# Κεφάλαιο 3

# ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ FORTRAN

## 3.1 Εισαγωγή

Η *FORTRAN* είναι μια γλώσσα προγραμματισμού κυρίως για μαθηματικές εφαρμογές. Η λέξη FORTRAN προέρχεται από τα αρχικά των λέξεων *FORmula TRANslation* που σημαίνει «Μετάφραση Τύπου». Η FORTRAN ήταν η πρώτη γλώσσα προγραμματισμού «υψηλού επιπέδου». Παρουσιάστηκε από την IBM την δεκατεία του 1950 και από τότε έχουν βγεί πολλές εκδόσεις της γλώσσας αυτής. Κάθε καινούργια έκδοση της FORTRAN συμβολίζεται με τα 2 τελευταία ψηφία τους έτους που παρουσιάστηκε. Έτσι έχουμε μέχρι τώρα : FORTRAN 66, FORTRAN 77, FORTRAN 90. Η πιό διαδεδομένη έκδοση της FORTRAN ακόμα και σήμερα είναι η FORTRAN 77, παρόλο που η FORTRAN 90 γίνεται ολοένα και πιό δημοφιλής. Επίσης υπάρχουν και εκδόσεις της FORTRAN αποκλειστικά και μόνο για παράλληλους υπολογιστές όπως η «FORTRAN Υψηλής Απόδοσης»(HPF).

Η FORTRAN είναι η κυρίαρχη γλώσσα προγραμματισμού για εφαρμογές που προέρχονται κυρίως από φυσικά προβλήματα, κατασκεύες, μηχανικά προβλήματα. Πολλοί είχαν και έχουν προβλέψει ότι η FORTRAN θα εξαλειφθεί σε λίγα χρόνια αλλά παραμένει η γλώσσα με την μεγαλύτερη αντοχή από όλες τις γλώσσες προγραμματισμού. Ένας από τους βασικούς λόγους για τους οποίους δεν έχει εγκαταλειφθεί είναι ότι υπάρχει πολύ λογισμικό γραμμένο σε FORTRAN , το οποίο δεν είναι εύκολο να αντικατασταθεί. Επιπλέον η FORTRAN έχει το μεγάλο πλεονέκτημα ότι έχει γίνει παγκόσμιο στάνταρτ. Αν ένας κώδικας έχει γραφτεί σε FORTRAN 77 τότε θα «τρέξει» σε οποιοδήποτε υπολογιστή που έχει τον αντίστοιχο μεταγλωττιστή(compiler).

## 3.2 Τα βασικά

Ένας κώδικας FORTRAN δεν είναι τίποτα άλλο παρά μια ακολουθία εντολών. Οι εντολές αυτές έχουν κάποιο συγκεκριμμένο συντακτικό. Ξεκινάμε με ένα παράδειγμα. Το παρακάτω πρόγραμμα διαβάζει ένα πραγματικό αριθμό r, υπολογίζει το εμβαδό του κύκλου με ακτίνα r και τυπώνει το αποτέλεσμα :

```
c234567
      program circle
      real r,area
c
c Read the real number r
c
      read*,r
c
c Compute the area of the circle
c
      area=3.14159*r*r
```



```
c
c Print the result
c
      print*, area
c
c End of the program
c
      stop
      end
```

Οι γραμμές που ξεκινούν με το γράμμα «c» είναι σχόλια και ο αποκλειστικός τους σκοπός είναι να κάνουν το πρόγραμμα πιό κατανοητό και ευανάγνωστο για το χρήστη. Αρχικά οι κώδικες FORTRAN γράφονταν με κεφαλαία γράμματα, αλλά έχει επικρατήσει τα τελευταία χρόνια οι κώδικες να γράφονται με μικρά γράμματα, πράγμα που θα ακολουθήσουμε σε αυτό το κεφάλαιο.

Ένας κώδικας FORTRAN αποτελείται από το κυρίως πρόγραμμα και ίσως από ένα ή περισσότερα υποπρογράμματα. Προς το παρόν θα ασχοληθούμε με το κυρίως πρόγραμμα, ενώ για τα υποπρογράμματα θα αναφερθούμε σε ξεχωριστή παράγραφο. Η δομή του κυρίως προγράμματος είναι η ακόλουθη

```
      program name

      declarations

      statements

      stop
      end
```

**Βασικός Κανόνας.** Ο πιό βασικός κανόνας στη FORTRAN είναι ο *Κανόνας Στηλών (ΚΣ)*. Πιο συγκεκριμμένα έχουμε ότι :

Στήλη 1η : Είτε κενή είτε υπάρχει ο χαρακτήρας c.

Στήλες 2η-5η : Ετικέτες εντολών (αν υπάρχουν).

Στήλη 6η : Χαρακτήρας συνέχειας της προηγούμενης γραμμής.

Στήλες 7η-72η : Εντολές του προγράμματος.

Όπως βλέπουμε ένας κώδικας FORTRAN ξεκινά με την εντολή program με την οποία δίνουμε ένα όνομα στο προγραμμά μας. Κατόπιν ακολουθούν οι *δηλώσεις declarations* των τύπων των μεταβλητών που χρησιμοποιούνται και στη συνέχεια οι *εντολές(statements)* του κώδικα. Οι περισσότερες εντολές ξεκινούν με 6 κενά και τελειώνουν πριν την 72η στήλη. Ο κώδικας ολοκληρώνεται με τις εντολές stop και end. Η εντολή stop είναι προεραιτική μιας και το πρόγραμμα θα σταματήσει έτσι και αλλιώς όταν εκτελεστεί η εντολή end, συνίσταται όμως να τερματίζουμε την ροή του προγράμματος με την εντολή stop δίνοντας έτσι έμφαση για το που σταματά η ροή της εκτέλεσης.

### 3.2.1 Σχόλια

Η γραμμή που ξεκινά έχοντας στη 1η της στήλη το χαρακτήρα c (comment), είναι *σχόλιο*. Τα σχόλια μπορούν να μπουν οπουδήποτε στο πρόγραμμα. Τα σχόλια σε ένα κώδικα είναι απαραίτητα προκειμένου να μπορούμε ανά πάσα στιγμή να καταλάβουμε τι κάνει ο κώδικας ή κάποιο κομμάτι αυτού και πως μπορούμε να τον χρησιμοποιήσουμε, (βλ. παράδειγμα)

### 3.2.2 Συνέχιση γραμμής

Όπως είδαμε από τον ΚΣ οι εντολές ενός κώδικα μπορεί να εκτείνονται εως και την 72η στήλη. Σε πολλές όμως περιπτώσεις οι 72 στήλες δεν είναι αρκετές ή για κάποιο λόγο θα θέλαμε να χωρίσουμε μια εντολή σε 2 γραμμές. Σε τέτοιες περιπτώσεις υπάρχει η δυνατότητα να συνεχίσουμε στη αμέσως επόμενη γραμμή προσθέοντας στη 6η στήλη της γραμμής αυτής τον *Χαρακτήρα Συνέχειας* που μπορεί να είναι ένας οποιοσδήποτε χαρακτήρας. Συνηθίζεται να χρησιμοποιουμε τα εξής σύμβολα : +, & , π.χ.

```
c234567(This is an easy way to know the column number)
c
c The next statement expands in 2 lines
      area=3.14159265358979
     +    *r*r
```

### 3.2.3 Κενά

Κενά μπορούν να υπάρχουν σε οποιοδήποτε σημείο του προγράμματος χωρίς να δημιουργείται κανένα απολύτως συντακτικό πρόβλημα. Τα κενά απλώς αγνοούνται από τον μεταγλωττιστή της FORTRAN .

## 3.3 Μεταβλητές, Τύποι μεταβλητών, Δηλώσεις

### 3.3.1 Ονόματα μεταβλητών

Τα ονόματα των *μεταβλητών* στη FORTRAN αποτελούνται από 1-6 χαρακτήρες που συνήθως είναι είτε τα γράμματα a-z είτε οι αριθμοί 0-9. Ο πρώτος χαρακτήρας κάθε μεταβλητής πρέπει να είναι **απαραιτήτως γράμμα**. Η FORTRAN 77 δεν ξεχωρίζει μεταξύ μικρών και κεφαλαίων γραμμάτων και στην πραγματικότητα υποθέτει ότι ο κώδικας είναι γραμμένος με κεφαλαία. Όμως σήμερα όλοι ανεξαιρέτως οι μεταγλωττιστές της FORTRAN 77 δέχονται και τα μικρά γράμματα.

### 3.3.2 Τύποι, Δηλώσεις

Κάθε μεταβλητή (variable) πρέπει να ορίζεται με μια δήλωση. Έτσι καθορίζεται και ο τύπος της μεταβλητής. Οι πιο συνήθεις δηλώσεις είναι της μορφής :

| | |
|---|---|
| integer | Λίστα μεταβλητών |
| real | Λίστα μεταβλητών |
| complex | Λίστα μεταβλητών |
| logical | Λίστα μεταβλητών |
| character | Λίστα μεταβλητών |

Η λίστα μεταβλητών περιέχει ονόματα μεταβλητών τα οποία χωρίζονται με κόμμα. Κάθε μεταβλητή πρέπει να δηλώνεται μονο μία φορά. Αν μία μεταβλητή δεν δηλωθεί τότε η FORTRAN 77 χρησιμοποιεί αυτόματα έναν εσωτερικό κανόνα καθορισμού του τύπου της μεταβλητής. Ο κανόνας αυτός είναι ο εξής : οι μεταβλητές που ξεκινούν από με τα γράμματα i,j,k,l,m,n θεωρούνται ως ακέραιες (integer) μεταβλητές ενώ όλες οι υπόλοιπες θεωρούνται πράγματικές (real) μεταβλητές.

### 3.3.3 Ακέραιες μεταβλητές, Μεταβλητές κινητής υποδιαστολής

Η FORTRAN 77 έχει έναν μόνο τύπο *ακεραίων μεταβλητών (integer, integer*4)*. Οι ακέραιοι συνήθως αποθηκεύονται σαν 32 bits (4 bytes) μεταβλητές. Αυτό έχει σαν αποτέλεσμα όλες οι ακέραιες μεταβλητές να παίρνουν τιμές στο διάστημα $[-M, M]$ όπου M είναι περίπου $2 \cdot 10^9$.

Οι τύποι μεταβλητών κινούμενης υποδιαστολής είναι δύο : οι *απλές πραγματικές (real ή real*4)* και οι *πραγματικές μεταβλητές διπλής ακρίβειας(double precision ή real*8)*. Για συνήθεις εφαρμογές η απλή ακρίβεια (real) είναι αρκετή όμως υπάρχουν και εφαρμογές που απαιτείται υψηλή ακρίβεια οπότε πρέπει να χρησιμοποιηθεί διπλή ακρίβεια (real*8). Συνήθως οι απλές πραγματικές μεταβλητές είναι 4 bytes ενώ οι διπλής ακρίβειας είναι 8 bytes.



Εδώ θα πρέπει να σημειώσουμε ότι η αυξανόμενη απαίτηση των εφαρμογών για περισσότερη ακρίβεια στους υπολογισμούς οδήγησε στην δημιουργία μεταγλωττιστών με την δυνατότητα να δέχονται μεγαλύτερες μεταβλητές, τόσο ακέραιες όσο και κινητής υποδιαστολής. Έτσι σήμερα όλοι ανεξαιρέτως οι μεταγλωττιστές μπορούν να διαχείριζονται μεταβλητές μέχρι τετραπλής ακρίβειας. Έτσι οι παρακάτω δηλώσεις είναι πλέον εφικτές :

| | |
|---|---|
| integer*8 | Λίστα μεταβλητών (Διπλή ακρίβεια) |
| real*16 | Λίστα μεταβλητών (Τετραπλή ακρίβεια) |

### 3.3.4 Η εντολή *parameter*

Πολλές σταθερές εμφανίζονται πολύ συχνά μέσα στο πρόγραμμα. Είναι λοιπόν επιθυμητό να τις ορίσουμε μια φορά μόνο στην αρχή του προγράμματος και να τις χρησιμοποιούμε μετά όταν μας χρειάζονται. Αυτή την δουλειά την κάνει η εντολή *parameter*. Για παράδειγμα το πρόγραμμα που είδαμε παραπάνω θα μπορούσε να είχε γραφτεί ως εξής :

```
c234567
      program circle
      real r,area,pi
      parameter(pi=3.14159)
c
c Read the real number r
c
      read*,r
c
c Compute the area of the circle
c
      area=pi*r*r
c
c Print the result
c
      print*, area
c
c End of the program
c
      stop
      end
```

Το *συντακτικό* της εντολής είναι :

**parameter** (όνομα=σταθερά,...,όνομα=σταθερά)

Οι κανόνες για την χρήση της εντολής *parameter* είναι οι εξής :

α) Η «μεταβλητή» που ορίζεται στη εντολή *parameter* δεν είναι πλέον μεταβλητή αλλά μια σταθερά της οποίας η τιμή δεν μπορεί να αλλάξει.

β) Μια «μεταβλητή» μπορεί να εμφανίζεται μόνο σε μία εντολή *parameter*.

γ) Η εντολή *parameter* μπαίνει πριν από την πρώτη εκτελέσιμη εντολή του προγράμματος.

Οι βασικοί λόγοι που χρησιμοποιούμε την εντολή *parameter* είναι ότι πολύ εύκολα μπορούμε να αλλάξουμε την τιμή μιας σταθεράς που εμφανίζεται πολλές φορές μέσα σε ένα κώδικα, αποφεύγουμε τυχόν τυπογραφικά λάθη και ο κώδικας είναι πιό εύκολο να διαβαστεί.

## 3.4 Εκφράσεις, Αναθέσεις

Η απλούστερη μορφή μιας έκφρασης είναι η *σταθερά*. Υπάρχουν 5 διαφορετικοί τύποι σταθερών που αντιστοιχούν στους 5 διαφορετικούς τύπους μεταβλητών που υπάρχουν.

**Ακέραιες Σταθερές:** 1, 0, -100, 32767, +15.

**Πραγματικές Σταθερές:** 1.0, -0.25, 2.0E6, 3.3333E-1. Ο συμβολισμός «E» σημαίνει ότι πρέπει να πολλαπλασιάσουμε την σταθερά με 10 στη δύναμη τον αριθμό που ακολουθεί το «E». Έτσι το 2.0E6 σημαίνει $2.0 \cdot 10^6$. Για τις σταθερές που είναι μεγαλύτερες από τον μεγαλύτερο επιτρεπτό πραγματικό αριθμό θα πρέπει να χρησιμοποιηθεί διπλή ακρίβεια. Ο συμβολισμός είναι ο ίδιος με τις σταθερές απλής ακρίβειας με την μόνη διαφορά ότι αντι για «E» υπάρχει «D», π.χ. 2.0D-1, 1.0D99.

**Μιγαδικές Σταθερές:** αναπαραστούνται με ένα ζευγάρι σταθερών(ακεραίων ή πραγματικών) που χωρίζονται με ένα κόμμα και περικλείονται από παρανθέσεις, π.χ. (2,-3)=2-3i, (1.5,9.3E-1)=1.5+0.93i. Ο πρώτος αριθμός αντιστοιχεί στο πραγματικό μέρος και ο δεύτερος στο φανταστικό μέρος του μιγαδικού αριθμού.

**Λογικές Σταθερές:** Οι λογικές σταθερές παίρνουν μόνο δύο τιμές :

**.TRUE.** (Αληθές)
**.FALSE.** (Ψευδές)

*ΠΡΟΣΟΧΗ:* Οι τελείες που περικλείουν τις τιμές είναι απαραίτητες.

**Σταθερές τύπου Χαρακτήρα.** Συνήθως οι σταθερές αυτές χρησιμοποιούνται σαν διάνυσμα χαρακτήρων και καλούνται *ορμαθοί χαρακτήρων(string)*. Αποτελούνται από αυθαίρετο αριθμό χαρακτήρων που εσωκλείονται σε αποστρόφους, π.χ. 'Καλημέρα'

Οι ορμαθοί χαρακτήρων και σταθερές τύπου χαρακτήρα είναι ευαίσθητες στα μικρά και κεφαλαία γράμματα

### 3.4.1 Εκφράσεις

Η απλούστερη έκφραση έχει την μορφή :

*Όρισμα Τελεστής Όρισμα* π.χ. *x+y*.

Το αποτέλεσμα μιας έκφρασης είναι έκφραση και μπορεί να χρησιμοποιηθεί εκ νέου σε μια καινούργια έκφραση:

x+2*y.

Η σειρά των πράξεων στη FORTRAN 77 γίνεται με την ακόλουθη σειρά :

** Ύψωση σε δύναμη.

*,/ Πολλαπλασιασμός, Διαίρεση.

+,- Πρόσθεση, Αφαίρεση.

Όλες αυτές οι πράξεις γίνονται απο αριστερά-προς-δεξιά εκτός από την ύψωση σε δύναμη που γίνεται από δεξιά-προς-αριστερά. Σε περίπτωση που θέλουμε να αλλάξουμε την σειρά των πράξεων χρησιμοποιούμε παρενθέσεις:

x+2*(a-y**5).

Στη περίπτωση αυτή οι πράξεις που βρίσκονατι μέσα στις παρενθέσεις θα εκτελεσθούν πρώτα με την συνήθη σειρά και μετα θα εκτελεστούν οι υπόλοιπες πράξεις. Όλοι οι παραπάνω τελεστές είναι δυαδικοί, δηλαδή απαιτούν δύο ορίσματα. Υπάρχει και ο τελεστής της *άρνησης:* - ο οποίος απαιτεί ένα μόνο όρισμα, π.χ.

-x+2*a.

Θα πρέπει να προσέξουμε ιδιαίτερα τη χρήση του τελεστή διαίρεση / ο οποίος δίνει τελείως διαφορετικά αποτελέσματα όταν χρησιμοποιείται με ακεραίους και πραγματικούς αριθμούς, π.χ.

5/2 Θα δώσει σαν αποτέλεσμα 2 και όχι το αναμενόμενο 2.5 .
5.0/2.0 Θα δώσει σαν αποτέλεσμα 2.5 .



### 3.4.2 Ανάθεση

Η ανάθεση έχει την μορφή :

όνομα-μεταβλητής=έκφραση

το οποίο ερμηνεύεται ως εξής : πρώτα αποτιμάτε το δεξί μέρος και η τιμή αυτή αναθέτεται στη μεταβλητή στο αριστερό μέρος. Η έκφραση στα δεξιά μπορεί να περιέχει άλλες μεταβλητές οι οποίες όμως δεν αλλάζουν τιμή κατα την ανάθεση, π.χ.

area=pi*r**2.

Η ανάθεση αυτή αλλάζει το περιεχόμενο της μεταβλητής area αλλά δεν αλλάζει τις τιμές των pi και r.

### 3.4.3 Αλλαγή Τύπου

Όταν σε μια έκφραση εμφανίζονται διαφορετικού τύπου μεταβλητές τότε γίνεται αυτόματα «αλλαγή τύπου», π.χ

```
real x
x=x+1
```

Η FORTRAN 77 θα μετατρέψει αυτόματα τον ακέραιο αριθμό 1 σε πραγματικό 1.0 και θα εκτελέσει την πράξη. Όμως σε πιο πολύπλοκες εκφράσεις είναι καλό να κάνει ο ίδιος ο προγραμματιστής τις απαραίτητες αλλαγές τύπων των μεταβλητών. Για το σκοπό αυτό η FORTRAN 77 μας περέχει από μια σειρά εσωτερικών συναρτήσεων που μας βοηθούν στη μετατροπή αυτή. Οι συναρτήσεις αυτές είναι :

| | |
|---|---|
| **int** | Μετατρέπει ένα πραγματικό αριθμό σε ακέραιο. |
| **real, float** | Μετατρέπει ένα ακέραιο αριθμό σε πραγματικό. |
| **dble** | Μετατρέπει ένα πραγματικό αριθμό σε πραγματικό αριθμό διπλής ακρίβειας. |
| **ichar** | Μετατρέπει ένα χαρακτήρα σε ακέραιο. |
| **char** | Μετατρέπει ένα ακέραιο σε χαρακτήρα. |

π.χ.

```
real x,y
real*8 w
w=dble(x)*dble(y)
```

*Σημείωση.* Για περισσότερες πληροφορίες σχετικά με αυτές τις συναρτήσεις θα πρέπει να ανατρέξει κανείς στο σχετικό βιβλίο οδηγιών του μεταγλωττιστή που χρησιμοποιεί.

## 3.5 Λογικές Εκφράσεις

Οι λογικές εκφράσεις παίρνουν μόνο δύο τιμές : .TRUE. ή .FALSE.. Μια λογική έκφραση μπορεί να παραχθεί συγκρίνοντας αριθμητικές εκφράσεις χρησιμοποιώντας τους παρακάτω *τελεστές σχέσης* :

| | |
|---|---|
| **.LT.** (less than) | $<$ |
| **.LE.** (less equal) | $\leq$ |
| **.GT.** (greater than) | $>$ |
| **.GE.** (greater equal) | $\geq$ |
| **.EQ.** (equal) | $=$ |
| **.NE.** (not equal) | $\neq$ |

Βλέπουμε λοιπόν ότι δεν μπορούμε να χρησιμοποιούμε στη FORTRAN 77 σύμβολα σαν $<$ ή $=$ για να κάνουμε συγκρίσεις αλλά θα πρέπει να χρησιμοποιούμε τις παραπάνω συντμήσεις των 2 γραμμάτων περικλειόμενες από τις τελείες!!! Οι λογικές εκφράσεις μπορούν να συνδυαστούν με τους παρακάτω *λογικούς τελεστές* :

| | |
|---|---|
| **.AND.** | Λογικό «ΚΑΙ». |
| **.OR.** | Λογικό «Η». |
| **.NOT.** | Λογικό «ΟΧΙ». |

### 3.5.1 Λογικές Μεταβλητές και Αναθέσεις

Οι λογικές τιμές αποθηκεύονται σε λογικές μεταβλητές. Η ανάθεση είναι ανάλογη αυτής της αριθμητικής ανάθεσης, π.χ.

```
logical a,b
a=.TRUE.
b=a.AND.3.LT.5/2
```

Η σειρά εκτέλεσης των «πράξεων» είναι σημαντική όπως βλέπουμε από την δεύτερη ανάθεση. Ο κανόνας είναι ότι οι αριθμητικές εκφράσεις αποτιμούνται πρώτα, μετά οι τελεστές σχέσης και τέλος οι λογικοί τελεστές. Έτσι η τιμή της μεταβλητής b στο παραπάνω παράδειγμα θα είναι .FALSE..

## 3.6 Η εντολή IF

Ένα σημαντικό κομμάτι μιας γλώσσας προγραμματισμού είναι οι *εντολές υπο συνθήκη*. Η πιό απλή τέτοια εντολή στη FORTRAN είναι η εντολή **IF**, η οποία έχει πολλές μορφές. Η πιό απλή από αυτές είναι η εξής :

if (Λογική Έκφραση) Εκτελέσιμη Εντολή

Η παραπάνω εντολή πρέπει να βρίσκεται στη ίδια γραμμή. Το επόμενο παράδειγμα υπολογίζει την απόλυτη τιμή του x

if (x.LT.0.0) x=-x

Αν χρειάζονται να εκτελεστούν παραπάνω από μία εντολές μέσα στο if τότε το συντακτικό της εντολής είναι το ακόλουθο,

if (Λογική Έκφραση) then
Εκτελέσιμες Εντολές
endif

Η πιο γενική μορφή της εντολής είναι :

if (Λογική Έκφραση) then
Εκτελέσιμες Εντολές
else
if (Λογική Έκφραση) then
Εκτελέσιμες Εντολές
:
:
else
Εκτελέσιμες Εντολές
endif

Η ροή της εκτέλεσης είναι από πάνω προς τα κάτω. Οι εκφράσεις υπό συνθήκη αποτιμούνται στη σειρά μέχρι κάποια από αυτές να είναι αληθής. Τότε εκτελούνται ο σχετικές εντολές και η εκτέλεση συνεχίζεται με την εντολή που βρίσκεται μετά το endif.

### 3.6.1 Φωλιασμένες εντολές IF

Εντολές με if μπορούν να υπάρχουν η μία μέσα στη άλλη χωρίς κανένα πρόβλημα. Για να μπορούμε να διαβάζουμε πιο εύκολα τον κώδικα είναι καλό να χρησιμοποιούμε τα απαραίτητα κενά στη αρχή των εντολών, όπως στο παρακάτω παράδειγμα



```
      if (x.gt.0.0) then
        if (x.ge.y) then
          print*, ' x is positive and x >= y'
        else
          print*, ' x is positive but x < y'
        endif
      else
        if (x.lt.0.0) then
          print*, ' x is negative '
        else
          print*, ' x is 0 '
        endif
      endif
```

Γενικά θα πρέπει να αποφεύγουμε να χρησιμοποιούμε φωλιασμένα if σε πολλά επίπεδα γιατί τα πράγματα γίνονται πολύ γρήγορα δύσκολα να τα παρακολουθήσουμε.

## 3.7 Ανακυκλώσεις, (LOOPS)

Για την επαναλαμβανόμενη εκτέλεση παρόμοιων προγραμμάτων χρησιμοποιούμε ***ανακυκλώσεις***. Αν γνωρίζεται κάποια άλλη γλώσσα προγραμματισμού μπορεί να έχετε δεί τις ανακυκλώσεις, *for* , *while* , *until.* Η FORTRAN 77 έχει μόνο την ανακύκλωση *do.* Η ανακύκλωση αυτή αντιστοιχεί στην ανακύκλωση for στις άλλες γλώσσες προγραμματισμού. Οι άλλες δομές ανακύκλωσης μπορούν να επιτευχθούν με την χρήση των εντολών if και goto.

### 3.7.1 Ανακυκλώσεις do

Η ανακύκλωση do χρησιμοποιείται για απλή καταμέτρηση. Το παρακάτω παράδειγμα υπολογίζει το άθροισμα των ακεραίων από το 1 μέχρι το n, όπου n είναι μιά ακέραια μεταβλητή που έχει πάρει κάποια τιμή.

```
      integer i, n, sum
c
      sum=0
      do 10 i=1,n
        sum=sum+i
 10   continue
      print*, ' sum= ', sum
```

Ο αριθμός 10 είναι μια ετικέττα εντολής. Συνήθως σε ένα πρόγραμμα θα υπάρχουν πολλές ανακυκλώσεις και εντολές που θα χρειάζονται κάποια ετικέττα. Είναι ευθύνη του προγραμματιστή να αναθέσει σε κάθε τέτοια εντολή μία και μοναδική ετικέττα. Η αριθμητική τιμή της κάθε ετικέττας δεν έχει απολύτως καμμία σημασία. Υπενθυμίζουμε ότι οι στήλες 2-5 είναι αποκλειστικά και μόνο για την χρήση των ετικεττών. Η μεταβλητή ανακύκλωσης που χρησιμοποιείται από την do αυξάνεται αυτόματα κατά 1 στη διαρκεία της εκτέλεσης. Το παρακάτω κομμάτι προγράμματος τύπωνει όλους τους άρτιους αριθμούς ανάμεσα στο 1 και 10 με φθίνουσα σειρά:

```
      integer i
      do 20 i=10,1,-2
        print*, 'i=',i
 20   continue
```

Η γενική μορφή μιας ανακύκλωσης do είναι η παρακάτω :

```
      do label var=expr1, expr2, expr3
        statments
label continue
```

όπου label είναι η ετικέττα, var είναι η μεταβλητή (δείκτης) της ανακύκλωσης και η οποία πρέπει να είναι ακέραιος, expr1 είναι η αρχική τιμή της var, expr2 η τελική τιμή της var και expr3 είναι η μονάδα μεταβολής(βήμα) της var.
*Παρατηρήσεις*

1. Η τιμή του δείκτη ανακύκλωσης δεν πρέπει να αλλάζει από άλλες εντολές μέσα στη ίδια την ανακύκλωση.
2. Πολλοί μεταγλωττιστές FORTRAN 77 επιτρέπουν το κλείσιμο της ανακύκλωσης με την εντολή enddo. Το πλεονέκτημα σε αυτή την περίπτωση είναι ότι δεν χρειαζόμαστε πλέον ετικέττα για να κλείσουμε την ανακύκλωση, π.χ

```
      integer i
      do i=10,1,-2
        print*, 'i=',i
      enddo
```

### 3.7.2 Ανακυκλώσεις *While*

Οι ανακυκλώσεις αυτού του τύπου δεν υποστηρίζονται απο την FORTRAN 77 αλλά σχεδόν όλοι οι μεταγλωττιστές περιέχουν την δομή αυτή. Αξίζει λοιπόν να δούμε την ανακύκλωση αυτή, το συναχτικό της οποίας είναι,

```
      while (logical expression) do
        statments
      enddo
```

ή εναλλακτικά

```
      do while (logical expression)
        statments
      enddo
```

Οι εντολές της ανακύκλωσης θα εκτελούνται εφ΄ οσον η συνθήκη παραμένει αληθής. Στη περίπτωση που ο μεταγλωττιστής δεν υποστηρίζει την ανακύκλωση while τότε μπορούμε να γράψουμε έναν ισοδύναμο κώδικα κάνοντας χρήση των εντολών if και goto. Το γενικό συντακτικό σε αυτή την περίπτωση είναι :

```
label if (logical expression) then
        statments
        goto label
      endif
```

Το παρακάτω παράδειγμα υπολογίζει και τύπωνει όλες τις δυνάμεις του 2 που είναι μικρότερες ή ίσες του 100. Η πρώτη υλοποίηση κάνει χρήση της ανακύκλωσης while ενώ η δεύτερη είναι με την χρήση των εντολών if και goto.

```
c Use of while loop
c
      integer n
c
      n=1
      do while(n.le.100)
        n=n*2
        print*, n
      enddo
```



```
c Use of if and goto
c
      integer n
c
      n=1
 10   if (n.le.100) then
        n=n*2
        print*, n
        goto 10
      endif
```

## 3.8 Διανύσματα, Πίνακες

Για την επίλυση πολλών προβλημάτων χρειάζονται τόσο τα διανύσματα όσο και οι πίνακες. Η δομή της FORTRAN που χειρίζεται αυτά είναι η *array*. Ένα μονοδιάστατο array είναι ένα διάνυσμα και ένα διδιάστατο array είναι ένας πίνακας. Για να κατανοήσουμε πλήρως την δομή αυτή δεν πρέπει να γνωρίζουμε μόνο το συντακτικό της αλλά και τον τρόπο με τον οποίο η FORTRAN 77 αποθηκεύει στην μνήμη τους πίνακες και τα διανύσματα.

### 3.8.1 Διανύσματα

Το *διάνυσμα* είναι η απλούστερη δομή και δεν είναι τίποτε άλλο παρά μιά ακολουθία από στοιχεία τα οποία είναι αποθηκευμένα διαδοχικά στη μνήμη. Για παράδειγμα η δήλωση

real vec(20)

δηλώνει ότι η μεταβλητή vec είναι ένα διάνυσμα μήκους 20 από πραγματικές μεταβλητές, με άλλα λόγια το vec αποτελείται από 20 πραγματικούς αριθμούς αποθηκευμένους διαδοχικά στη μνήμη. Κάθε στοιχείο του πίνακα είναι μια ξεχωριστή μεταβλητή. Αναφερόμαστε στο στοιχείο i του διανύσματος vec ως vec(i). Τα διανύσματα στη FORTRAN αριθμούνται από 1 και πάνω. Άρα ο πρώτος αριθμός στο διάνυσμα είναι ο vec(1) και ο τελευταίος ο vec(20). Υπάρχει η δυνατότητα να ορίσουμε την δική μας αρίθμηση σε ένα διάνυσμα :

real b(0:25), weird(-23:121)

Στο παράδειγμα αυτό το b είναι ένα διάνυσμα με 26 στοιχεία και η αρίθμηση του ξεκινά από το 0 και καταλήγει στο 25, ενώ το διάνυσμα weird έχει μήκος 145. Ο τύπος ενός διανύσματος μπορεί να είναι ένας από τους βασικούς τύπους δεδομένων που υποστηρίζει η FORTRAN 77, π.χ

integer ka(10)
logical aa(0:1)
real*8 x(100)

Το παρακάτω παράδειγμα αποθηκεύει τα τετράγωνα των 10 πρώτων αριθμών στο διάνυσμα sq.

```
      integer i, sq(10)
c
      do 100 i=1,10
        sq(i)=i*i
 100  continue
```

Ένα συχνό λάθος στη FORTRAN είναι κάποιο κομμάτι του κώδικα να προσπαθεί να προσπελάσει στοιχεία του διανύσματος που βρίσκονατι έξω από τα όρια του διανύσματος ή δεν έχουν οριστεί. Είναι ευθύνη του προγραμματιστή να αποφύγει τέτοια λάθη μιας και ο μεταγλωττιστής δεν είναι σε θέση να τα αναγνωρίσει.

### 3.8.2 Πίνακες

Οι *πίνακες* είναι πολύ σημαντικοί στη Γραμμική Άλγεβρα και πολύ συχνά χρησιμοποιούνται στη πράξη για την υλοποίηση διαφόρων δομών δεδομένων. Οι πίνακες αναπαραστούνται σαν διδιάστατα διανύσματα. Για παράδειγμα η δήλωση,

real aa(3,5)

δηλώνει ένα διδιάστατο πίνακα με 3*5=15 στοιχεία. Ο πίνακας αυτός έχει 3 γραμμές και 5 στήλες. Οι δείκτες των πινάκων μπορούν να έχουν αυθαίρετη αρίθμηση. Το γενικό συντακτικό είναι

τύπος όνομα(κάτω-άκρο1:πάνω-άκρο1, κάτω-άκρο2:πάνω-άκρο2)

με συνολικό αριθμό στοιχείων

Διάσταση=(πάνω-άκρο1 - κάτω-άκρο1 + 1)*(πάνω-άκρο2 - κάτω-άκρο2 + 1).

Συνήθως στη FORTRAN δηλώνουμε πίνακες με μεγαλύτερη διάσταση από αυτή που πραγματικά χρησιμοποιούμε, το οποίο είναι απολύτως νόμιμο, και ο βασικός λόγος είναι ότι η FORTRAN δεν έχει δυναμική παραχώρηση μνήμης, π.χ.

```
      real a(3,5)
      integer i, j
c
c Uses only the upper 3 x 3 part of the matrix
c
      do 20 j=1,3
        do 10 i=1,3
          a(i,j)=real(i)*float(j)
 10     continue
 20   continue
```

Τα στοιχεία του υποπίνακα a(1:3,4:5) δεν έχουν οριστεί. Δεν πρέπει να υποθέσουμε ότι στα στοιχεία αυτά ο μεταγλωττιστής έχει βάλει την τιμή 0, παρόλο ότι πολλοί μεταγλωττιστές το κάνουν αυτό, αλλά όχι όλοι.

### 3.8.3 Αποθήκευση πινάκων στη FORTRAN 77

Η FORTRAN αποθηκεύει τους πίνακες σαν μιά ακολουθία στοιχείων, ειδικότερα η αποθηκεύση γίνεται *κατά στήλες*. Έτσι στο παραπάνω παράδειγμα μετά το στοιχείο (3,1) ακολουθεί το (1,2) και το υπόλοιπο της 2ης στήλης, μετά η 3η στήλη κ.ο.χ.

Ας θεωρήσουμε ξανά το παραπάνω παράδειγμα όπου χρησιμοποιούμε μόνο το πάνω $3x3$ κομμάτι του πίνακα a(3,5). Τα 9 αυτά στοιχεία θα αποθηκευτούν σε σειρά στις 9 πρώτες θέσεις μνήμης, ενώ τα υπόλοιπα 6 δεν χρησιμοποιούνται καθόλου. Η αποθήκευση αυτή είναι ιδανική μιας και τα στοιχεία που χρειαζόμαστε είναι αποθηκευμένα διαδοχικά στη μνήμη. Ο βασικός λόγος είναι ότι η *πρωτεύουσα διάσταση*, δηλαδή ο αριθμός των γραμμών, του πίνακα που έχουμε δηλώσει είναι ίση με την πρωτεύουσα διάσταση του πίνακα που χρησιμοποιούμε. Αυτό όμως δεν συμβαίνει συχνά και συνήθως η πρωτεύουσα διάσταση του πίνακα είναι μεγαλύτερη από την πρωτεύουσα διάσταση του πίνακα που πραγματικά χρησιμοποιούμε, με αποτέλεσμα ο πίνακας να μην αποθηκεύεται σε συμπαγή μορφή. Για παράδειγμα αν είχαμε την δήλωση a(5,3), τότε θα υπήρχαν 2 αχρησιμοποίητες κενές θέσεις μνήμης ανάμεσα στο τέλος της μια στήλης και στη αρχή της επόμενης, υποθέτοντας ότι χρησιμοποιούμε πάλι μόνο το πάνε $3x3$ κομμάτι του πίνακα.

### 3.8.4 Πίνακες Πολλαπλών Διαστάσεων

Η FORTRAN 77 μας επιτρέπει να δηλώσουμε πίνακες μέχρι 7 διαστάσεων. Το συνακτικό και η αποθήκευση των πινάκων αυτών είναι ανάλογη με αυτή των διδιάστατων πινάκων που είδαμε παραπάνω.



### 3.8.5 Η εντολή dimension

Η εντολή αυτή μας δίνει έναν εναλλακτικό τρόπο για να δηλώσουμε διανύσματα και πίνακες στη FORTRAN Οι εντολές

real a, x
dimension a(10,20), x(100)

είναι ισοδύναμες με

real a(10,20), x(100)

## 3.9 Υποπρογράμματα

Όταν ένας κώδικας είναι αρκετά μεγάλος γίνεται δύσκολο να τον παρακολούθησουμε. Οι κώδικες της FORTRAN που λύνουν αριθμητικά, μεγάλα προβλήματα είναι αρκετές χιλιάδες γραμμές. Ο μόνος τρόπος για την διαχείρηση τέτοιων μεγάλων προγραμμάτων είναι να το σπάσουμε σε πολλά μικρότερα κομμάτια τα οποία λέγονται *υποπρογράμματα*.

Ένα υποπρόγραμμα είναι ένα μικρό πρόγραμμα το οποίο λύνει ένα πλήρως ορισμένο υποπρόβλημα του κυρίως προβλήματος. Από την άλλη μεριά σε ένα μεγάλο πρόγραμμα χρειάζεται αρκετά συχνά να λύσουμε το ίδιο ακριβώς πρόβλημα αλλά με διαφορετικά δεδομένα. Έτσι αντί να ξαναγράφουμε το ίδιο ακριβώς κώδικα, απλώς καλούμε το υποπρόγραμμα με διαφορετικά δεδομένα εισόδου. Η FORTRAN έχει δύο διαφορετικά είδη υποπρογραμμάτων : *Συναρτήσεις* και *Υπορουτίνες*.

### 3.9.1 Συναρτήσεις(FUNCTIONS)

Οι συναρτήσεις της FORTRAN είναι αρκετά όμοιες με τις μαθηματικές συναρτήσεις, και οι δύο παίρνουν ένα σύνολο παραμέτρων σαν είσοδο και επιστρέφουν μια τιμή κάποιου συγκεκριμμένου τύπου. Στη FORTRAN υπαρχούν οι συναρτήσεις που ορίζονται από τον χρήστη και οι έτοιμες συναρτήσεις οι οποίες μπορούν να χρησιμοποιηθούν ανά πάσα στιγμή, π.χ.

x=cos(pi/3.0) .

Στο παράδειγμα αυτό η *cos* είναι η έτοιμη συνάρτηση που υπολογίζει το συνιμήτονο του ορίσματος. Στη προκειμένη περίπτωση το x θα πάρει την τιμή 0.5 εφ΄ όσον έχουμε δώσει στο pi=π την σωστή τιμή. Εδώ θα πρέπει να σημειώσουμε ότι η FORTRAN 77 δεν έχει έτοιμες σταθερές. Υπάρχουν πολλές έτοιμες συναρτήσεις, για τον πλήρη κατάλογο θα πρέπει να συμβουλευτεί κανείς τον οδηγό χρήσης του μεταγλωττιστή που χρησιμοποιεί. Παρακάτω αναφέρουμε μερικές από τις πιο βασικές έτοιμες συναρτήσεις :

**abs** Απόλυτη τιμή.

**min** Μικρότερη τιμή.

**max** Μεγαλύτερη τιμή.

**sqrt** Τετραγωνική ρίζα.

**sin** Ημίτονο.

**cos** Συνημίτονο.

**tan** Εφαπτομένη

**asin** Τόξο ημιτόνου.

**acos** Τόξο συνημιτόνου.

**atan** Τόξο εφαπτομένης.

**exp** Εκθετικό (φυσικό).

**log** Λογάριθμος (φυσικός).

Οι παραπάνω συναρτήσεις είναι αυτό που λέμε *βασικές*. Κάθε μία από αυτές έχει το δικό της τύπο αποτελέσματος. Έτσι ο χρήστης θα πρέπει να είναι προσεκτικός ώστε τα ορίσματα στις συναρτήσεις αυτές να ανταποκρίνονται στο τύπο αποτελέσματος που παρέχει η κάθε συνάρτηση. Στο προηγούμενο παράδειγμα το x και pi μπορεί να είναι είτε απλής ακρίβειας είτε διπλής, οπότε θα πρέπει να χρησιμοποιούμε την κατάλληλη παραλλαγή της βασικής συνάρτησης έτσι ώστε να συμφωνεί ο τύπος αποτελέσματος με αυτό των ορισμάτων, π.χ.

```
real*8 pi, x
pi=3.14159d0
x=dcos(pi/3.0d0)
```

Βλέπουμε ότι στην περίπτωση που οι μεταβλητές είναι διπλής ακρίβειας θα πρέπει να χρησιμοποιήσουμε την παραλλαγή της cos την dcos για να πάρουμε το αποτέλεσμα σε διπλή ακρίβεια καθώς επίσης και στις σταθερές θα πρέπει να προστεθεί στο τέλος η ένδειξη d0.

Ας δούμε τώρα τις συναρτήσεις που μπορεί να ορίσει ο χρήστης. Ξεκινάμε με ένα παράδειγμα. Ένας μετεωρολόγος για μιά συγκεκριμμένη περιοχή έχει βρεί ένα μοντέλο B(m,t) το οποίο μετρά το ύψος της βροχόπτωσης. Έτσι το B είναι το ποσό της βροχής, m είναι ο μήνας και t είναι μιά παράμετρος που εξαρτάται από την θέση της περιοχής. Δοσμένου λοιπόν του t θα θέλαμε να υπολογίσουμε την ετήσια βροχόπτωση.

Ένας τρόπος για να λύσουμε το πρόβλημα θα ήταν να γράψουμε μια ανακύκλωση πάνω σε όλους τους μήνες και να αθροίσουμε τις τιμές της B. Αφού ο υπολογισμός της τιμής της B είναι ανεξάρτητο πρόβλημα είναι λογικό να την υλοποιήσουμε σαν συνάρτηση. Το κυρίως πρόγραμμα είναι :

```
c234567
      program Rain
      real b, t, sum
      integer m
c
c Read the parameter t
c
      read*, t
      sum=0.0
c
c Loop over all months
c
      do 10 m=1,12
        sum=sum+b(m,t)
 10   continue
c
c Print the total rainfall
c
      print*, ' The annual rainfall is :', sum
c
      stop
      end
```

Επιπλέον η συνάρτηση B πρέπει να οριστεί σαν μια συνάρτηση FORTRAN . Ο τύπος που βρήκε ο μετεωρολόγος είναι ο εξής :

$$B(m,t) = \begin{cases} \frac{t}{10}(m^2+14m+46) & t>0 \\ 0 & t \leq 0 \end{cases}$$

Η αντίστοιχη συνάρτηση στη FORTRAN είναι :



```
      real function b(m,t)
      integer m
      real t
c
      b=0.1*t*(m**2+14.0*m+46.0)
      if (t.le.0.0) b=0.0
c
      return
      end
```

Βλέπουμε ότι η δομή της συνάρτησης είναι παρόμοια με αυτή του κυρίως προγράμματος. Οι βασικές διαφορές είναι ,

α) Οι συναρτήσεις έχουν συγκεκριμμένο τύπο. Ο τύπος αυτός θα πρέπει να δηλωθεί και στο κυρίως πρόγραμμα.

β) Η τιμή που επιστρέφουν πρέπει να αποθηκευτεί σε μία μεταβλητή με το ίδιο ακριβώς όνομα με την συνάρτηση.

γ) Οι συναρτήσεις τελειώνουν με την εντολές return,end αντί stop, end.

Συνοψίζοντας το γενικό συντακτικό μιας συνάρτησης FORTRAN 77 είναι το ακόλουθο,

```
τύπος function όνομα(Λίστα - μεταβλητών)
δηλώσεις
εντολές
return
end
```

Η συνάρτηση πρέπει να δηλωθεί με τον σωστό τύπο στη μονάδα προγράμματος που καλείται. Η συνάρτηση απλά καλείται χρησιμοποιώντας το ονομά της και με τις παραμέτρους. Ο κώδικας που υλοποιεί την συνάρτηση τοποθετείται μετά το κυρίως πρόγραμμα.

### 3.9.2 Υπορουτίνες(SUBROUTINES)

Μια συνάρτηση στη FORTRAN όπως είδαμε μπορεί να επιστρέψει μία τιμή. Συχνά θα θέλαμε να επιστρέψουμε παραπάνς από μία τιμές, αλλά και μερικές φορές καμμία τιμή!!! Για το σκοπό αυτό χρησιμοποιούμε την δομή της *υπορουτίνας,(subroutine)*. Το συντακτικό είναι το ακόλουθο :

```
subroutine όνομα(λίστα-ορισμάτων)
δηλώσεις
εντολές
return
end
```

Παρατηρούμαι ότι οι υπορουτίνες δεν έχουν συγκεκριμμένο τύπο και συνεπώς δεν μπορούν και δεν δηλώνονται στη μονάδα προγράμματος που καλούνται. Δίνουμε παρακάτω ένα απλό παράδειγμα μιας υπορουτίνας που ανταλλάζει τις τιμές δύο ακεραίων μεταβλητών.

```
c234567
      subroutine iswap(a,b)
      integer a, b
c
c Local Variables
      integer temp
c
      temp=a
```

```
      a=b
      b=tmp
c
      return
      end
```

Παρατηρούμαι ότι υπάρχουν δύο ειδών μεταβλητές στις δηλώσεις της υπορουτίνας. Πρώτα δηλώνουμε τις *μεταβλητές εισαγωγής/εξαγωγής* δηλαδή τις μεταβλητές οι οποίες είναι κοινές στη υπορουτίνα και στο κομμάτι του προγράμματος που καλεί την υπορουτίνα. Έπειτα δηλώνονται οι *τοπικές μεταβλητές* δηλαδή οι μεταβλητές που χρησιμοποιούνται μόνο από την υπορουτίνα. Μπορούμε να χρησιμοποιούμε τα ίδια ονόματα μεταβλητών σε διάφορες υπορουτίνες και ο μεταγλωττιστής θα αναγνωρίσει ότι είναι διαφορετικές μεταβλητές που απλώς συμβαίνει να έχουν το ίδιο όνομα.
*Σημείωση.* Η FORTRAN 77 χρησιμοποιεί το λεγόμενο *κάλεσμα κατά αναφορά* όταν καλεί υπορουτίνες και όχι το *κάλεσμα κατά τιμή* πράγμα που σημαίνει ότι στη υπορουτίνα δεν περνούν απλώς οι τιμές των ορισμάτων αλλά οι διευθύνσεις της μνήμης των ορισμάτων αυτών. Τι ακριβώς εννοούμε γίνεται άμεσα κατανοητό με το παρακάτω παράδειγμα :

```
c234567
      program call
      integer m, n
c
      m=1
      n=2
      call iswap(m,n)
      print*, m, n
c
      stop
      end
```

Όπως είδαμε παραπάνω η υπορουτίνα *iswap* ανταλλάσει τις τιμές δύο ακεραίων, έτσι το αναμενόμενο αποτέλεσμα του παραπάνω κώδικα είναι : 2 1 . Αν η FORTRAN 77 έκανε απλώς *κάλεσμα κατά τιμή* τότε το αποτέλεσμα θα ήταν : 1 2 , δηλαδή οι τιμές των μεταβλητών θα παράμεναν αμετάβλητες. Ο λόγος είναι ότι με το *κάλεσμα κατά τιμή* οι τιμές των m,n απλώς θα αντιγραφόταν στη υπορουτίνα και οποιάδηποτε αλλαγή που επιφέρει η υπορουτίνα στις μεταβλητές δεν θα περνούσε πίσω στο κυρίως πρόγραμμα. Θα πρέπει να επισημάνουμε εδώ ότι χρειάζεται κάποια προσοχή έτσι ώστε να μην χρησιμοποιούμε κάποια από τις μεταβλητές εισόδου σαν τοπική μεταβλητή σε μια υπορουτίνα, πράγμα που θα είχει ανεπιθύμητα αποτελέσματα.

## 3.10 Εντολές Εισόδου/Εξόδου

Ένα σημαντικό κομμάτι ενός προγράμματος είναι αφιερωμένο στη επικοινωνία με το χρήστη και πιό συγκεκριμένα στη είσοδο δεδομένων και εξαγωγή αποτελεσμάτων. Στα παραδείγματα που έχουμε δεί μέχρι τώρα έχουμε χρησιμοποιήσει τις πιό συνήθεις εντολές της FORTRAN 77 τις :*read* και *print*.

### 3.10.1 Οι εντολές *READ, WRITE, PRINT*

Η εντολή *read* χρησιμοποιείται για την εισαγωγή δεδομένων ενώ οι εντολές *write, print* για την εμφάνιση δεδομένων. Η απλούστερη μορφή των εντολών αυτών είναι η εξής :

read(αρχείο, τρόπος) λίστα-μεταβλητών
write(αρχείο, τρόπος) λίστα-μεταβλητών

Τα *αρχείο* και *τρόπος* είναι φυσικοί αριθμοί και καθορίζουν ο μεν την μονάδα αρχείου από/στην οποία θα διαβαστούν/αποθηκευτούν οι μεταβλητές, και ο άλλος την μορφή με την οποία θα διαβαστούν/αποθηκευτούν οι μεταβλητές. Περισσότερες λεπτομέρειες δίνονται στη επόμενη παράγραφο.
Είναι δυνατό να απλοποιήσουμε ακόμα περισσότερο τις εντολές αυτές χρησιμοποιώντας αστερισκούς (*) τόσο για το *αρχείο* όσο και για τον *τρόπο*,



read(*,*) λίστα-μεταβλητών
write(*,*) λίστα-μεταβλητών

Η πρώτη εντολή απλά διαβάζει από το πληκτρολόγιο-οθόνη δεδομένα και τα καταχωρεί στις μεταβλητές της ***λίστα-μεταβλητών***, ενώ η δεύτερη εμφανίζει το περιεχόμενο των μεταβλητών της ***λίστα-μεταβλητών*** στη οθόνη. Η πιό απλή μορφή της εντολής read είναι

read*, λίστα-μεταβλητών

Επίσης μπορούμε να εμφανίζουμε αποτελέσματα στη οθόνη με την βοήθεια της εντολής print, όπως έχουμε ήδη κάνει μέχρι τώρα στα παραδείγματα,

print*, λίστα-μεταβλητών
print*, 'Μήνυμα'

Η εντολή αυτή απλά εμφανίζει στη οθόνη το περιεχόμενο των μεταβλητών ή εμφανίζει ότι υπάρχει ανάμεσα στις αποστρόφους, όπως μας δείχνει το δεύτερο παράδειγμα.

## 3.11 Η εντολή FORMAT

Μέχρι τώρα στα παραδείγματα που έχουμε δεί δεν χρησιμοποιήσαμε κανένα ιδιαίτερο τρόπο για να εκτύπωσουμε το περιεχόμενο των μεταβλητών ή να εισάγουμε δεδομένα με κάποιο συγκεκριμμένο τρόπο. Συχνά ο προγραμματιστής θα ήθελε να εμφανίσει/διαβάσει ο κώδικας κάποιες μεταβλητές με μια ιδιαίτερη μορφή. Για το σκοπό αυτό η FORTRAN 77 έχει την εντολή *format*. Η εντολή αυτή είναι η ίδια τόσο για την είσοδο όσο και την εμφάνιση δεδομένων. Το συντακτικό είναι :

        read(* , label1) λίστα-μεταβλητών
label1 format( κώδικας )

        write(* , label2) λίστα-μεταβλητών
label2 format( κώδικας )

Ένα απλό παράδειγμα : Έστω ότι θέλουμε να τύπωσουμε μια ακέραια μεταβλητή σε ένα πεδίο 4 θέσεων και μια πραγματική μεταβλητή με την μορφή σταθερής υποδιαστολής σε πεδίο 8 θέσεων με 3 δεκαδικά ψηφία.

    write(* , 100) k, a
100 format(i4 , f8.3)

Η επιλογή του συγκεκριμμένου αριθμού ετικέττας (label) είναι αυθαίρετη. Μετα την λέξη format ακολουθεί ο τρόπος μορφοποίησης των μεταβλητών περικλειόμενος από παρενθέσεις. Το i4 δηλώνει ότι θα τυπωθεί ένας ακέραιος πλάτους 4 θέσεων, ενώ το f8.3 σημαίνει ένα πραγματικό αριθμό πλάτους 8 θέσεων εκ των οποίων οι 3 είναι για τα δεκαδικά ψηφία. Η εντολή format μπορεί να βρίσκεται οπουδήποτε στο πρόγραμμα. Οι συνήθεις τακτικές είναι :

α) είτε η εντολή format βρίσκεται αμέσως μετά από το αντίστοιχο read/write,

β) είτε όλες οι εντολές format βρίσκονται μαζί στη αρχή ή τέλος του προγράμματος.

### 3.11.1 Τρόποι Μορφοποίησης

Οι πιό συνήθεις τρόποι μορφοποίησης είναι :

**A** Για ορμαθούς χαρακτήρων.

**D** Για πραγματικούς αριθμούς διπλής ακρίβειας, εκθετική αναπαράσταση.

**E** Για πραγματικούς αριθμούς, εκθετική αναπαράσταση.

**F** Για πραγματικούς αριθμούς, μορφή σταθερής υποδιαστολής.

**I** Για ακεραίους.

**X** Ένα κενό.

/ Αλλαγή γραμμής.

Ο κωδικός F όπως και οι D,E, έχουν τη γενική μορφή : Fw.d όπου w είναι ένας ακέραιος αριθμός που δηλώνει το πλάτος του αριθμού και d είναι ένας ακέραιος που δηλώνει τον αριθμό των σημαντικών ψηφίων που θα χρησιμοποιηθούν. Για τους ακεραίους δηλώνουμε μόνο το πλάτος του αριθμού, οπότε το συντακτικό είναι, Iw. Όμοια ορμαθοί χαρακτήρων προσδιορίζονται από Aw αλλά συνήθως το πλάτος του πεδίου παραλείπεται. Αν ένας αριθμός ή ορμαθός χαρακτήρων δεν μπορεί να γεμίσει ολόκληρο το πεδίο τότε προστίθενται κενά στα αριστερά του αριθμού αφού έχουμε δεξιά στοίχηση. Για να αφήσουμε κενά (οριζοντίως) χρησιμοποιούμε το nX όπου n είναι ο αριθμός των κενών που θέλουμε να βάλουμε. Αν παραλείψουμε το n τότε υποτίθεται ότι n=1. Επίσης αν θέλουμε να μεταφερθούμε σε νέα γραμμή δίνουμε : / . Κάθε / αντιστοιχεί και σε νέα γραμμή.

*Παραδείγματα* Οι παρακάτω εντολές

```
      x=0.025
      write(* , 100) 'x=',x}
 100  format(a,f)
      write(* , 110) 'x=',x
 110  format(a,f5.3)
      write(* , 200) 'x=',x
 200  format(a,e)
      write(* , 250) 'x=',x
 250  format(a,e8.1)
```

θα δώσουν τα εξής αποτελέσματα

```
      x=      0.025000
      x=0.025
      x=  0.2500000E-1
      x=   0.3E-1
```

Παρατηρήστε τα κενά που υπάρχουν μπροστά από το πρώτο παράδειγμα και ότι το στάνταρτ πλάτος πεδίου για τους πραγματικούς στη FORTRAN 77 είναι 14. Επίσης από την 4η περίπτωση του παραδείγματος βλέπουμε οτι η FORTRAN 77 ακολουθεί το κανόνα στρογγύλευσης όπου τα ψηφία 0-4 στρογγυλοποιούνται προς τα κάτω ενώ τα ψηφία 5-9 προς τα πάνω. Σε αυτό παράδειγμα κάθε εντολή write χρησιμοποιεί ξεχωριστή εντολή format. Από την άλλη μεριά είναι απολύτως σωστό πολλές εντολές write ή read να χρησιμοποιούν την ίδια εντολή format, αυτό είναι άλλωστε ένα από τα μεγαλύτερα πλεονεκτήματα της. Μια εναλλακτική χρήση της format είναι :

```
      x=0.025
      y=1.12575
      write(* , 999) x,y
 999  format('x=',f8.3,/,'y=',e12.6)
```

θα δώσει

```
      x=   0.025
      y=0.112575E+01
```



### 3.11.2 Έμμεσες Ανακυκλώσεις

Ας δούμε ένα λίγο πιό σύνθετο παράδειγμα. Ας υποθέσουμε ότι έχουμε ένα διδιαστατο πίνακα με ακεραίους και θέλουμε να τυπώσουμε το πάνε αριστερό υποπίνακα με διάσταση $5x10$ με 10 αριθμούς σε κάθε μία από τις 5 γραμμές. Η υλοποίηση μπορεί να γίνει ως εξής

```
      do 100 i=1,5
        write(* , 200) (a(i,j),j=1,10)
 100  continue
 200  format(i6)
```

Σε αυτό το παράδειγμα έχουμε μία άμμεση ανακύκλωση πάνω στις γραμμές του πίνακα και μια *έμμεση ανακύκλωση* πάνω στις στήλες. Συχνά μιά εντολή format μπορεί να περιλαμβάνει επαναλήψεις, για παράδειγμα,

900 format(2x,f6.2,2x,f6.2,2x,f6.2)

το οποίο μπορεί να συντομευτεί και να γραφτεί ως

900 format(3(2x,f6.2))

Είναι επίσης εφικτό να υπάρχουν επαναλήψεις χωρίς να δηλώσουμε άμμεσα πόσες φορές η συγκεκριμμένη μορφή θα επαναληφθεί. Ας υποθέσουμε ότι έχουμε ένα διάνυσμα και θέλουμε να τυπώσουμε τα 50 πρώτα στοιχεία του, με 10 στοιχεία ανά γραμμή. Ένας τρόπος είναι ο ακόλουθος,

```
      write(* , 1010) (x(i),i=1,50)
 1010 format(10f6.2)
```

Η εντολή format λέει ότι θα τυπωθούν 10 αριθμοί αλλά στη εντολή write προσπαθούμε να τυπώσουμε 50 αριθμούς. Έτσι μετά τους 10 πρώτους αριθμούς που θα τυπωθούν η ίδια εντολή format θα χρησιμοποιηθεί αυτόματα για να τυπώσει τους επόμενους 10 αριθμούς κ.ο.χ.

## 3.12 Είσοδος/Έξοδος Αρχείων

Μέχρι τώρα είχαμε υποθέσει ότι η είσοδος/εξόδος κατευθυνόταν από/προς τις συνήθεις εισόδους/εξόδους που δεν ήταν τίποτα άλλο από το πληκτρολόγιο και την οθόνη. Βέβαια στη FORTRAN 77 έχουμε την δυνατότητα να διαβάσουμε ή να γράψουμε δεδομένα σε αρχεία(files) τα οποία είναι αποθηκευμένα σε σκληρό ή μαλακό δίσκο ή ταινία. Στη FORTRAN κάθε αρχείο συνδέεται με κάποιο *αριθμό μονάδας* που δεν είναι τίποτε άλλο από ένα ακέραιο από 1-99. Κάποιοι αριθμοί μονάδων είναι δεσμευμένοι: το 5 από τη στάνταρτ είσοδο και το 6 από την στάνταρτ έξοδο.

### 3.12.1 Ανοίγοντας/κλείνοντας ένα αρχείο

Πριν μπορέσουμε να χρησιμοποιήσουμε ένα αρχείο θα πρέπει να το ανοίξουμε. Η εντολή είναι :

open(Λίστα - παραμέτρων)

όπου οι συνήθεις παράμετροι είναι :

**UNIT=** Αριθμός Μονάδος.

**FILE=** Όνομα Αρχείου.

**STATUS=** Κατάσταση Αρχείου.

Ο αριθμός UNIT είναι ένας ακέραιος αριθμός που καθορίζει τον αριθμό μονάδος του αρχείου. Η παράμετρος FILE είναι ένας ορμαθός χαρακτήρων που καθορίζει το όνομα του αρχείου και η STATUS δηλώνει την κατάσταση του αρχείου και παίρνει τις εξής τιμές : NEW όταν πρόκειται για καινούργιο αρχείο, OLD όταν το αρχείο ήδη υπάρχει, SCRATCH για ένα αρχείο το οποίο σβήνεται όταν το κλείσουμε.

*Σημείωση.* Υπάρχουν πολλές περισσότερες παράμετροι της εντολής open. Για περισσότερες πληροφορίες ο χρήστης θα πρέπει να συμβουλευτεί το εγχειρίδιο χρήσης του μεταγλωττιστή της FORTRAN 77 που χρησιμοποιεί.

Από την στιγμή που ένα αρχείο ανοιχτεί μπορεί να προσπελαθεί από εντολές read και write. Όταν τελειώσουμε θα πρέπει να κλείσουμε το αρχείο με την εντολή close

close(Αριθμός - Μονάδας)

π.χ.

```
      open(12,file='mesh.dat')
      write(12,10) x
      close(12)
 10   format(f8.2)
```

Στο παραπάνω παράδειγμα έχοντας ανοίξει το αρχείο με όνομα mesh.dat και με αριθμό μονάδος 12, γράφεται το περιεχόμενο της μεταβλητής x με την εντολή format 10 και έπειτα κλείνει το αρχείο με την εντολή close.

Στο παρακάτω παράδειγμα υποθέτουμε ότι υπάρχει ένα αρχείο με όνομα points.dat με τις συντεταγμένες σημείων x, y, z. Στη πρώτη γραμμή του αρχείου υπάρχει ο αριθμός σημείων. Η μορφή της κάθε συντεταγμένης είναι f10.4. Το παρακάτω πρόγραμμα διαβάζει τις συντεταγμένες και τις αποθηκεύει σε 3 διανύσματα.

```
c234567
      program Points
      integer nmax, u
      parameter(nmax=1000, u=20)
      real x(nmax), y(nmax), z(nmax)
c
c File unit number
      u=1
c
c Open the file
      open(u,file='points.dat',status='old')
c
c Read the number of elements in file
      read(u,*) n
      if (n.gt.nmax) then
        write(*,*) 'Error in n=',n,'is greater than nmax=', nmax
        goto 9999
      endif
c
c Loop over the elements
      do 10 i=1,n
        read(u,100) x(i), y(i), z(i)
 10   continue
 100  format(3(f10.4))
c
c Closing the file
      close(u)
c
 9999 stop
      end
```

## 3.13 Πίνακες σε υποπρογράμματα

Όπως είδαμε η κλήση των υποπρογραμμάτων στη FORTRAN γίνεται με κλήση κατά αναφορά, πράγμα που σημαίνει ότι οι παράμετροι κλήσης δεν αντιγράφονται στο υποπρόγραμμα αλλά οι διευθύνσεις των παραμέτρων



(μεταβλητές) περνούν στο υποπρόγραμμα. Το γεγονός αυτό εξικονομεί μνήμη ιδιαίτερα όταν έχουμε να κάνουμε με διανύσματα και πίνακες. Είναι δυνατό σε μια υπορουτίνα να δηλώσουμε διανύσματα και πίνακες σαν τοπικές μεταβλητές, πράγμα όμως που δεν συνίσταται. Όλα τα διανύσματα και οι πίνακες δηλώνονται στο κυρίως πρόγραμμα και μετά απλώς περνούν στα υποπρογράμματα.

### 3.13.1 Διανύσματα/Πίνακες Μεταβλητής Διάστασης

Η βασικότερη ίσως πράξη μεταξύ διανυσμάτων είναι :

y = a * x + y

όπου a είναι μια σταθερά και x, y είναι διανύσματα. Μια απλή υπορουτίνα που υλοποιεί την παραπάνω πράξη είναι :

```
      subroutine saxpy(n,a,x,y)
      integer n
      real a, x(*), y(*)
c
c Local variables
      integer i
c
      do 10 i=1,n
        y(i)=a*x(i)+y(i)
 10   continue
c
      return
      end
```

Το καινούριο στοιχείο εδώ είναι οι αστερίσκοι (*) στις δηλώσεις x(*), y(*). Ο συμβολισμός αυτός λέει ότι τα x,y είναι διανύσματα αυθαίρετου μήκους. Το πλεονέκτημα αυτής της δήλωσης είναι ότι μπορούμε να χρησιμοποιήσουμε την ίδια υπορουτίνα για οποιαδήποτε διανύσματα όση και αν είναι η διαστασή τους. Όπως είδαμε παραπάνω δεν χρειάζεται επιπλέον μνήμη για την αποθήκευση των διανυσμάτων αυτών, πέρα από αυτή που έχει δηλωθεί στο κυρίως πρόγραμμα. Θα μπορούσαμε επίσης να δηλώσουμε τα διανύσματα ως εξής :

real x(n), y(n)

Βέβαια στα περισσότερα υποπρογράμματα η δήλωση των διανυσμάτων γίνεται με τους αστερίσκους για να υποδηλώσουμε ότι η «πραγματική» τους διάσταση είναι άγνωστη. Επίσης σε παλιά προγράμματα FORTRAN υπάρχει και η ακόλουθη δήλωση :

real x(1), y(1)

το οποίο από συντακτικής άποψης είναι απολύτως σωστό εφ' όσον οι διαστάσεις των διανυσμάτων είναι μεγαλύτερες του 1.

### 3.13.2 Υποπίνακες σε υπορουτίνες

Ας υποθέσουμε ότι θέλουμε να γράψουμε μιά υπορουτίνα που να εκτελεί το πολλαπλασιασμό πίνακα επί διάνυσμα. Αναλυτικότερα θα θέλαμε να υπολογίσουμε το διάνυσμα $y \in R^m$, όπου y=Ax, $A \in R^{m,n}, x \in R^n$ Υπάρχουν δύο τρόποι για γίνει αυτό. Ένας από αυτούς είναι να χρησιμοποιήσουμε την υπορουτίνα saxpy που είδαμε στην προηγούμενη παράγραφο. Ο κώδικας είναι :

```
      subroutine matvec(m,n,a,lda,x,y)
      integer m,n,lda
      real x(*), y(*), a(lda,*)
c
c Local variables
```

```
      integer i, j
c
c Initialize y
      do 10 i=1,m
        y(i)=0.0
 10   continue
c
c Matrix-vector multiplication using saxpy over the
c columns of A with lenght m
      do 20 j=1,n
        call saxpy(m,x(j),a(1,j),y)
 20   continue
c
      return
      end
```

Υπάρχουν μερικά πράγματα που πρέπει να παρατηρήσουμε εδώ. Πρώτον, παρόλο που ο κώδικας είναι αρκετά γενικός και μπορεί να δουλέψει για οποιονδήποτε m και n πίνακα, πρέπει να δηλώσουμε την πρωτεύουσα διάσταση του πίνακα. Η χρήση του αστερίσκου μπορεί να γίνει μόνο για την τελευταία διάσταση του πίνακα και ο λόγος είναι ο τρόπος που η FORTRAN αποθηκεύει τους πίνακες.

Όταν υπολογίζουμε το $y = A*x$ με πράξεις της μορφής saxpy πρέπει να αποκτήσουμε πρόσβαση στις στήλες του πίνακα που είναι της μορφής A(1:m,j). Όμως στη FORTRAN 77 δεν μπορούμε να χρησιμοποιήσουμε τέτοια δομή. Έτσι αντί αυτού παρέχουμε ένα δείκτη στο πρώτο στοιχείο της στήλης το οποίο είναι A(1,j). Η υπορουτίνα saxpy θα πάρει το A(1,j) σαν πρώτο στοιχείο ενός διανύσματος δίχως να ξέρει ότι αυτό το διάνυσμα είναι στήλη κάποιου πίνακα.

## 3.14 Η εντολή COMMON

Στη FORTRAN 77 δεν υπάρχουν καθολικές μεταβλητές, δηλαδή μεταβλητές που να είναι κοινές τόσο στο κυρίως πρόγραμμα όσο και στα υποπρογράμματα. Ο μόνος τρόπος να περάσουν πληροφορίες ανάμεσα σε υπορουτίνες είναι διαμέσου της λίστας παραμέτρων της υπορουτίνα, το οποίο πολλές είναι ασύμφορο διότι πολλές υπορουτίνες μοιράζονται μεγάλο πλήθος παραμέτρων. Σε αυτές τις περιπτώσεις μπορούμε να χρησιμοποιήσουμε την εντολή *common*. Αυτός είναι ένας τρόπος διάφορες υπορουτίνες να μοιράζονται συγκεκριμμένες μεταβλητές. Γενικά η χρήση της common θα πρέπει να είναι περιορισμένη. Ξεκινάμε με ένα παράδειγμα. Ας υποθέσουμε ότι έχουμε δύο παραμέτρους alpha και beta που τις χρειάζονται πολλές υπορουτίνες. Βλέπουμε παρακάτω πως μπορεί να γίνει αυτό με την χρήση της common.

```
      program main
      declaration
      real alpha, beta
      common /coeff/alpha, beta
c
      statments
      stop
      end
c
      subroutine sub1(arguments)
      declaration of arguments
      real alpha, beta
      common /coeff/alpha, beta
c
      statments
      return
      end
```



```
c
      subroutine sub2(arguments)
      declaration of arguments
      real alpha, beta
      common /coeff/alpha, beta
c
      statments
      return
      end
```

Σε αυτό το παράδειγμα η εντολή common με όνομα coeff περιέχει 2 μεταβλητές alpha και beta. Σε μια εντολή common μπορούν να υπάρχουν όσες μεταβλητές θέλουμε και οι οποίες δεν είναι απαραίτητο να είναι το ίδιου τύπου. Παρόλο που κάποια υπορουτίνα μπορεί να χρησιμοποιεί ένα μέρος των παραμέτρων της common, θα πρέπει να δηλωθούν όλες οι μεταβλητές της common στη υπορουτίνα.

Το γενικό ***συντακτικό*** της εντολής είναι το ακόλουθο,

common/όνομα/λίστα-μεταβλητών

Θα πρέπει να γνωρίζουμε ότι

α) Η εντολή common βρίσκεται αμέσως μετά τις δηλώσεις και πριν τις εντολές.

β) Διαφορετικές εντολές common πρέπει να έχουν διαφορετικά ονόματα. Επίσης μια μεταβλητή μπορεί να ανήκει σε παραπάνω από μία εντολή common.

γ) Οι μεταβλητές στη εντολή common δεν είναι απαραίτητο να έχουν το ίδιο όνομα, πράγμα όμως που συνίσταται. Εκείνο όμως που είναι απαραίτητο είναι να δηλώνονται με τη ίδια σειρά.

Σαν συνέχεια του προηγούμενου παραδείγματος έχουμε :

```
      subroutine sub3(arguments)
      declaration of arguments
      real alpha, beta
      common /coeff/a, b
c
      statments
      return
      end
```

Η δήλωση αυτή είναι ισοδύναμη με την προηγούμενη όπου είχαμε χρησιμοποιήσει τα alpha και beta. Συνίσταται πάντως να χρησιμοποιούμε πάντα τα ίδια ονόματα στις εντολές common. Ένα κακό παράδειγμα είναι το ακόλουθο :

```
      subroutine sub4(arguments)
      declaration of arguments
      real alpha, beta
      common /coeff/beta, alpha
c
      statments
      return
      end
```

Έτσι το alpha είναι beta και το ανάποδο. Αν υπάρχει κάτι τέτοιο τότε είναι σίγουρα λάθος και τέτοια λάθη είναι δύσκολα να βρεθούν σε ένα κώδικα.

### 3.14.1 Πίνακες στη εντολή common

Η εντολή common μπορεί να περιέχει και πίνακες το οποίο όμως δεν συνίσταται. Ο βασικός λόγος είναι ότι περιοριζόμαστε αρκετά. Ας δούμε το παρακάτω παράδειγμα. Ας υποθέσουμε ότι έχουμε τις παρακάτω δηλώσεις στο κυρίως πρόγραμμα ,

```
program main
integer nmax
parameter(nmax=20)
integer n
real A(nmax,nmax)
common /matrix/A,n,nmax
```

Κατά αρχήν η common περιέχει όλα τα στοιχεία του A καθώς επίσης και τους ακεραίους n, nmax. Φυσικά η ίδια δήλωση πρέπει να συμπεριληφθεί και σε όλες τις υπορουτίνες, π.χ.

```
subroutine sub1(arguments)
integer nmax
parameter(nmax=20)
integer n
real A(nmax,nmax)
common /matrix/A,n,nmax
```

Πίνακες μεταβλητής διάστασης δεν μπορούν να υπάρχουν στη common, οπότε η τιμή της nmax πρέπει να είναι η ίδια με αυτή του κυρίως προγράμματος. Από την άλλη μεριά η διάσταση του πίνακα πρέπει να είναι γνωστή και στη φάση της μεταγλωττισης οπότε η nmax θα πρέπει να οριστεί με την εντολή parameter. Αν σβήσουμε την parameter από την υπορουτίνα μιας και υπάρχει στη common αυτό θα ήταν συντακτικό λάθος. Το παράδειγμα αυτό μας δείχνει ότι δεν κερδίζουμε τίποτα δηλώνοντας πίνακες στη common. Ο προτιμότερος λοιπόν τρόπος για να περάσουμε πίνακες σε μια υπορουτίνα της FORTRAN 77 παραμένει η λίστα ορισμάτων της υπορουτίνας.

## 3.15 Η εντολή DATA

Η εντολή αυτή μας δίνει τη συνατότητα να εισάγουμε δεδομένα τα οποία είναι γνωστά όταν γράφεται ο κώδικας. Το *συντακτικό* της εντολής είναι,

data λιστα-μεταβλητών/λίστα-τιμών/, ... ,λιστα-μεταβλητών/λίστα-τιμών/

π.χ.

data m/10/, n/20/, x/2.5/, y/2.5/

το οποίο μπορεί να γραφτεί και ως εξής

data m/10/, n/20/, x,y/2*2.5/

Η παραπάνω εντολή είναι ισοδύναμη με τις παρακάτω καταχωρήσεις:

m=10
n=20
x=2.5
y=2.5

Η εντολή data είναι πιό συμπαγής οπότε και πιό βολική θα έλεγε κανείς. Παρατηρείστε επίσης και την σύντμηση της εντολής όταν πρόκειται να καταχωρηθούν ίδιες τιμές. Η εντολή data εκτελείται πρίν αρχίσει η εκτέλεση του προγράμματος, για αυτό το λόγο η εντολή αυτή χρησιμοποιειται βασικά στο κυρίως πρόγραμμα και όχι στις υπορουτίνες. Με την εντολή data μπορούμε να αναθέσουμε τιμές σε διανύσματα και πίνακες. Το επόμενο παράδειγμα δείχνει πως σιγουρεύουμε ότι πριν ξεκινήσει το πρόγραμμα ένας πίνακας είναι μηδενικός,



real a(10,20)
data a/200*0.0/

Μερικοί μεταγλωττιστές μηδενίζουν αυτόματα όλες τις μεταβλητές πριν αρχίσει η εκτέλση του προγράμματος, αλλά αυτός δεν είναι ο κανόνας. Αν ο κώδικας βασίζεται σε αυτή τη ιδιότητα θα πρέπει να κάτι ανάλογο με το προηγούμενο παράδειγμα. Φυσικά μπορούμε να αναθέσουμε μη-μηδενικές τιμές στους πίνακες και ακόμα να αναθέσουμε τιμές σε συγκεκριμμένα στοιχεία του πίνακα :

data a(1,1)/1.25/, a(2,1)/-33.2/, a(2,2)/1.0/

η ακόμα όταν η διάσταση του πίνακα είναι μικρή

integer v(5)
real b(2,2)
data v/10,2,-5,8,0/, b/1.0,-3.5,0.5,7.0/

Εδώ θα πρέπει να επισημάνουμε ότι η ανάθεση των τιμών γίνεται ανά στήλες.