικός ισχυρισμός και το δεύτερο μια εντολή η οποία θα εκτελεστεί στην περίπτωση που ο ισχυρισμός είναι αληθής. Ορίστε ένα απλό παράδειγμα:

```

>>> IF x<>y: WRITE "they are not equal!"
        they are not equal!

```

Επειδή οι μεταβλητές x και y περιέχουν πληροφορίες που δεν είναι ίσες, ο ισχυρισμός  $x <> y$ , δηλ. η τιμή της x είναι *διάφορη* από την τιμή της y, είναι αληθής και κατά συνέπεια εκτελείται η εντολή WRITE "they are not equal!". Αν αλλάζουμε τον ισχυρισμό δείτε τι συμβαίνει:

```
>>> IF x=y: WRITE "they are equal!"
>>>
```

Απλά τίποτα! Όμως η εντολή IF δεν είναι η μόνη εντολή ελέγχου που παρέχει η ABC. Η εντολή SELECT είναι ουσιαστικά μια γενικευμένη μορφή της εντολής IF. Η γενική μορφή της εντολής φαίνεται παρακάτω:

```
SELECT:
  ισχυρισμός1: εντολή1
  ισχυρισμός2: εντολή2
      :
  ισχυρισμόςn: εντολήn
ELSE:      εντολήn+1
```

Αν κάποιος από τους  $n$  ισχυρισμούς είναι αληθής, τότε εκτελείται η εντολή που τον συνοδεύει. Ο ισχυρισμός ELSE δεν είναι υποχρεωτικό να συμπεριλαμβάνεται σε μία εντολή SELECT. Αν όμως υπάρχει, τότε εκτελείται η συνοδός εντολή του εφόσον κανένας άλλος ισχυρισμός δεν είναι αληθής. Στην εξαιρετική περίπτωση που κανένας ισχυρισμός δεν είναι αληθής και επίσης δεν υπάρχει ο ισχυρισμός ELSE, τότε η ABC σταματά την εκτέλεση της εντολής και διαμαρτύρεται! Ορίστε ένα απλό παράδειγμα χρήσης της εντολής:

```
>>> SELECT:
      x=y: WRITE "x=y"
      x<>y: WRITE "x<>y"

x<>y
```

## 8 Δημιουργία νέων εντολών και συναρτήσεων

Μέχρι τώρα μάθαμε να χρησιμοποιούμε τις εντολές που μας παρέχει η γλώσσα ABC, για να υπολογίσουμε κάποια πράγματα. Αν όμως για κάποιο υπολογισμό χρησιμοποιούμε κάποιο κομμάτι κώδικα πολλές φορές και σε διαφορετικά σημεία, είμαστε υποχρεωμένοι να πληκτρολογούμε πολλές φορές το ίδιο πράγμα. Αυτό έχει ως συνέπεια η διαδικασία γραφής ενός προγράμματος να γίνεται εξαιρετικά δυσχερής. Η λύση που δίνεται σε αυτό το πρόβλημα είναι η δυνατότητα δημιουργίας νέων εντολών με τη χρήση κάποιου μηχανισμού που παρέχει η ίδια η γλώσσα, που δεν είναι άλλος από την οικογένεια δηλώσεων HOW TO.

Υπάρχουν τρία είδη δηλώσεων HOW TO: αυτές που δημιουργούν νέες εντολές, όπως για παράδειγμα η εντολή WRITE, αυτές που δημιουργούν νέες συναρτήσεις, όπως για παράδειγμα η συνάρτηση sin η οποία υπολογίζει το ημίτονο ενός αριθμού και αυτές που δημιουργούν νέες εντολές ελέγχου. Τις τελευταίες δεν θα τις μελετήσουμε καθόλου.

### 8.1 Νέες εντολές

Όπως αναφέρθηκε η δημιουργία νέων εντολών γίνεται με την χρήση της εντολής HOW TO. Αμέσως μετά τις δύο αυτές λέξεις γράφουμε το όνομα τις νέας εντολής το οποίο όμως δεν είναι

υποχρεωτικό να είναι μια λέξη, αλλά μπορεί να είναι και μια σύντομη πρόταση· για παράδειγμα αν θέλουμε να δημιουργήσουμε μια νέα εντολή που θα μετατρέπει μια θερμοκρασία σε βαθμούς Celsius από βαθμούς Fahrenheit, τότε το όνομα PRINT CELSIUS είναι σαφώς καλύτερο από μια απλή λέξη. Επιπλέον, θα πρέπει να υπάρχει και μία παράμετρος η οποία θα είναι η θερμοκρασία σε βαθμούς Fahrenheit που θα μετατραπεί σε βαθμούς Celsius. Ας δούμε το πρώτο ολοκληρωμένο παράδειγμα δημιουργίας μιας νέας εντολής:

```
>>> HOW TO PRINT CELSIUS f:
HOW TO PRINT CELSIUS f:
    PUT (f-32)*5/9 IN c
    WRITE c
```

Όπως βλέπετε αρχικά βάζουμε την εντολή HOW TO, μετά το όνομα της νέας εντολής και μετά την/τις παραμέτρους. Αμέσως μετά γράφουμε σε τι θα αντιστοιχεί η κλήση της εντολής, δηλ. όταν στην προτροπή της γλώσσας θα γράφουμε PRINT CELSIUS t, όπου t ένας αριθμός ή μία μεταβλητή, ποιές θα είναι οι εντολές που θα εκτελούνται:

```
>>> PRINT CELSIUS 10
-12.22222222222222
>>> PRINT CELSIUS 12
-11.11111111111111
>>> PRINT CELSIUS 120
48.88888888888889
>>> ?
```

Βλέπουμε λοιπόν ότι η εντολή PRINT CELSIUS λειτουργεί ως να ήταν μια κανονική εντολή της γλώσσας. Ας δούμε όμως ένα πιο πολύπλοκο παράδειγμα: Έστω ότι θέλουμε να δημιουργήσουμε μια εντολή η οποία θα μετατρέπει ένα εύρος βαθμών Fahrenheit σε βαθμούς Celsius:

```
>>> HOW TO PRINT CELSIUS FROM a TO b:
HOW TO PRINT CELSIUS FROM a TO b:
    PUT a, b IN lo, hi
    IF lo > hi:
        PUT hi, lo IN lo, hi
    FOR f IN {lo..hi}:
        PUT (f-32)*5/9 IN c
        WRITE f, "Fahrenheit =", 2 round c, "Celsius" /
```

Προσέξτε ότι εδώ έχουμε δύο παραμέτρους την a και τη b και ότι το όνομα της εντολής μοιάζει πιο πολύ με πρόταση τώρα. Ορίστε και ένα απλό παράδειγμα χρήσης της νέας εντολής:

```
>>> PRINT CELSIUS FROM 10 TO 20
10 Fahrenheit = -12.22 Celsius
11 Fahrenheit = -11.67 Celsius
12 Fahrenheit = -11.11 Celsius
13 Fahrenheit = -10.56 Celsius
14 Fahrenheit = -10.00 Celsius
```

```
15 Fahrenheit = -9.44 Celsius
16 Fahrenheit = -8.89 Celsius
17 Fahrenheit = -8.33 Celsius
18 Fahrenheit = -7.78 Celsius
19 Fahrenheit = -7.22 Celsius
20 Fahrenheit = -6.67 Celsius
```

Δίνουμε τώρα μερικά ακόμη παραδείγματα δημιουργίας νέων εντολών, οι οποίες έχουν πολύ ενδιαφέροντα αποτελέσματα:

```
>>> HOW TO PREPEND head TO text:
HOW TO PREPEND head TO text:
  PUT head^text IN text
>>> HOW TO REVERSE text:
HOW TO REVERSE text:
  PUT "" IN reverse
  FOR c IN text: PREPEND c TO reverse
  WRITE reverse
>>>?
```

Αν τώρα δώσουμε την εντολή REVERSE "solaris" η γλώσσα θα απαντήσει siralos, δηλ. έχουμε δημιουργήσει μια εντολή η οποία γράφει ανάποδα την λέξη ή πρόταση που τις δίνουμε. Ας δούμε όμως τι ακριβώς κάνει η κάθε νέα εντολή:

- Η εντολή PREPEND τοποθετεί στην αρχή της λέξεως text την λέξη tail.
- Η εντολή REVERSE δημιουργεί μια νέα λέξη, την reverse, η οποία είναι κενή αρχικά. Έπειτα διατρέχοντας όλους τους χαρακτήρες της με μία εντολή FOR, τοποθετεί στην αρχή της λέξεως reverse τον κάθε χαρακτήρα. Τελικά τυπώνουμε τη νέα λέξη.

## 8.2 Νέες συναρτήσεις

Γενικά μιλώντας όταν λέμε συνάρτηση αναφερόμαστε σ' ένα μηχανισμό με τον οποίο μπορούμε να αντιστοιχούμε τιμές από ένα σύνολο σε τιμές από ένα άλλο σύνολο. Για παράδειγμα η συνάρτηση  $f(x) = x^2$  αντιστοιχεί σε κάθε αριθμό το τετράγωνό του, ενώ η συνάρτηση  $g(x, y) = x^y$  αντιστοιχεί τους αριθμούς  $x$  και  $y$  στη παράσταση  $x^y$ . Βλέπουμε λοιπόν ότι μια συνάρτηση μπορεί να αντιστοιχεί περισσότερα του ενός στοιχεία του συνόλου ορισμού, δηλ. του συνόλου από το οποίο παίρνουμε στοιχεία, σ' ένα στοιχείο του συνόλου τιμών, δηλ. του συνόλου στο οποίο αντιστοιχούμε τα στοιχεία του συνόλου ορισμού. Φυσικά τα δύο αυτά σύνολα μπορεί να είναι το ίδιο σύνολο.

Για την ABC κάθε συνάρτηση θα πρέπει να έχει ένα όνομα και να καθορίζουμε τον αριθμό των παραμέτρων της (2 το μέγιστο), δηλ. το πόσα στοιχεία του συνόλου ορισμού αντιστοιχούμε σ' ένα στοιχείο του συνόλου τιμών. Δίνουμε αμέσως τον ορισμό μιας συνάρτησης που επιστρέφει το τετράγωνο του ορίσματός της:

```
>>> HOW TO RETURN f x:
HOW TO RETURN f x:
  RETURN x*x
```

Αν τώρα γράψουμε την εντολή `WRITE f 4` το σύστημα θα μας τυπώσει 16, δηλ. το τετράγωνο του 4. Ας δούμε τώρα το πως δηλώνουμε μια συνάρτηση με δύο παραμέτρους. Καταρχήν θα πρέπει να πούμε ότι η ABC θεωρεί ότι οι συναρτήσεις είναι κάτι σαν τελεστές, κάτι δηλαδή σαν τα σύμβολα `+`, `-`, `*`, `/`, κ.λπ. Έτσι το όνομα της συνάρτησης γράφεται μετά το όνομα της πρώτης μεταβλητής και φυσικά πριν το όνομα της δεύτερης μεταβλητής. Ας δούμε ένα απλό παράδειγμα ώστε να γίνουμε πιο κατανοητοί:

```
>> HOW TO RETURN x pyth y:
HOW TO RETURN x pyth y:
    RETURN root (x**2 + y**2)

>>> WRITE 4 pyth 3
5
```

**Άσκηση 8.1** Σχεδιάστε μια συνάρτηση η οποία θα υπολογίζει το μήκος περιφέρειας ενός κύκλου με ακτίνα `R`. Σημειώστε ότι η προκαθορισμένη μεταβλητή `pi` της ABC περιέχει την τιμή του αριθμού  $\pi$ .

## 9 Εντολή εισόδου

Σε αρκετές περιπτώσεις θέλουμε μια διαδικασία να μπορεί να ζητά από τον χρήστη της να δίνει κάποια δεδομένα από το πληκτρολόγιο. Για τον λόγο αυτό η ABC παρέχει την εντολή `READ`, της οποίας η γενική μορφή φαίνεται παρακάτω:

```
READ απλή-διεύθυνση EG απλή-παράσταση
READ απλή-διεύθυνση RAW
```

όπου απλή-διεύθυνση είναι είτε το όνομα μιας μεταβλητής, είτε ένα στοιχείο ενός πίνακα και απλή-παράσταση ένας αριθμός, ένα κορδόνι, κ.λπ. Με την πρώτη μορφή μπορούμε να διαβάζουμε πολύπλοκες μορφές δεδομένων όπως σύνθετα δεδομένα, αλλά και απλά όπως αριθμούς. Με την δεύτερη μορφή μπορούμε να διαβάζουμε ολόκληρες γραμμές. Ας δούμε μερικά παραδείγματα για να γίνουμε κατανοητοί:

```
>>> READ x EG 0
76
>>> WRITE x
76
>>> READ y RAW
this is a nice day my friend!
>>> WRITE y
this is a nice day my friend!
>>> READ z EG (0, "a")
(12, "help")
>>> WRITE z
(12, "help")
```

Βλέπουμε ότι μεταβλητή `x` είναι ένας αριθμός, η μεταβλητή `y` μια ολόκληρη γραμμή κειμένου, η οποία δεν χρειάζεται εισαγωγικά, και τέλος η μεταβλητή `z` μια σύνθετη δομή.