Φώτης Φωτόπουλος – Αριστοτέλης Χαραλαμπάκης

Fortran

ΒΑΣΙΚΕΣ ΑΡΧΕΣ

ΕΝΤΟΛΕΣ

ΑΝΑΛΥΤΙΚΑ ΛΥΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΑ

ΛΥΜΕΝΑ ΘΕΜΑΤΑ ΕΞΕΤΑΣΕΩΝ

ΑΘΗΝΑ 1995

Περιεχόμενα

Πρόλογος

Οι σημειώσεις αυτές γράφτηκαν για τους φοιτητές του Εθνικού Μετσόβιου Πολυτεχνείου και καλύπτουν πλήρως το μάθημα της χρήσης Ηλεκτρονικών Υπολογιστών που περιλαμβάνει τη γλώσσα προγραμματισμού FORTRAN.Η σειρά για τους Η/Υ περιλαμβάνει άλλα δυο βοηθήματα , τη γλώσσα προγραμματισμού BASIC καθώς και την Αριθμητική Ανάλυση.

Σκοπός των σημειώσεων αυτών είναι να δοθούν με σαφήνεια και απλότητα όλες οι έννοιες και οι εφαρμογές που περιέχονται στη γλώσσα FORTRAN διατηρώντας όμως παράλληλα την επιστημονική αυστηρότητα και ευκρίνεια που πρέπει να διέπει τέτοιες προσπάθειες.

Ο καλύτερος τρόπος για την εκμάθηση της γλώσσας αυτής είναι η ταυτόχρονη επεξεργασία των προγραμμάτων σε Η/Υ. Αν αυτό καθίσταται αδύνατο, προτείνομε να αρχίσει η εκμάθηση καταρχήν από το 1ο κεφάλαιο , το οποίο διαπραγματεύεται γενικές γνώσεις πάνω στη FORTRAN. Οι έννοιες που περιγράφονται είναι απαραίτητες για την ορθή κατανόηση των υπολοίπων κεφαλαίων. Κατόπιν το 2ο κεφάλαιο μπορεί να το διαβάσει κανείς συντάσσοντας ταυτόχρονα τα προγράμματα του 3ου κεφαλαίου. Το 3ο κεφάλαιο περιέχει αναλυτικότατα λυμένα παραδείγματα με τον πιο απλό και κατανοητό τρόπο. Επίσης περιέχονται ορισμένα συμπληρωματικά στοιχεία της θεωρίας. Τέλος υπάρχει και το κεφάλαιο 4, στο οποίο επεκτείνεται η χρήση της FORTRAN σε πολλές εφαρμογές. Το κεφάλαιο 4 χωρίστηκε σε δυο ενότητες. Στην πρώτη έχομε κατατάξει προγράμματα για όλους , πολλά από αυτά ήταν και θέματα εξετάσεων, ενώ στη δεύτερη υπάρχουν δυσκολότερα προγράμματα για όσους δεν αρκούνται στη διεθνή φοιτητική σταθερά και θέλουν το "κάτι παραπάνω".

Όλα τα θέματα προέρχονται από παραδόσεις και φροντίστηκε ώστε να υπάρχει ομοιογένεια στην έκφραση καθώς και στην διατύπωση για να μη δημιουργούνται προβλήματα στους αναγνώστες.

Φ. Φωτόπουλος

Α. Χαραλαμπάκης

ΚΕΦΑΛΑΙΟ 1:Τα βασικά στοιχεία της Fortran

1.0 Εισαγωγή

Η γλώσσα προγραμματισμού Fortran 77 είναι μια από τις πολλές εκδόσεις της επιτυχημένης μαθηματικής φόρμουλας επιλύσεων Standard Fortran. Αν και ως γλώσσα δεν έχει γραφικές δυνατότητες και χρησιμοποιεί ξεπερασμένες θεωρητικά τεχνικές, παρόλα αυτά αποτελεί ένα δυνατό εργαλείο για επιλύσεις μαθηματικών προβλημάτων κυρίως.

Σ' αυτό συντελούν δυο λόγοι, η μεγάλη βιβλιοθήκη προγραμμάτων που έχουν στη διάθεση τους αυτοί που ασχολούνται με τη γλώσσα καθώς και η μαθηματικής ακρίβειας δομή που απαιτείται για υλοποίηση εφαρμογών.

1.1 Οι χαρακτήρες της Fortran

Το αλφάβητο της Fortran είναι ιδιαίτερα φτωχό. Δέχεται μόνο τα 26 γράμματα της αλφαβήτου και τους 10 αριθμητικούς χαρακτήρες του δεκαδικού συστήματος καθώς και το κενό (διάστημα) μαζί με τους χαρακτήρες + - * / = ( ) ' , . $: .Επίσης πολλές φορές συναντάμε (στη βιβλιογραφία κυρίως) το κενό ως b.

1.2 Μεταβλητές

Στη Fortran έχει μεγάλη σημασία το αρχικό γράμμα της μεταβλητής , το οποίο καθορίζει αν είναι ακέραια ή πραγματική (δηλαδή δεκαδική).Οι μεταβλητές που αρχίζουν από τα I,J,K,L,M,N είναι ακέραιες , όλες οι άλλες πραγματικές.

Προσοχή πρέπει να δοθεί στο γεγονός ότι το όνομα μιας μεταβλητής δεν πρέπει να υπερβαίνει τους 6 χαρακτήρες. Καλό είναι επίσης να μην χρησιμοποιούνται κενά στα ονόματα.

Δηλαδή οι μεταβλητές I,K123,MK,MIN,MAX,LALAKH κτλ περιέχουν ακέραιους αριθμούς ενώ οι FOTIS, THE, GOD περιέχουν σαφώς πραγματικές.

1.3 Αριθμοί στη Fortran

Παραδείγματα ακεραίων αριθμών είναι τα παρακάτω: 0,-128,1919,57,+60,-1.

Παραδείγματα πραγματικών αριθμών: 0.,5.,-128.,88.82,-99.99,κλπ. Δηλαδή αν σε ακέραιο βάλομε στο τέλος του μια τελεία τότε έχομε πραγματικό αριθμό. Όλοι εξάλλου οι δεκαδικοί αριθμοί είναι πραγματικοί.

1.4 Εκθετική μορφή των αριθμών

Ιδιαίτερα χρήσιμη για πολύ μεγάλους ή πολύ μικρούς αριθμούς είναι η δυνατότητα παράστασής των με τη βοήθεια δυνάμεων. Πιο συγκεκριμένα ο αριθμός 0.0000001 είναι προτιμότερο να γραφτεί ως 1E-07 ή 1E-7.

Ομοίως ο αριθμός 5E12 είναι ο $5*10^{12}$ κοκ. Βεβαίως ισχύει 7E0=7.Γενικά μετά το Ε ακολουθεί ο εκθέτης με το πρόσημό του (αν παραλείπεται τότε εννοείται το θετικό) του 10.

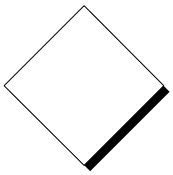
1.5 Διαγράμματα Ροής

Αυτά είναι λογικά διαγράμματα που περιγράφουν με ποιο τρόπο "ενεργεί" ο υπολογιστής κατά την εκτέλεση του εκάστοτε προγράμματος, χωρίς όμως να μας δίνει λεπτομέρειες. Δηλαδή παρουσιάζει με γενικό τρόπο το πως λειτουργεί το πρόγραμμα. Για να φτιάξουμε ένα διάγραμμα ροής πρέπει καταρχήν να γνωρίζομε τους συμβολισμούς που χρησιμοποιούνται.

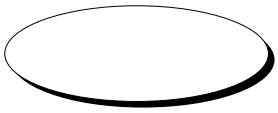
Στη συνέχεια παρουσιάζονται τα δομικά στοιχεία που απαρτίζουν ένα διάγραμμα ροής.

Το **ορθογώνιο** περιέχει ομάδα γενικών εντολών. (Δηλαδή όλες τις αριθμητικές πράξεις)

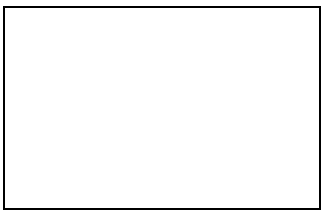
Ο **ρόμβος** περιέχει τις εντολές ελέγχου μεταφοράς ήτοι τις εντολές σύγκρισης.



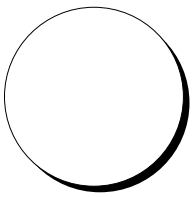
Η **έλλειψη** χρησιμοποιείται για εντολές μεταφοράς χωρίς όρους, δηλαδή η εκτέλεση του προγράμματος θα κινηθεί αναγκαστικά προς κάποια κατεύθυνση.



Το **τραπέζιο** περιέχει εντολές εκτύπωσης ή ανάγνωσης δεδομένων (read & write).



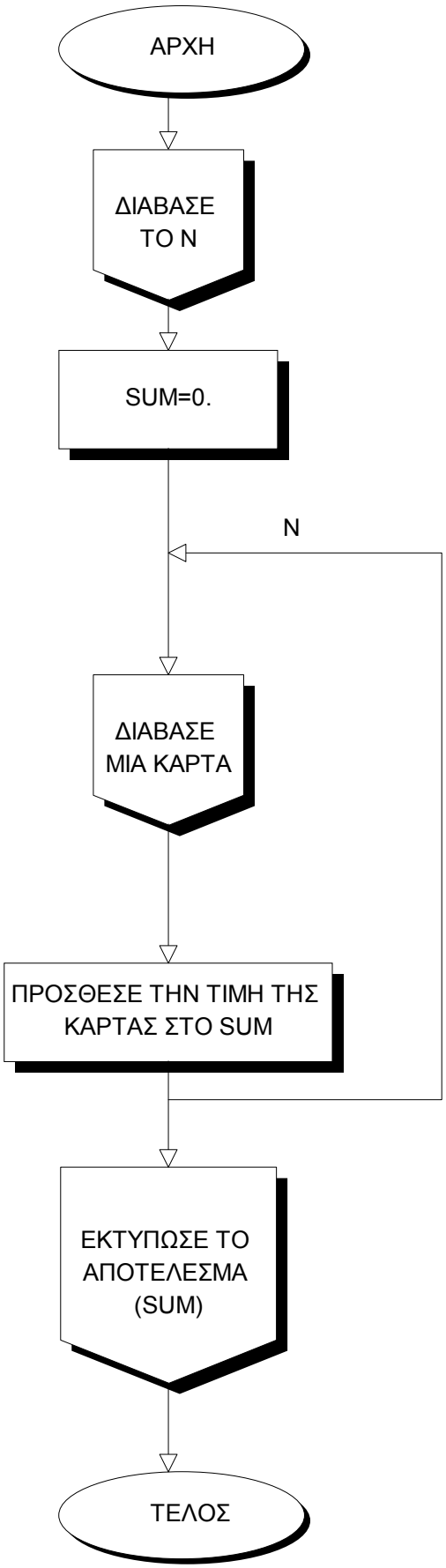
Τέλος έχομε τον **κύκλο** που συνδέει δυο σημεία του ιδίου προγράμματος (συνήθως για Goto εντολές).



Γενικά τα διαγράμματα ροής χρησιμοποιούνται για να γίνει απλούστερη η σύνταξη κάποιου προγράμματος, του οποίου αποτελεί τη "στρατηγική".

Παράδειγμα

Να συνταχθεί το διάγραμμα ροής και η αντίστοιχη ονοματολογία ενός προγράμματος που να βρίσκει το άθροισμα των αριθμών μιας λίστας.


ΑΡΧΗ
ΔΙΑΒΑΣΕ ΤΟ Ν
SUM=0.
Ν
ΔΙΑΒΑΣΕ ΜΙΑ ΚΑΡΤΑ
ΠΡΟΣΘΕΣΕ ΤΗΝ ΤΙΜΗ ΤΗΣ ΚΑΡΤΑΣ ΣΤΟ SUM
ΕΚΤΥΠΩΣΕ ΤΟ ΑΠΟΤΕΛΕΣΜΑ (SUM)
ΤΕΛΟΣ

Καλώ με N το πλήθος των αριθμών. Επειδή δε γνωρίζω ποια ακριβώς είναι η τιμή του N, ζητώ από το χρήστη να τον καθορίσει. Το γράμμα N ταιριάζει επίσης για τον καθορισμό του αριθμού τιμών της λίστας καθόσον περιγράφει ακέραιο αριθμό.

Ας καλέσω SUM το άθροισμα (πραγματική μεταβλητή γιατί μπορεί οι αριθμοί να είναι δεκαδικοί) οπότε πριν αρχίσω να προσθέτω τους αριθμούς , χρειάζεται να το μηδενίσω. Δηλαδή να πω SUM=0. Χρειάζεται ιδιαίτερη προσοχή στο σημείο αυτό. Για το μηδενισμό του SUM που είναι πραγματική μεταβλητή θα ήταν **λάθος** να χρησιμοποιήσω την έκφραση SUM=0 χωρίς δίπλα από το μηδέν να βάλω μια τελεία. Στη Fortran συνεπώς υπάρχει μεγάλη διαφορά μεταξύ του μηδέν που ακολουθείται από μια τελεία και του απλού μηδέν. Το πρώτο μηδενίζει μια πραγματική μεταβλητή, ενώ το δεύτερο ακέραια.

Κατόπιν πρέπει να ορίσω ένα βρόχο δηλαδή μια διαδικασία που θα επαναληφθεί τόσες φορές όσες και το πλήθος των στοιχείων. Θα διαβάζεται ένας αριθμός και ύστερα θα προστίθεται εις το SUM το οποίο εκφράζει το άθροισμα όλων των αριθμών. Όταν θα τελειώσουν όλοι οι αριθμοί, θα πρέπει να γίνει εκτύπωση του αποτελέσματος και να σταματήσει η εκτέλεση του προγράμματος.

1.6 Κανόνες πληκτρολόγησης εντολών

Παλιότερα οι εντολές της Fortran εισάγονταν στον Η/Υ με κάρτες. Μάλιστα σε κάθε κάρτα υπήρχε μια μόνον εντολή. Αν λοιπόν ένα πρόγραμμα αποτελούνταν από 50 εντολές, τότε έπρεπε να χρησιμοποιηθούν 50 διαφορετικές κάρτες. Σήμερα βέβαια αυτό έχει εγκαταλειφθεί, αλλά για λόγους συμβατότητας έχουν παραμείνει οι ίδιοι κανόνες εισαγωγής εντολών στις περισσότερες εκδόσεις Fortran καθώς και σε αυτή που διδάσκεται σήμερα. Έτσι:

Η κάρτα θεωρείται ότι έχει 80 στήλες. Χωρίζεται δε σε 5 πεδία εκ των οποίων εμείς θα χρησιμοποιήσομε τα τέσσερα. Το πρώτο είναι η πρώτη στήλη. Αν αυτή διατρηθεί με (*) ή με (C) τότε ο μεταφραστής την αγνοεί. Αυτό είναι ιδιαίτερα χρήσιμο για την παρεμβολή σχολίων.

Το δεύτερο μέρος είναι οι στήλες 2,3,4 και 5.Σ'αυτές εισάγεται (προαιρετικά) ο αριθμός της εντολής. Καταρχήν δεν είναι απαραίτητο να είναι αριθμημένες οι εντολές και αν αριθμηθούν ο υπολογιστής δεν ακολουθεί την αρίθμηση, αλλά απλώς τις αναγνωρίζει αν από κάποιο σημείο του προγράμματος τις καλέσομε. Είναι απαραίτητο να γνωρίζομε ότι ο μεταφραστής θα εκτελέσει τις εντολές της Fortran με τη σειρά που είναι γραμμένες στο χαρτί μας (κάρτα). Την εντολή 1500 θα τη γράψουμε ως εξής: το 1 στην στήλη 2, το 5 στη στήλη 3 και στις στήλες 4 και 5 από ένα μηδέν. Το 150 θα το γράψουμε ομοίως αρχίζοντας από τη στήλη 3 (η 2 μένει κενή) κοκ.

Το τρίτο μέρος χρησιμοποιείται για να δείξουμε ότι η προηγούμενη εντολή συνεχίζεται και σε αυτή τη γραμμή , καθότι είναι πιθανό να μην χωράει μια εντολή με μια γραμμή. Έτσι λοιπόν αν διατρήσουμε ένα χαρακτήρα στη στήλη 6 (εκτός του μηδενός) τότε ο υπολογιστής θα εκτελέσει τις δυο (ή περισσότερες) εντολές ως μια. Προσοχή στο ότι αυτό δε γίνεται με τις γραμμές σχολίων. Γραμμές σχολίων δεν μπορούν να παρεμβάλλονται μεταξύ μιας γραμμής που συνεχίζεται σε πολλές κάρτες.

```
CbbbbbΤο πρόγραμμα αυτό εφαρμόζει τους κανόνες που δώσαμε.
CbbbbbΠροσέξτε ότι όπου b είναι το κενό (διάστημα).
   10bΠρώτη εντολή που εκτελείται
 1500bΔεύτερη εντολή που εκτελείται
      Αυτή είναι η τρίτη εντολή που όμως καταλαμβάνει τρεις γραμμές
     1αυτή είναι η δεύτερη γραμμή
     2και αυτή η τρίτη γραμμή που καταλαμβάνει η εντολή νο 3.
 9999bΚάθε εντολή μπορεί να έχει αριθμηθεί από το 1 ως το 9999.
```

- Δεν είναι δυνατόν δυο εντολές να έχουν τον ίδιο αριθμό.
- Οι εντολές τοποθετούνται στις στήλες 7 ως 72.
- Οι στήλες 73 ως 80 σήμερα αγνοούνται, παλιότερα τοποθετούνταν χαρακτηριστικά ψηφία αναγνωρισμού των καρτών για να μπαίνουν ανά πάσα στιγμή στην σωστή σειρά.

Οι κανόνες αυτοί θα γίνουν πιο κατανοητοί με την πληθώρα προγραμμάτων των κεφαλαίων 3 και 4 που ακολουθούν.

1.7 Εισαγωγή δεδομένων στον υπολογιστή

Σήμερα αυτό γίνεται μέσω του πληκτρολογίου. Παλιότερα αυτό γινόταν επίσης με χρήση καρτών. Αυτές διέφεραν από τις κάρτες που χρησιμοποιούνταν για εισαγωγή εντολών , στο ότι δεν χωρίζονταν σε πέντε μέρη , αλλά ήταν ενιαίες , και τα δεδομένα μπορούσαν να γραφτούν σε οποιεσδήποτε στήλες τους.

Στα προγράμματα που παρατίθενται, όλα τα δεδομένα εισάγονται μέσω του πληκτρολογίου. Αξίζει τέλος να σημειωθεί ότι πρέπει ακόμα και σήμερα να δείχνομε στον μεταφραστή της Fortran σε ποιες στήλες έχουν εισαχθεί τα δεδομένα με τη διαφορά ότι αναφερόμαστε στις στήλες της οθόνης του Η/Υ.

1.8 Δυνατότητες και εφαρμογές της Fortran

Τα προγράμματα που ακολουθούν αποτελούνται μόνον από τις εντολές που διδάσκονται στο μάθημα της χρήσης ηλεκτρονικών υπολογιστών του ΕΜΠ. Για το λόγο αυτό δεν υπάρχει καμία αναφορά σε χρήση αρχείων καθώς και σε γραφικές απεικονίσεις (που ούτως ή άλλως εκτός της 5.1 έκδοσης της Fortran οι δυνατότητες για γραφικές απεικονίσεις είναι ελάχιστες).

Τα είδη των προγραμμάτων που δύναται να ζητηθούν είναι επίσης περιορισμένα. Αποτελούνται από προγράμματα μαθηματικών πράξεων, από προγράμματα στατιστικής (όπως για εύρεση μέσων όρων) και από σύνθετα μαθηματικά προβλήματα. Για την καλύτερη κατανόηση συνίσταται η διεξοδική μελέτη του κεφαλαίου 2, που περιγράφει τη χρήση των εντολών της Fortran και δίνει πολλά κατατοπιστικά παραδείγματα.

Η Fortran βρίσκει ακόμα και σήμερα εφαρμογές σε σύνθετα μαθηματικά προβλήματα, όχι όμως με μόνα εργαλεία τις εντολές που περιέχονται στη διδακτέα ύλη. Για το λόγο αυτό αποφύγαμε να προσθέσομε τέτοια προβλήματα, παρά μόνο λίγα της δεύτερης ενότητας του κεφαλαίου 4.

1.9 Συντάσσοντας ένα πρόγραμμα

Ένα από τα μεγάλα μειονεκτήματα της Fortran είναι η έλλειψις περιβάλλοντος εργασίας. Δηλαδή για να εκτελέσομε ένα πρόγραμμα είμαστε αναγκασμένοι να ακολουθήσομε τα εξής βήματα:

- Αρχικά γράφουμε το πρόγραμμα σε έναν editor αρχείων κειμένου.
- Καλούμε το μεταγλωττιστή της Fortran F77L <όνομα αρχείου> ή με όποιο άλλο όνομα είναι διαθέσιμος.Αυτός δημιουργεί αρχείο με συλλογή αντικειμένων (objects).
- Κατόπιν καλούμε το πρόγραμμα που συνδέει τα αντικείμενα σε εκτελέσιμη γλώσσα μηχανής (executable machine code) σε .EXE δηλαδή μορφή. Το πρόγραμμα αυτό είναι παίρνει το αρχείο με κατάληξη .OBJ που δημιουργεί ο μεταγλωττιστής και το κάνει εκτελέσιμο (LINKer). Για να γίνει αυτό γράφομε LINK <όνομα αρχείου.obj>
- Τώρα από την γραμμή εντολών (command line) πληκτρολογούμε το όνομα του προγράμματος που είναι πλέον έτοιμο να τρέξει.

Αυτή είναι η μόνη διαδικασία για να γίνει εκτελέσιμο ένα πρόγραμμα που γράφτηκε στην εντολή Fortran. Με άλλα λόγια, κάθε φορά που επιθυμούμε να ελέγξουμε ένα πρόγραμμα, πρέπει να το μετατρέπομε σε εκτελέσιμη μορφή σύμφωνα με τα ανωτέρω.

ΚΕΦΑΛΑΙΟ 2: Οι εντολές της Fortran

2.1 Εντολές Εισόδου

2.1.1 Read (y,xxxx) A_1 , A_2 ,...,A_n

Γενικά: Η εντολή αυτή είναι αντίστοιχη της INPUT εντολής της basic. Χρησιμοποιείται δε για την ανάγνωση δεδομένων από το πληκτρολόγιο. Η μόνη διαφορά είναι ότι δεν μπορεί να χρησιμοποιηθεί μόνη της αλλά πρέπει **απαραίτητα** να συνοδεύεται από την εντολή format.

Παράμετροι: Το πρώτο στοιχείο της εντός της παρένθεσης (y) δείχνει το μέσον εισόδου που μπορεί να είναι κάρτα, πληκτρολόγιο, αρχείο, κλπ. Εμείς θα το βάζουμε πάντα ίσο με 5, σε όλες τις περιπτώσεις εννοώντας την κάρτα ως μέσον εισόδου.
Το δεύτερο στοιχείο (xxxx) δηλώνει τον αριθμό της εντολής που βρίσκεται η format. Η εντολή format είναι από τις ελάχιστες εντολές που οπωσδήποτε πρέπει να είναι αριθμημένη αλλιώς παίρνομε μήνυμα λάθους.

Τέλος τα A_1, A_2,...,A_η είναι τα ονόματα των μεταβλητών στα οποία θα αποθηκευτούν οι τιμές που θα διαβαστούν από την (τις) κάρτα(ες). Σημειώνεται ότι μπορούμε να χρησιμοποιήσομε τόσο ονόματα μεταβλητών που παίρνουν ακέραιες τιμές όσο και ονόματα που παίρνουν πραγματικές τιμές.

Παραδείγματα: Παρακάτω δίνονται ορισμένα παραδείγματα της χρήσης της εντολής read.

```
READ(5,1000) I5,IJK,AJH
```

Στην εντολή 1000 πρέπει να υπάρχει η format. Θα διαβαστούν τρεις μεταβλητές.Οι δυο πρώτες θα είναι ακέραιες (αρχίζουν με I) ενώ η τρίτη πραγματική (αρχίζει με A).

```
READ(5,150) A(I)
```

Στην εντολή 150 υπάρχει η format. Θα διαβαστεί ένα στοιχείο κάποιου πίνακα που περιέχει πραγματικές μεταβλητές (αρχίζει από A) . Τώρα το ποιο στοιχείο του μονοδιάστατου αυτού πίνακα θα διαβαστεί, αυτό εξαρτάται από την τιμή του I. Δηλαδή για I=4 , θα διαβαστεί το στοιχείο του πίνακα που βρίσκεται στην 1η στήλη και στην 4η γραμμή.

2.1.2 xxxx format(<έκφραση>)

Γενικά: η εντολή αυτή είναι απαραίτητη για την εντολή READ. Αυτό συμβαίνει γιατί ενώ ο μεταγλωττιστής της Fortran γνωρίζει ότι για παράδειγμα θα διαβαστούν από κάρτα τρεις μεταβλητές καθώς και τον τύπο των μεταβλητών αυτών (ακέραια ή πραγματική) δεν γνωρίζει σε ποιο σημείο της κάρτας είναι τοποθετημένα τα ψηφία των μεταβλητών , ούτε επίσης πόσα είναι αυτά . Δηλαδή δεν μπορεί να διαβάσει ο υπολογιστής έναν αριθμό αν προηγουμένως δεν του διευκρινίσομε πόσα θα είναι τα ψηφία του.

Παράμετροι: θα τους χωρίσομε σε τέσσερις κατηγορίες. Πρώτον η παράμετρος που σχετίζεται με ακεραίους, δεύτερον αυτή που σχετίζεται με πραγματικούς , ύστερα αυτή που δίνει τα κενά (αν υπάρχουν) μεταξύ των αριθμών που έχουν διατρηθεί και τέλος αυτή που εκτυπώνει αλφαβητικούς χαρακτήρες στην οθόνη.

2.1.2.1 Τύπος Ix

Με το γράμμα I εννοούμε ακέραιο (integer) και με το x δηλώνομε τον αριθμό των ψηφίων του συμπεριλαμβάνοντας το αρνητικό πρόσημο (αν είναι θετικό, τότε παραλείπεται εκτός και αν έχει διατρηθεί).

Παράδειγμα 1ο:

```
     READ (5,1000) INT1,INT2,INT3,INT4
1000 FORMAT(I4,I5,I1,I3)
```

Η δε κάρτα έχει διατρηθεί ως εξής:

```
1234bbb129-12bbbb
```

τότε INT1=1234, INT2=12, INT3=9 ΚΑΙ INT4=-12

Παράδειγμα 2ο:

```
     READ(5,1000) INT1,INT2,INT3
```

```
1000 FORMAT (I5,I2,I4)
```

Η δε κάρτα έχει διατρηθεί ως εξής:

```
b+12b-11bbbbbbb
```

τότε INT1=+120=120, INT2=-1, INT3=1000

Δηλαδή όσες στήλες είναι άδειες θεωρούνται ότι έχουν διατρηθεί με το μηδέν.

Για παράδειγμα το INT2 του 1ου παραδείγματος στην πραγματικότητα είναι το 00012=12. Αυτό ισχύει γενικότερα και πρέπει να προσεχθεί ιδιαίτερα.

Παράδειγμα 3ο:

```
     READ(5,1000) L,M,N
1000 FORMAT(I4,I1,I2)
```

Η δε κάρτα έχει διατρηθεί ως εξής:

+312678 οπότε παίρνομε L=+312=312, M=6 και N=78

Αν ήταν επίσης διατρημένη :

b45bb-7 τότε το αποτέλεσμα θα ήταν L=0450=450, M=b=0 , και N=-7.

2.1.2.2 Τύπος Fx.d

Με το γράμμα F εννοούμε τους πραγματικούς αριθμούς (Floating number). Όπως είδαμε ένας πραγματικός αριθμός δύναται να αποθηκευθεί σε μεταβλητή που το όνομα της **δεν** αρχίζει από I,J,K,L,M,N. Κατά τη γραφή του πραγματικού αριθμού, εάν αυτός δεν έχει δεκαδικά ψηφία για να τον διαχωρίζομε από τους ακέραιους προσθέτομε στο τέλος του μια τελεία (**.**) ή μια τελεία ακολουθούμενη από ένα μηδενικό (**.**0). Όπως ήδη είδαμε, το 5 είναι ένας ακέραιος, αλλά το 5**.** ή το 5.0 είναι προφανώς πραγματικός αριθμός.

Χρησιμοποιούμε λοιπόν το **Fx.d** όταν πρόκειται να διαβαστεί ένας πραγματικός αριθμός από κάρτα. Με το **x** λέμε στο μεταφραστή της Fortran ποιο είναι το μήκος του αριθμού. Στο μήκος συμπεριλαμβάνεται το αρνητικό πρόσημο (αν υπάρχει) , το δεκαδικό

σημείο και φυσικά το πλήθος των ψηφίων του. Για παράδειγμα ο 5**.** έχει μήκος 2, ο +99**.**01 έχει μήκος 6, αλλά και μήκος 5 αν παραλείψουμε το θετικό του πρόσημο το οποίο δεν παίζει κανένα ιδιαίτερο ρόλο. Ο -7**.**891 έχει όμως μήκος 6 και ο -99**.** μήκος 4.

Μπορεί τώρα ο Η/Υ να διαβάσει έναν πραγματικό αριθμό , του οποίου θα γνωρίζει πόσα είναι τα ψηφία του και ποια είναι αυτά, πρέπει όμως για να ολοκληρωθεί η λειτουργία αυτή να του ορίσουμε **πόσα** είναι τα δεκαδικά ψηφία του αριθμού αυτού. Δηλαδή να ορίσουμε το **d**. Για τους αριθμούς των παραπάνω παραδειγμάτων είναι d=0, 2, 3 και 0 αντίστοιχα.

Δηλαδή, για να διαβαστεί ο αριθμός +99**.**01 θα πρέπει εντός του format να συμπεριλάβομε την έκφραση **F6.2** όπου το 6 δίνει το μήκος και το 2 το πλήθος των δεκαδικών του ψηφίων. Ιδιαίτερη προσοχή πρέπει να δοθεί στα παρακάτω:

Αν υποτεθεί ότι το d του τύπου Fx.d που δώσαμε είναι λάθος, δηλαδή ο αριθμός στην κάρτα είναι ο 99**.**01 και εμείς δώσαμε F5.3 τότε ο μεταφραστής θα προτιμήσει την τιμή που είναι στην κάρτα , δηλαδή την 99.01 και όχι την 9.901 όπως θα περιμέναμε. Δηλαδή η Fortran παρακάμπτει πιθανά λάθη στην τοποθέτηση του δεκαδικού σημείου, καλό είναι όμως να το προσέχομε ιδιαίτερα για να εκμηδενίσουμε την περίπτωση σφάλματος.

Ένα άλλο βασικό σημείο είναι να έχομε δώσει λάθος το x.Ας υποτεθεί λοιπόν ότι προκειμένου να διαβαστεί ο αριθμός 123.45 εμείς δώσαμε F5.2 και όχι F6.2 όπως κανονικά έπρεπε. Τότε ο μεταφραστής θα μας εκτυπώσει ***** (πέντε αστερίσκους) που σημαίνει ότι ο αριθμός στην κάρτα είναι μεγαλύτερος από το μήκος που περίμενε (και που υπάρχει βέβαια στην εντολή format). Δηλαδή είναι **απαραίτητο** να έχομε σωστό μήκος αριθμού, αλλιώς το πρόγραμμα **δεν** θα τρέχει.

Ακολουθούν πολλά παραδείγματα για να γίνουν κατανοητές οι έννοιες αυτές:

```
Κάρτα >b153.2bb37.
     READ(5,1000) A,B
1000 FORMAT(F6.1,F5.0)
Μετά A=153.2 και B=37.

Κάρτα >.1b5b2b7.b
```

```
      READ(5,1000)U,V,X,Y
1000 FORMAT(F2.0,F3.2,F1.0,F4.3)
```

Μετά U=.1 (υπερισχύει η θέση του δεκαδικού στην κάρτα)

V=0.50=.5, X=2.0=2. , Y=7.0=7. (ομοίως με το U)

```
Κάρτα >bbbbb99887.bb-77.22b+45.b
      READ(5,1000) A,B,C
1000 FORMAT(F11.0, F9.3 , F5.1)
```

Τελικά A=0000099887.0=99887. , B=-77.220=-77.22, C=+45.0=45.

Μπορούμε επίσης να συνδυάσομε ακέραιους και πραγματικούς αριθμούς σε μια κάρτα και να τους διαβάσουμε χρησιμοποιώντας ένα μόνο READ και μια μόνο εντολή FORMAT.

```
Κάρτα >b12bb-7.2b-9
             READ(5,1000) I1,R1,I2
    1000 FORMAT(I4,F6.3,I2)
```

Οπότε παίρνομε:

I1=0120=120 (μην ξεχνάμε ότι οι κενές στήλες θεωρούνται μηδενικά)

R1=-7.20=-7.2 (διότι υπερισχύει η θέση του δεκαδικού σημείου στην κάρτα)

I2=-9

2.1.2.3 Τύπος nX

Υπάρχει επίσης η περίπτωση να μην έχουν διατρηθεί οι αριθμοί σε μια κάρτα με τη σειρά. Ας πούμε για παράδειγμα ότι έχομε διατρήσει στις στήλες 3,4,5 και 6 έναν τετραψήφιο ακέραιο και στις στήλες 11,12,13 και 14 έναν τριψήφιο πραγματικό ο οποίος έχει ένα δεκαδικό ψηφίο, δηλαδή έχομε την κάρτα:

```
   >bb1234bbbb12.3bbbb
```

Καταρχήν για την ανάγνωση των δυο αριθμών θα μπορούσαμε να πούμε:

```
      READ(5,1000) I,A
1000 FORMAT(I6,F8.1)
```

οπότε I=001234=1234 και A=000012.3=12.3

Είναι όμως πολύ πιο εύκολο να πούμε ότι οι πρώτες δυο στήλες είναι κενές, ακολουθεί τετραψήφιος ακέραιος και ύστερα 4 στήλες κενές κοκ. Ελαχιστοποιείται έτσι η πιθανότης να κάνομε κάποιο λάθος απροσεξίας. Τον αριθμό των κενών στηλών τον καλούμε **n** και για να τον δηλώσομε στην έκφραση της εντολής format τον τοποθετούμε πριν το X. Οι στήλες αυτές αγνοούνται. Προσοχή: ενώ στα I,F τα x και d μπαίνανε μετά τώρα το n προηγείται του X. Για το παράδειγμά μας θα μπορούσαμε εντελώς ισοδύναμα να γράψομε:

```
      READ(5,1000) I,A
 1000 FORMAT(2X,I4,4X,F4.1,4X)
```

το οποίο είναι σαφώς πιο κατανοητό και εύκολο στην παρακολούθησή του. Άλλη περίπτωση που μπορεί να φανεί εξαιρετικά χρήσιμη η χρησιμοποίηση του τύπου nX είναι όταν εκτυπώνομε αριθμούς στην οθόνη ή στον εκτυπωτή (βλ. παρακάτω τη write εντολή). Αν εκτυπώναμε τους αριθμούς στη σειρά, θα μπλέκονταν καθώς όταν τελείωνε ο ένας θα άρχιζε ο άλλος. Όμως τώρα θα αφήνομε κενά μεταξύ των και θα είναι ευκολότερη ανάγνωσή των.

2.1.2.4 Τύπος επαναλαμβανόμενων πεδίων

Τα επαναλαμβανόμενα πεδία είναι μια απλή μέθοδος γραφής της έκφρασης εντός της εντολής format που το μόνο που προσδίδει είναι ταχύτητα κατά την πληκτρολόγηση και μείωση του μεγέθους της εντός της format έκφρασης. Ας αρχίσομε με ένα απλό παράδειγμα για να γίνει σαφής η έννοια αυτή. Έστω έχομε ένα πρόγραμμα που διαβάζει 10 ακέραιους αριθμούς. Οι αριθμοί αυτοί είναι τριψήφιοι και διαχωρίζονται με παρεμβολή δυο κενών χαρακτήρων. Ζητείται να διαβαστούν αυτοί οι αριθμοί με εντολή READ από κάρτα.

```
     READ(5,1000) I1,I2,I3,I4,I5,I6,I7,I8,I9,I10
1000 FORMAT(I3,3X,I3,3X,I3,3X,I3,3X,I3,3X,I3,3X,I3,3X,I3,3X,I3,3X,I3)
```

Κατά την πληκτρολόγηση των παραπάνω διαπιστώνομε:

- η διαδικασία είναι πολύ επίπονη

- η εντολή format μόλις που χωράει σε μια γραμμή , αν είχαμε 11 αριθμούς θα έπρεπε να γραφτεί σε δυο κάρτες (κάθε κάρτα μην ξεχνάμε είναι και από μια εντολή) πράγμα όχι τόσο εύκολο και απλό.

Για την αντιμετώπιση των παραπάνω προβλημάτων χρησιμοποιούμε το επαναλαμβανόμενο πεδίο . Καταρχήν εντοπίζομε ποια έκφραση της format επαναλαμβάνεται διαρκώς και δημιουργεί το πρόβλημα. Προφανώς για το παράδειγμά μας η έκφραση είναι η I3,3X , παρόλο που "λείπει" ένα 3X στο τέλος (και να εκτυπωθεί παίρνουμε ακριβώς το ίδιο πράγμα). Η έκφραση αυτή επαναλαμβάνεται 10 φορές, συνεπώς έχω 10(I3,3X). Αυτό ακριβώς γράφω στην εντολή format και καλώ επαναλαμβανόμενο πεδίο. Δηλαδή έχω:

```
1000 FORMAT(10(I3,3X))
```

Αν ήθελα να αφήσω δυο στήλες κενές πριν εκτυπώσω τον πρώτο αριθμό τότε θα έγραφα:

```
1000 FORMAT(2X,10(I3,3X))
```

2.1.2.5 Χρησιμοποίηση Format από πολλές εντολές

Είπαμε ότι η εντολή READ(5,1000) σημαίνει ότι θα διαβαστούν από κάρτες αριθμοί οι οποίοι περιγράφονται από μια εντολή format που βρίσκεται στην γραμμή με αριθμό 1000. Δεν είναι δεσμευτικό να χρησιμοποιηθεί μόνον από μια εντολή READ ή WRITE μια συγκεκριμένη εντολή format. Με άλλα λόγια αν θέλομε να διαβάσομε και να εκτυπώσομε τρεις αριθμούς οποιουδήποτε τύπου, μπορούμε να χρησιμοποιήσομε την ίδια εντολή format εφόσον κάτι τέτοιο μας εξυπηρετεί και εφόσον οι αριθμοί διαβάζονται και εκτυπώνονται στις ίδιες στήλες .

Ας πάρομε ένα παράδειγμα που διαβάζει λοιπόν τρεις αριθμούς από κάρτες. Η εντολή WRITE αναπτύσσεται αμέσως μετά και μπορεί να ανατρέξει κανείς εκεί πριν διαβάσει τα επόμενα.

```
     READ (5,1000) I,J,R1
1000 FORMAT(1X,I4,3X,I2,3X,F10.4)
```

Αν υποτεθεί ότι θέλομε να εκτυπώσομε κάποια στιγμή μέσα στο πρόγραμμα τους τρεις αριθμούς, με την ίδια όμως κατανομή σε σχέση με τις στήλες, δηλαδή αφήνοντας αρχικά ένα κενό , μετά τυπώνουμε τον πρώτο ακέραιο ακολουθούμενο από τρεις κενές στήλες , μετά το δεύτερο κοκ, τότε μπορούμε να γράψουμε την εξής γραμμή:

```
WRITE(6,1000) I,J,R1
```

οπότε και γίνεται οικονομία στη μνήμη (λιγότερες εντολές αποθηκεύονται).

2.1.2.6 Τύπος nH

Πολλές φορές κατά την εκτύπωση ενός αριθμού, θέλουμε να φαίνεται και κάποιο στοιχείο του. Δηλαδή σε ένα στατικό πρόγραμμα που επιλύει ένα δικτύωμα, θέλουμε όταν γράφουμε τα αποτελέσματα των εξισώσεων στερεοστατικής ισορροπίας, να προσδιορίζομε τι αντιπροσωπεύει ο αριθμός που εκτυπώνεται. Αυτό γίνεται με τον τύπο **nH** όπου n το πλήθος των χαρακτήρων που εκτυπώνονται.

Παράδειγμα: θέλομε να υπολογίσομε τον όγκο ενός κύβου, του οποίου η ακμή διαβάζεται από κάρτα. Συντάσσομε λοιπόν το παρακάτω πρόγραμμα:

```
     READ (5,1000) A
1000 FORMAT(1X,F10.4)
     V=A**3.
     WRITE (6,1000) V
     STOP
     END
```

Όπου A η ακμή του κύβου και V ο όγκος του. Προσέξτε ότι υψώνεται η ακμή στο 3. θα ήταν το ίδιο αν θέταμε 3 χωρίς τελεία. Συμφέρει να εκτυπωθεί το αποτέλεσμα με το ίδιο format. Επειδή όμως ενδέχεται να ξεχάσουμε κάποια στιγμή πχ σε τι μονάδες είναι το αποτέλεσμα ή τι ακριβώς αντιπροσωπεύει, μπορούμε να εκτυπώνομε ταυτόχρονα κάποιες πληροφορίες γι’ αυτό. Έτσι αντικαθιστούμε τη γραμμή WRITE με τις παρακάτω γραμμές:

```
       WRITE(6,2000) V
  2000 FORMAT(1X,18Hogkos kybou (m^3)=,F10.4)
```

Για να εκτυπώνουμε κάτι μετράμε πόσα είναι τα γράμματά του συμπεριλαμβάνοντας και τα κενά. Επειδή εδώ είναι 18, γράφομε 18H. Πιο εύκολος τρόπος τον οποίο και θα χρησιμοποιούμε από εδώ και πέρα είναι να κλείνουμε την έκφραση σε ανεστραμμένες αγκύλες (') οπότε δεν χρειάζεται να μετράμε το μήκος της έκφρασης και να υποπέσομε σε κάποιο λάθος. Στο παράδειγμά μας είναι εντελώς ισοδύναμο το παρακάτω format:

```
2000 FORMAT(1X,'ogkos kybou (m^3)=',F10.4)
```

Σημειώνεται ότι στη Fortran **δεν** υπάρχει δυνατότητα απεικόνισης ελληνικών χαρακτήρων.

2.1.2.7 Χαρακτήρες ελέγχου

Πρέπει να δοθεί ιδιαίτερη προσοχή στους χαρακτήρες ελέγχου. Γενικά ο πρώτος χαρακτήρας κάθε εντολής format (όταν χρησιμοποιείται από εντολή WRITE και μόνον τότε) ελέγχει το που θα τυπωθούν τα αμέσως επόμενα. Εμείς θα χρησιμοποιούμε πάντα το 1X στην αρχή εννοώντας ότι θέλομε να εκτυπωθεί η εκάστοτε γραμμή **αμέσως** μετά την προηγούμενη.

Χρήσιμος είναι και ο χαρακτήρας slash (/) που χρησιμοποιείται για να αφήνεται μια κενή γραμμή. Θέλει όμως κάποια προσοχή διότι αν δεν έχει φτάσει στο τέλος της τρέχουσας γραμμής τότε απλά αρχίζει από την αρχή της επόμενης. Παραδείγματα:

```
1000 FORMAT(1X,I4,/,1X,I5,/,2X,I6)
b1234
b12345
b123456
```

βλέπουμε λοιπόν ένα παράδειγμα χρήσης απλού slash (/) στην εντολή format.Επειδή δεν έχει γεμίσει η κάθε γραμμή όταν συναντιέται το slash απλά αρχίζει να γράφει από την αρχή της επόμενης γραμμής.

```
2000 FORMAT(1X,I3,//,1X,I5,////,2X,I6)
b123
(κενή γραμμή)
b12345
```

```
(τρεις κενές γραμμές)
bb123456
```

Τύπος format	Συνοπτική περιγραφή
Ixx	ακέραιος αριθμός xx ψηφίων
Fxx.d	πραγματικός αριθμός xx ψηφίων με d δεκαδικά
nX	παρεμβολή n κενών στηλών
nH ή ‘’	εκτύπωση (μόνο) κειμένου

Ο παραπάνω πίνακας δείχνει συνοπτικά τους τύπους που χρησιμοποιούμε στην format.

2.2 Εντολές Εξόδου

2.2.1 Write (y,xxxx) A_1 , A_2 ,…,A_n

Η εντολή write είναι το αντίθετο της Read. Εμείς θα θεωρούμε ότι όλα τα δεδομένα εκτυπώνονται στον εκτυπωτή του συστήματος. Η σύνταξη της εντολής write είναι ανάλογη της σύνταξης της εντολής read. Διαφέρουν μόνο στη λειτουργία. Η πρώτη εκτυπώνει τα δεδομένα στον εκτυπωτή (έξοδος 6) ενώ η δεύτερη τα διαβάζει από κάρτες (είσοδος 5).
Ενώ λέγαμε READ(5,1000) τώρα θα γράφομε **πάντα** WRITE(6,1000) εννοώντας ότι στην γραμμή 1000 βρίσκεται η εντολή format που υπαγορεύει στον μεταφραστή τον τρόπο εκτύπωσης των μεταβλητών και συντάσσεται με τον τρόπο που αναλυτικά περιγράψαμε. Σημειώνομε ότι όπου A_1 , A_2 ,…,A_n μπορεί να είναι τόσο ακέραιες, όσο και πραγματικές μεταβλητές. Τώρα πλέον είμαστε σε θέση να γράψομε ένα απλό πρόγραμμα που θα διαβάζει κάποιες τιμές (δεν έχει σχέση ποιου τύπου) και θα τις εκτυπώνει.

```
      READ(5,1000) I,A,C
1000  FORMAT(1X,I5,3X,F10.3,1X,F8.2)
      WRITE(6,2000) I,A
2000  FORMAT(1X,I5,/,1X,F10.3)
      WRITE(6,2100) C
2100  FORMAT(1X,////,'H timh ths C einai=',F8.2)
      STOP
```

```
END
```

Αν τώρα υποτεθεί ότι I=12345 , A=988723.123 και C=00001.12=1.12, τότε θα εκτυπωθεί στην πρώτη γραμμή μετά από μια κενή στήλη η μεταβλητή I και στην επόμενη γραμμή, η μεταβλητή A. Τέλος μετά από 4 γραμμές (3 λόγω των slash και 1 λόγω της αλλαγής της εντολής format), θα εκτυπωθούν η έκφραση στις ανεστραμμένες παρενθέσεις και η τιμή C.

2.3 Αριθμητικαί Πράξεις

2.3.1 Απλές πράξεις

Οι απλές πράξεις της Fortran είναι η πρόσθεση (+) , η αφαίρεση (-) , ο πολλαπλασιασμός (*) , η διαίρεση (/) και η ύψωση σε δύναμη (**, λόγω ελλείψεως συμβόλων).Γίνονται μεταξύ **μεταβλητών του ιδίου τύπου**. Δεν επιτρέπεται να προσθέσομε σε μια ακέραια μεταβλητή μια πραγματική ή το αντίστροφο κτλ. Στον πίνακα I φαίνεται η χρήση των απλών πράξεων μεταξύ δυο πραγματικών μεταβλητών A και B.

Συμβολισμός πράξης	**Αποτέλεσμα**
A+B	Προσθέτει στο A το B
A-B	Αφαιρεί το B από το A
A*B	Πολλαπλασιάζει τα A,B
A/B	Διαιρεί το A με το B
A**B	Υψώνει το A στη B δύναμη

πίνακας I : απλές πράξεις

Ας υποθέσομε ότι έχομε πολλά είδη πράξεων σε μια έκφραση:

A+(B+C)/D-E*F+G**H

Τίθεται το θέμα με ποια σειρά θα εκτελεσθούν αυτές.Ο πίνακας II δίνει τη σειρά των πράξεων στην Fortran ο οποίος περιέχει και ορισμένες πράξεις που δεν έχομε ορίσει ακόμη. Σύμφωνα με τον πίνακα αυτό αρχικά θα εκτελεσθεί η παρένθεση, μετά η ύψωση σε δύναμη κατόπιν πολλαπλασιασμός και διαίρεση και τέλος πρόσθεση και αφαίρεση. Μάλιστα η εξίσωση σαρώνεται από αριστερά προς τα δεξιά.

Παραδείγματα:

1.+(3.+4.)/7.-5.*2.+1.**0 ⇒ 1.+7./7.-5.*2.+1.**0 ⇒ 1.+7./7.-5.*2.+1. ⇒
1.+1.-5.*2.+1. ⇒ 1.+1.-10.+1. ⇒ 2.-10.+1. ⇒ -8.+1. ⇒ -7.

((6+8)*2)+7*8*2 ⇒ (14*2)+7*8*2 ⇒ 28+7*8*2 ⇒ 28+56*2 ⇒ 28+112 ⇒ 140

Σειρά	**Πράξη**
1η	Παρενθέσεις
2η	Ύψωσις σε δύναμη
3η	Πολλαπλασιασμός και διαίρεση
4η	Πρόσθεση και Αφαίρεση
5η	Εκφράσεις Σύγκρισης (EQ,NE,LT,LE,GT,GE)
6η	Λογικός τελεστής .AND.
7η	.OR.

Πίνακας ΙΙ : ιεραρχία πράξεων

2.3.2 Μετατροπαί

Ενώ δεν μπορούμε να κάνομε πράξεις μεταξύ ακεραίων και πραγματικών μεταβλητών, είναι δυνατό να μετατρέψομε μια ακέραια μεταβλητή σε πραγματική και το αντίστροφο. Αυτό γίνεται πολύ απλά αν εξισώσομε μια ακέραια μεταβλητή με μια πραγματική:

A+I-5 : δεν έχει νόημα και προκαλεί πρόβλημα λάθους κατά την εκτέλεση
A=I+J (1) K=B+C (2) D=B+C (3)

Αν υποθέσω ότι I=1 και J=3 τότε μετά την εκτέλεση της πράξης (1) θα πάρω A=4**.**0 δηλαδή έναν πραγματικό αριθμό. Αν επίσης B=1.3 και C=1.4 τότε προφανώς η (3) θα δώσει D=2.7 δηλαδή όπως αναμέναμε έναν πραγματικό αριθμό. Η (2) όμως θα δώσει ακέραιο. ΠΡΟΣΟΧΗ δε θα στρογγυλοποιήσει τον αριθμό, θα του κόψει τα δεκαδικά ψηφία. Δεν θα πάρομε δηλαδή K=3 αλλά K=2 .Έτσι γίνεται η μετατροπή ακεραίου σε πραγματικό και το αντίστροφο.

2.3.3 Διαίρεση δυο ακεραίων

Αυτά που ειπώθηκαν στην παράγραφο 2.3.2 βρίσκουν άμεση εφαρμογή σε ένα πολύ απλό πρόβλημα: στη διαίρεση δυο ακεραίων αριθμών. Ας υποθέσομε ότι I=5 και M=2. Αναζητώ το αποτέλεσμα (το ακριβές) της διαίρεσής των. Μπορώ να κάνω τα ακόλουθα:

K=I/M (1) A=I/M (2) XI=I,XM=M και A=XI/XM (3)

Το πρώτο (1) θα δώσει προφανώς 2 καθότι αποκόπτεται το .5 από το 2.5 .Το δεύτερο (2) δε, θα μας δώσει 2**.** διότι το δεξιό μέρος της έκφρασης ισούται με 2 και άρα ο αντίστοιχος πραγματικός είναι το 2**.**0 .Στο τρίτο (3) όμως θα πάρομε XI=5**.** και XM=2**.** άρα θα διαιρέσω δυο πραγματικούς 5./2. και θα πάρω προφανώς A=2.5 δηλαδή το ζητούμενο. Αν αντί για A έβαζα στην τρίτη σχέση J τότε ναι μεν το δεξιό μέλος θα έδινε 2.5 αλλά το J θα έπαιρνε τιμή 2.

2.3.4 Ύψωση σε δύναμη

Σ' αυτό το σημείο θα πρέπει να γίνει μια ειδική μνεία δια την ύψωσιν ενός αριθμού (ακεραίου ή πραγματικού) σε μια δύναμη. Καταρχήν επαναλαμβάνομε ότι η έλλειψις ειδικών χαρακτήρων στη Fortran, οδήγησε στην παράσταση της ύψωσης κάποιας αριθμητικής τιμής σε δύναμη με τη χρήση δυο αστερίσκων που χρησιμοποιούνται δια την πράξη του πολλαπλασιασμού (**). Άρα γράφομε A**B εννοώντας ότι η πραγματική τιμή A υψώνεται σε μια πραγματική τιμή B. Δίδονται ορισμένα παραδείγματα:

A**2 , A**I , 3**.**2 , 3**.**I

Αυτά τα παραδείγματα δίνουν τις δυνατές χρήσεις της πράξεως όταν η βάση είναι πραγματικός. Πράγματι **ένας πραγματικός δύναται να υψωθεί σε μια ακέραια ή πραγματική δύναμη**. Δεν συμβαίνει όμως το ίδιο με τις ακέραιες βάσεις οι οποίες υψώνονται **μόνον σε ακέραιες δυνάμεις**. Έτσι:

2**3 , I**K είναι σωστά ενώ τα 2**A, 2**3**.** , I**A, I**3. είναι λανθασμένα.

2.4 Σύγκρισις

Στη συνέχεια θα αναπτυχθεί η διαδικασία της σύγκρισης καταρχήν δυο αριθμητικών τιμών και εν συνεχεία δυο ή περισσοτέρων λογικών εκφράσεων.

2.4.1 Σύγκριση αριθμητικών τιμών

Αυτή γίνεται με μια λογική έκφραση. Ας πάρομε ένα παράδειγμα. Διαβάζονται από κάρτες αριθμοί ακέραιοι. Θέλομε να σταματήσει η ανάγνωση καρτών μόλις συναντηθεί κάποιος ακέραιος με τιμή μεγαλύτερη του 18. Για να εκτελέσομε τη σύγκριση έχομε στη διάθεσή μας τις λογικές εκφράσεις του παρακάτω πίνακα.

Έκφραση	**Σημασία**
.EQ.	Ίσο (Equal)
.NE.	Άνισον (Not Equal)
.LT.	Μικρότερο Από (Lower Than)
.LE.	Μικρότερο ή ίσο (Lower or Equal)
.GT.	Μεγαλύτερο Από (Greater Than)
.GE.	Μεγαλύτερο ή ίσο (Greater or Equal)

Προφανώς η έκφραση που χρειάζεται είναι το "μεγαλύτερη" δηλαδή .GT. .Προσέξτε ότι κάθε λογική έκφραση αρχίζει και τελειώνει με μια τελεία (.).Για να εισάγομε μια λογική έκφραση σε ένα πρόγραμμα χρησιμοποιούμε την εντολή IF με την εξής σύνταξη:

IF <Λογική έκφρασις**> <**κάνε κάτι αν αληθεύει**>**

Στο παράδειγμά μας θέλομε αριθμούς μεγαλύτερους του 18. Οπότε πρέπει αμέσως μετά το διάβασμα των αριθμών να εισάγομε την εντολή:

IF (INT**.**GT**.**18) <να πάει στην εντολή που σταματάει το πρόγραμμα>

Μπορούμε να εισάγομε ότι επιθυμούμε τόσο στη θέση της λογικής έκφρασης όσο και στη θέση της ενέργειας που θα γίνει σε περίπτωση που αληθεύει η έκφραση. Προσοχή ιδιαίτερη θα πρέπει να δοθεί στο ότι **μπορούμε να συγκρίνομε αριθμητικές τιμές του ιδίου τύπου μόνον**. Αυτό με απλά λόγια σημαίνει ότι δεν μπορούμε να συγκρίνομε μια ακέραια τιμή με μια πραγματική. Αν υποθέσομε ότι στο παράδειγμά μας θέλαμε να σταματήσομε τη ροή του προγράμματος μόλις εμφανιστεί κάποια ακέραια τιμή μεγαλύτερη του 18, τότε θα γράφαμε:

IF (R**.**GT**.**18.0) < να πάει στην εντολή που σταματάει το πρόγραμμα>

Αν η έκφραση είναι αληθής (R>18) τότε εκτελείται η εντολή σε παρένθεση (<>) αλλιώς αγνοείται η γραμμή της εντολής IF και συνεχίζεται κανονικά η ροή του προγράμματος.

2.4.2 Σύγκριση δυο ή περισσοτέρων λογικών εκφράσεων

Τίθεται τώρα το θέμα της 2.4.1 ως εξής. Ναι μεν να διαβαστεί αριθμός από κάρτα και να σταματήσει το πρόγραμμα, αλλά να μην είναι μεγαλύτερος του 30. Αναφερόμεθα σε ακέραιους αριθμούς καταρχήν. Ένας ίσως όχι τόσο κομψός τρόπος θα ήταν ο εξής:

IF (INT.GT.30) <παρέκαμψε την επόμενη εντολή>
IF (INT.GT.18) <σταμάτησε την εκτέλεση>

Δηλαδή το πρώτο IF πιάνει όλους τους αριθμούς μεγαλύτερους από 30. Επομένως αν στο δεύτερο IF υπάρξει αριθμός μεγαλύτερος του 18 , τότε αυτός προφανώς θα είναι μικρότερος του 30. Ο τρόπος αυτός μειονεκτεί σε δυο τομείς . Πρώτον είναι εύκολο να γίνει λάθος καθώς όσο αυξάνεται η πολυπλοκότητα των εκφράσεων τόσο αυξάνει και η πιθανότητα λαθών και δεύτερον καταλαμβάνει πάνω από μια γραμμή οπότε και περισσότερη μνήμη.

Υπάρχουν λοιπόν στη fortran δυο λογικοί τελεσταί που αναλαμβάνουν να κάνουν τα εξής:

.AND. Μόνο αν και οι δυο εκφράσεις είναι αληθείς ισχύει η πρόταση

.OR. Η πρόταση είναι αληθής όταν μια από τις δυο (οποιαδήποτε) είναι αληθής ή και οι δυο φυσικά. Σύμφωνα με τα παραπάνω το ζεύγος των δυο εντολών IF αντικαθίσταται με δυο τρόπους:

```
IF (INT.GT.18.AND.INT.LT.30)  <σταμάτησε την εκτέλεση>
```

ή με την ισοδύναμη εντολή:

```
IF (INT.GT.30.OR.INT.LT.18) <παρέκαμψε την επόμενη εντολή>
STOP
```

Είδαμε λοιπόν δυο εντελώς ισοδύναμους τρόπους για να αντικαταστήσομε τα δυο IF. Δίνεται τώρα το εξής παράδειγμα. Διάβαζε ακέραιους αριθμούς και σταμάτα αν είναι αρνητικός και μεγαλύτερος του -30 ή αν είναι ίσος με 1560.Αρχίζομε τη λύση του θέματος από τη σύνταξη της λογικής έκφρασης. Τα "και" και "ή" κατά την εκφώνηση μας παραπέμπουν σε μια **ταυτόχρονη** χρησιμοποίηση των δυο λογικών τελεστών. Χρησιμοποιούμε τον πίνακα ΙΙ της 2.3.1 παραγράφου οπότε διαπιστώνομε ότι πρώτα θα εκτελεστεί η σύγκριση των εκφράσεων που συνδέονται με AND και κατόπιν αυτών που συνδέονται με OR.

Προφανώς χρειάζεται να ενώσω με OR τις δυο λογικές εκφράσεις που προκύπτουν. Πρώτον να είναι ο αριθμός ίσος με 1560 και δεύτερον να είναι και αρνητικός να και μεγαλύτερος του -30. Η πρώτη λογική έκφραση είναι (υποθέτοντας ότι ο αριθμός διαβάζεται και αποθηκεύεται στην μεταβλητή I) I.EQ.1560 και η δεύτερη I**.**LT**.**0.AND.I**.**GT.-30 . Οι δυο αυτές εκφράσεις - για να αληθεύει η πρότασή μας - είναι δυνατόν να αληθεύουν εναλλακτικά και συνδέονται συνεπώς με OR. Άρα το απλό πρόγραμμα αυτό θα μπορούσε να έχει την εξής μορφή:

```
   10 READ(5,1000) I
 1000 FORMAT(1X,I5)
      IF (I.EQ.1560.OR.I.LT.0.AND.I.GT.-30.) GOTO 30
      GOTO 10
   30 STOP
      END
```

Αξίζει να παρατηρήσουμε ότι πριν συγκριθούν οι εκφράσεις δεξιά και αριστερά του τελεστή OR θα γίνει η σύνθεση των υποεκφράσεων που συνδέονται με AND καθώς στην ιεραρχία των πράξεων **προηγείται πάντα το AND και έπεται το OR**.

Παράδειγμα: διαβάζονται αριθμοί από κάρτα (έστω ακέραιοι) . Αν αυτοί είναι άρτιοι να τους εκτυπώνει το πρόγραμμα , αλλιώς να συνεχίζει.

Το πρόβλημα μας είναι να βρούμε τη συνθήκη που μας δίνει έναν άρτιο αριθμό. Προφανώς ένας αριθμός είναι άρτιος αν διαιρείται με το 2. Όμως οποιοσδήποτε ακέραιος στη Fortran διαιρεθεί με το 2 θα δώσει ακέραιο αποτέλεσμα (πχ 3/2=1 μα και 2/2=1). Αν όμως διαιρέσω το 3/2=1 και το 3**./**2**.**=1.5 τότε λαμβάνω διαφορετικά αποτελέσματα. Άρα τρέπω τον ακέραιο σε πραγματικό και συγκρίνω το υπόλοιπα των δυο διαιρέσεων:

```
   10 READ(5,1000) I
 1000 FORMAT(1X,I5)
      XI=I
      IF (I/2.NE.XI/2.) GOTO 10
      WRITE(6,1000) I
      GOTO 10
```

Το πρόγραμμα δεν έχει τέλος. Τα προγράμματα σε Fortran έχουν την εξής ιδιόμορφη απαίτηση: πως με τις λιγότερες γραμμές εντολών να γίνουν οι απαιτούμενες ενέργειες. Αυτό καταρχήν συνέβαινε λόγω της ανάγκης διατρήσεως κάθε εντολής Fortran σε κάρτα, οπότε αν δυο εντολές μπορούσαν να συμπτυχθούν σε μια, τότε χρειαζόμασταν μια κάρτα λιγότερη. Το κέρδος από τη διαδικασία αυτή ήταν προφανώς τεράστιο σε οικονομία χώρου & χρόνου. Στο παράδειγμα αυτό, οι γραμμές 3 και 4 μπορούν να γραφούν ως μια εντελώς ισοδύναμη:

```
IF(I/2*2.NE.I) GOTO 10
```

Δηλαδή αν έχω το 3, δια 2 παίρνω 1 και επί 2 παίρνω 2 <> 3 άρα έχω περιττό. Έτσι καταρχήν γλιτώνουμε το στάδιο μετατροπής ακεραίου σε πραγματικό και εκτέλεσης διαίρεσης δυο πραγματικών (XI=I, XI/2**.**) .

Παράδειγμα: Από τα μαθηματικά, ένας αριθμός λέγεται *πρώτος αριθμός* αν διαιρείται μόνον με τον εαυτό του. Ζητείται να συνταχθεί πρόγραμμα που να διαβάζει έναν ακέραιο αριθμό και να ελέγχει αν είναι πρώτος. Αν το τελευταίο ισχύει, να τον εκτυπώνει.

Καταρχήν πρέπει να διαβάσομε τον αριθμό από την κάρτα με τις γνωστές μας εντολές (read, format) αποθηκεύοντάς τον σε μια ακέραια μεταβλητή πχ την M12X. Μπορούμε κατόπιν να αρχίσομε από το 2 ως την τιμή του M12X-1 και να διαιρούμε τον αριθμό. Αν το αποτέλεσμα κάποια στιγμή είναι ακέραιο, τότε αυτός δεν είναι πρώτος αριθμός. Δηλαδή αν το 7 το διαιρέσομε με τα 2,3,4,5,6 και δε βρούμε ακέραιο πηλίκο, τότε αυτός είναι πρώτος αριθμός. Αν και αυτή η σκέψη δεν είναι λάθος, είναι εξαιρετικά χρονοβόρα. Επομένως θα πρέπει να σκεφτούμε κάτι πιο έξυπνο. (Σκεφτείτε ο M12X να ήταν ο 108720, θα χρειάζονταν 108718 διαιρέσεις και συγκρίσεις).Ας πάρομε την τετραγωνική ρίζα του ακεραίου. Ισχυριζόμαστε ότι αρκεί να ελέγξομε αν διαιρείται ο αριθμός από το 2 ως την τετραγωνική του ρίζα. Δηλαδή αν έχω το 17, η τετραγωνική του ρίζα θα δώσει 4 (μην ξεχνάμε ότι δουλεύομε με ακέραιους). Αν το 2,3,4 δε διαιρούν το 17, τότε είναι πρώτος αριθμός. Άρα:

```
      READ (5,1000) M12X
 1000 FORMAT(1X,I10)
      R=M12X
      RIZA=R**.5
      NC=2
      XNC=2
   30 IF (M12X/NC.NE.R/XNC) GOTO 50
      WRITE(6,1000) M12X
      STOP
   50 NC=NC+1
      IF (NC.EQ.RIZA) STOP
      GOTO 30
      END
```

2.5 Εντολές μεταφοράς

2.5.1 GOTO xxxx

Πολλές φορές μέσα σ' ένα πρόγραμμα συμβαίνει να επιθυμούμε την αλλαγή της πορείας εκτέλεσής του. Για παράδειγμα είδαμε προηγούμενα ότι ανάλογα με την τιμή του ακεραίου ζητούσαμε την εκτύπωσή του ή τον τερματισμό του προγράμματος. Η αλλαγή της σειράς εκτέλεσης των εντολών γίνεται απλά με την εντολή GOTO η οποία μπορεί να βρίσκεται μόνη της σε μια γραμμή όταν σε κάθε περίπτωση θέλουμε να αλλάξει η ροή ή μετά από τη λογική έκφραση μιας εντολής IF. Όπου xxxx ο αριθμός της εντολής η οποία θα εκτελεστεί στη συνέχεια.

Εδώ όμως τίθεται το εξής πρόβλημα. Έχομε διαπιστώσει ότι η αρίθμηση των εντολών στη Fortran δεν είναι απαραίτητη εν γένει. Τότε όμως πώς θα γνωρίζει ο μεταφραστής της Fortran ποια εντολή εννοούμε ; Η απάντηση είναι απλή και σ' αυτή βοηθάει και ο παρακάτω πίνακας. Εκτός από τα στάνταρ είδη εντολών Fortran που αριθμούμε, βάζομε και αριθμό σε εντολές που καλούμε από κάποιο άλλο σημείο του προγράμματος.

Εντολές που χρειάζονται αρίθμηση
Κάθε εντολή *format* .
Κάθε εντολή που καλούμε από άλλο σημείο του προγράμματος.
Κάθε εντολή *continue.*

Η εντολή continue περιγράφεται αναλυτικά παρακάτω στους βρόχους DO.

Πρέπει όμως να δώσουμε συνοπτικά και ορισμένους κανόνες για την αρίθμηση των εντολών:

- κάθε εντολή μπορεί να αριθμηθεί από το 1 ως το 9999 χωρίς όμως να υπάρχουν δυο εντολές με τον ίδιο αριθμό.
- κάθε εντολή διατρείται στις στήλες 2,3,4 και 5.
- δεν έχει σχέση αν θα είναι με αύξουσα ή φθίνουσα σειρά αριθμημένες (όπως στη basic) . Οι εντολές εκτελούνται με τη σειρά που τις έχομε βάλει.

Παράδειγμα:

```
  10  READ(5,1000) I,J,K
1000  FORMAT(1X,I4,1X,I5,1X,I6)
      IF (I.EQ.9) GOTO 30
  40  GOTO 10
  30  WRITE (6,1000) I,J,K
      STOP
      END
```

Σειρά εκτέλεσης (10-1000-IF-40-10 -αν I=9 τότε 30,1000,STOP,END).

2.6 Μητρώα ή λίστες (Πίνακες)

2.6.0 Εισαγωγικά στοιχεία

Ας υποθέσομε ότι έχομε 100 αριθμούς διατρημένους σε κάρτες. Θέλουμε να τους διαβάσομε, να βρούμε το άθροισμά τους, το μέσο όρο τους και πόσοι από αυτούς είναι μεγαλύτεροι από το μέσο όρο. Ένας τρόπος αντιμετώπισης του αθροίσματος και του μέσου όρου είναι καθώς διαβάζονται οι αριθμοί να προστίθενται σε μια μεταβλητή (πχ τη SUM) οπότε μετά να διαιρεθούν με το 100 για να βρούμε το μέσο τους όρο. Μετά όμως θα έπρεπε να διαβάσουμε ξανά τους αριθμούς έτσι ώστε να τους συγκρίνομε με το μέσο όρο. Αυτό δεν θα ήταν αναγκαίο αν βρίσκαμε ένα τρόπο να τους αποθηκεύσομε στη μνήμη του Η/Υ. Ακριβώς την αποθήκευση αυτή εννοούμε με τον όρο λίστες ή πίνακες ή μητρώα που συμπίπτουν με τους γνωστούς από τη Γραμμική Άλγεβρα Πίνακες.

2.6.1 Μορφή Πινάκων

Από τη στιγμή που συνειδητοποιήσαμε την ανάγκη χρησιμοποίησης κάποιου πίνακα, πρέπει να κάνομε δυο πράγματα. Πρώτον να βρούμε τι διαστάσεως πίνακα εξυπηρετεί το σκοπό μας. Στο παράδειγμά μας θέλουμε πίνακα που να αποθηκεύει 100 αριθμούς , άρα πίνακα με 100 γραμμές. Εδώ τονίζεται ότι η Fortran και γενικά οι γλώσσες προγραμματισμού δουλεύουν με πίνακες στήλες και όχι με πίνακες γραμμές. Δηλαδή ο πίνακας του

παραδείγματός μας θα είναι υποχρεωτικά διαστάσεως 100x1 και όχι 1x100. Το δεύτερο είναι να δηλώσουμε αυτά που έχουμε υπόψη μας ,στο μεταφραστή μέσω της εντολής DIMENSION που περιγράφεται στην επόμενη παράγραφο.

2.6.2 DIMENSION A(xx,yy),B(zz,cc),...

Η εντολή αυτή δίνει στον Η/Υ τη διάσταση των πινάκων που θα χρησιμοποιηθούν. Μέσα στην παρένθεση εισάγονται πρώτα το πλήθος των σειρών και κατόπιν των στηλών. Ας υποθέσομε ότι θέλομε να δημιουργήσομε τρεις πίνακες A,B και C με τις εξής διαστάσεις: 100x1, 50x30 και 77x77. Είναι σαφές ότι θα δώσουμε την εξής εντολή:

```
DIMENSION A(100),B(50,30),C(77,77)
```

Παρατηρήσεις:

- Το όνομα κάθε πίνακα συμπεριφέρεται ακριβώς όπως μια μεταβλητή: μπορεί να είναι μέχρι 6 γράμματα αλλά το πρώτο οπωσδήποτε γράμμα και εννοείται να μην είναι το ίδιο με το όνομα κάποιας μεταβλητής του προγράμματος.

- Αν το όνομα ενός πίνακα αρχίζει από I,J,K,L,M ή N τότε στον πίνακα αυτό **θα μπορούν να αποθηκευτούν μόνο ακέραιες τιμές** στους δε υπόλοιπους μόνον πραγματικές. Συμπέρασμα: **δεν είναι δυνατόν σε έναν πίνακα να αποθηκεύσομε και ακέραιους και πραγματικούς αριθμούς.** Στο παράδειγμά μας στα A,B & C θα αποθηκευτούν προφανώς πραγματικοί αριθμοί.

- Αν προσπαθήσουμε να διαβάσουμε κάποιο στοιχείο πίνακα που οι δείκτες του ξεπερνούν τη μέγιστη διάσταση που δηλώθηκε στην εντολή dimension θα πάρομε μήνυμα λάθους και η εκτέλεση θα σταματήσει. Για παράδειγμα δεν μπορούμε να πούμε B(51,28) ή A(200).

- Ο δείκτης του πίνακα προφανώς δε μπορεί να είναι αρνητικός ή μηδέν. Καλό θα είναι κάθε φορά που χρησιμοποιούμε δείκτες να ελέγχομε την τιμή των.

2.6.3 Το παράδειγμά μας με πίνακες

Ας προσπαθήσομε τώρα να συντάξομε το πρόγραμμά μας με τη βοήθεια πινάκων. Πρέπει σαφώς να προσέξομε ότι η εντολή dimension πρέπει να χρησιμοποιηθεί **μια μόνο φορά στην αρχή του προγράμματος**. Έτσι έχομε:

```
      DIMENSION A(100)
      SUM=0.0                 Σ' αυτή τη μεταβλητή θα αποθηκευτεί το άθροισμα των.
      N=1                     Θα είναι ο δείκτης μας (N μέχρι 100).
   10 READ(5,1000) A(N)
 1000 FORMAT (1X,F10.4)       Γιατί ονομάσαμε Α τον πίνακα άρα έχομε πραγματικούς.
      SUM=SUM+A(N)
      IF (N.EQ.100) GOTO 20   Αν φτάσομε τα 100 να σταματήσει η ανάγνωση καρτών.
      N=N+1
      GOTO 10

   20 SUM=SUM/100.0           Υπολογισμός μέσου όρου.
      N=1
      M=0                     Αριθμοί μεγαλύτεροι από το μέσον όρο.
   30 IF (A(N).GT.SUM) M=M+1
      IF (N.EQ.100) GOTO 40
      N=N+1
      GOTO 30

   40 WRITE(6,2000) M
 2000 FORMAT(1X,'Plhthos arithmwn megalytervn apo meso oro=',I2)
      STOP
      END
```

Αυτό το πολύ απλό πρόγραμμα κάνει αυτά που περιγράφτηκαν παραπάνω. Αργότερα θα δούμε ότι με τη χρησιμοποίηση των βρόχων DO θα γίνει πολύ πιο απλό. Εν τω μεταξύ αξίζει να προσεχθεί ο έμμεσος τρόπος σχηματισμού βρόχων. Παίρνω έναν ακέραιο δείκτη πχ

τον Ν. Αρχίζω με τιμή Ν=1 και του προσθέτω σε κάθε επανάληψη τη μονάδα. Ελέγχω αν η τιμή του είναι ίση με το επιθυμητό πλήθος επαναλήψεων αλλιώς συνεχίζω.

2.6.4 Πολλαπλοί δείκται

Όπως ήδη έχομε δει, είναι δυνατή η δημιουργία πινάκων δύο διαστάσεων. Ο πρώτος δείκτης σ' αυτήν την περίπτωση θα μας δίνει τον αριθμό των γραμμών , ο δε δεύτερος τον αριθμό των στηλών. Έτσι το (2,1) θα αναφέρεται στη δεύτερη γραμμή πρώτη στήλη ενώ το (18,17) στη 18η γραμμή και 17η στήλη. Χρειάζεται διπλός έλεγχος εδώ, και για να μην υπερβούμε το μέγιστο αριθμό στηλών και για το μέγιστο αριθμό γραμμών όπως αυτοί δηλώνονται στην εντολή dimension. Οι πίνακες (λίστες) δυο διαστάσεων ακολουθούν όλους τους κανόνες και περιορισμούς στους οποίους υπόκεινται και οι μονοδιάστατοι πίνακες. Στο σημείο αυτό να τονίσομε ότι η χρησιμοποίηση διδιάστατων λιστών χρησιμοποιείται εφόσον έχομε πχ δυο μονοδιάστατες λίστες με αριθμούς του ιδίου τύπου καθόσον **σε μια διδιάστατη λίστα μπορούν να τοποθετηθούν μόνο αριθμοί του ιδίου τύπου (ακέραιοι ή πραγματικοί).**

2.7 Υπορουτίνες

2.7.0 Εισαγωγικά στοιχεία

Οι υπορουτίνες είναι διάφορα αυτοτελή υποπρογράμματα μέσα σε ένα μεγάλο πρόγραμμα που αναλαμβάνουν να εκτελέσουν μια συγκεκριμένη διαδικασία. Η ανάγκη χρησιμοποίησής των προέκυψε από τη δημιουργία μεγάλων προγραμμάτων, στα οποία ορισμένες διαδικασίες επαναλαμβάνονταν σε διάφορα σημεία τους (πχ η εκτύπωση ενός πραγματικού αριθμού). Εξάλλου γίνεται αρκετά πιο εύκολος ο εντοπισμός και η διόρθωση λαθών σε δέκα μικρά προγράμματα παρά σε ένα πολύ μεγάλο.

2.7.1 Η εντολή END

Την εντολή αυτή τη συναντήσαμε και παραπάνω σε ορισμένα προγράμματα - παραδείγματα που δώσαμε. Αυτή η εντολή **δεν** είναι εκτελέσιμη και απλά δηλώνει στον μεταφραστή της Fortran ότι δεν ακολουθούν άλλες εντολές. Έτσι δεν θα πρέπει να

συγχέεται με κανένα τρόπο με την συνονόματη εντολή της basic. Αφού δηλώνει το τέλος της εισαγωγής εντολών, θα πρέπει προφανώς να χρησιμοποιείται **μια** μόνον φορά σε κάθε πρόγραμμα και μάλιστα στο τέλος του. Επίσης καθόσον οι υπορουτίνες αποτελούν αυτοτελή υποπρογράμματα, πρέπει όταν τελειώνουν να χρησιμοποιείται η εντολή END. Έτσι την εντολή αυτή θα την χρησιμοποιούμε μόνο για να δηλώσομε το τέλος του κυρίως προγράμματος ή των υπορουτινών και μάλιστα στο τέλος τους.

2.7.2 Η εντολή STOP

Η εντολή αυτή δηλώνει το τέλος του προγράμματος αν και μπορεί να υπάρχουν εντολές που δεν εκτελέστηκαν ακόμη. Μπορούν να χρησιμοποιηθούν οσαδήποτε STOP μέσα σ ’ένα πρόγραμμα (κυρίως ή υπορουτίνα) έτσι ώστε ανάλογα με ορισμένες προϋποθέσεις να τερματιστεί η ροή του προγράμματος. Η εντολή STOP πρέπει **πάντα** να προηγείται της εντολής END.

2.7.3 Η εντολή SUBROUTINE <όνομα υπορουτίνας>

Αυτή η εντολή είναι η πρώτη κάθε υπορουτίνας και φέρει το όνομά της. Δεν είναι εκτελέσιμη εντολή, αλλά λέει στον compiler (μεταφραστής) ότι οι εντολές που ακολουθούν από το σημείο αυτό ως το πλησιέστερο END αποτελούν ένα αυτοτελές υποπρόγραμμα. Το όνομα της υπορουτίνας είναι όμοιο με ένα όνομα μεταβλητής, αλλά προφανώς δεν υπάρχει ο διαχωρισμός ακέραιοι-πραγματικοί που υπήρχε στις μεταβλητές. Έτσι ο πρώτος χαρακτήρας του ονόματος πρέπει να είναι γράμμα και οι υπόλοιποι είτε γράμματα είτε αριθμοί. Συνολικά όμως δεν πρέπει να υπερβαίνομε τους 6 χαρακτήρες. **Το όνομα της υπορουτίνας δε θα πρέπει να είναι όμοιο με το όνομα άλλης υπορουτίνας ή κάποιας μεταβλητής**. Για να δηλώσομε το τέλος μιας υπορουτίνας στο μεταφραστή χρησιμοποιούμε στο τέλος της την εντολή END μόνο μια φορά.

2.7.4 Η εντολή RETURN

Με την εντολή αυτή λέμε στον compiler να επιστρέψει στο κυρίως πρόγραμμα εγκαταλείποντας την υπορουτίνα. Προφανώς η εγκατάλειψη αυτή μπορεί να συμβαίνει σε

παραπάνω από ένα σημείο του υποπρογράμματος και άρα δύναται να χρησιμοποιηθούν παραπάνω των ένα RETURN. Αυτό που θα πρέπει να προσεχθεί μόνο είναι να "κλείνει" η υπορουτίνα με ένα return, καθότι το END δεν είναι εκτελέσιμη εντολή και έτσι το πρόγραμμα θα κολλήσει (εξαρτάται από την έκδοση του compiler). Αν εξάλλου επιθυμούμε τον τερματισμό του προγράμματος ενώ είμαστε στην υπορουτίνα θα χρησιμοποιήσομε την εντολή STOP.

2.7.5 Η εντολή CALL <όνομα υπορουτίνας>

Με την εντολή αυτή καλούμε μια υπορουτίνα. Στη θέση του ονόματος θέτομε το όνομα που έχομε χρησιμοποιήσει στην εντολή SUBROUTINE. Όταν συναντηθεί μέσα στην υπορουτίνα η εντολή RETURN τότε θα εκτελεστεί η εντολή του κυρίου προγράμματος που βρίσκεται αμέσως μετά την εντολή CALL.

Παράδειγμα:

```
CALL ROUT1
CALL ROUT2
CALL ROUT3
STOP
    END

SUBROUTINE ROUT1
  ……
RETURN
END
SUBROUTINE ROUT2
……
RETURN
END
SUBROUTINE ROUT3
  ……
RETURN
END
```

Συμπεραίνομε ότι καταρχήν καλείται η πρώτη ρουτίνα. Εκτελούνται όλες οι εντολές που συμπεριλαμβάνονται και κατόπιν μέσω του RETURN επιστρέφεται ο έλεγχος στο κύριο πρόγραμμα που καλεί τη δεύτερη ρουτίνα κοκ. Παρατηρείστε ότι η εντολή END χρησιμοποιήθηκε μετά την εντολή STOP του κυρίου προγράμματος και την εντολή RETURN των υπορουτινών.

2.7.6 Η εντολή COMMON $A_1, A_2,...,A_\eta$

Είπαμε ότι κάθε υπορουτίνα αποτελεί ένα ξεχωριστό πρόγραμμα. Πρέπει να χρησιμοποιήσομε την εντολή END για να δηλώσομε το τέλος της. Επομένως και κάθε μεταβλητή που θα χρησιμοποιήσομε θα είναι ανεξάρτητη από μια μεταβλητή του ιδίου ονόματος που χρησιμοποιήθηκε σε κάποια άλλη υπορουτίνα ή στο κυρίως πρόγραμμα. Δηλαδή ακόμα και αν εμείς διαβάσομε από κάρτα στο κυρίως πρόγραμμα την τιμή ενός ακεραίου και την αποθηκεύσομε στη μεταβλητή K, θα διαπιστώσομε ότι η μεταβλητή K "διατηρεί" την τιμή της μόνον στο κυρίως πρόγραμμα. Όμως αυτό δεν είναι ιδιαίτερα βολικό καθότι εμείς επιθυμούμε τις περισσότερες φορές τη διατήρηση της τιμής της μεταβλητής. Αυτό επιτυγχάνεται μέσω της εντολής COMMON.

Για να γίνει αυτό πιο σαφές θα εξετάσομε ένα απλό παράδειγμα που παρόλο που δεν δικαιολογεί τη χρησιμοποίηση υπορουτινών, για λόγους απλότητας θα της δημιουργήσομε. Ας υποθέσομε λοιπόν ότι θέλομε να συντάξομε ένα πρόγραμμα που σε μια υπορουτίνα να διαβάζει τις τιμές A και B δυο πραγματικών αριθμών από κάρτα, σε μια δεύτερη υπορουτίνα να υπολογίζει το άθροισμά των και να το αποθηκεύει στη μεταβλητή SUM, ενώ σε μια τρίτη υπορουτίνα να εκτυπώνεται το αποτέλεσμα. Σύμφωνα με αυτά που έχομε γνωρίσει θα γράφαμε:

```
CALL INP
CALL CALC
CALL OUT
STOP
END
SUBROUTINE INP
READ(5,1000) A,B
```

```
1000 FORMAT(1X,F10.4,1X,F10.4)
     RETURN
     END
     SUBROUTINE CALC
     C=A+B
     RETURN
     END
     SUBROUTINE OUT
     WRITE (6,2000) C
2000 FORMAT(1X,'Athroisma=',F15.4)
     RETURN
     END
```

Εκτελώντας όμως το πρόγραμμα θα διαπιστώσομε ότι το αποτέλεσμα είναι μηδέν για κάθε τιμή των A και B και αυτό γιατί οι μεταβλητές A και B δεν έχουν κανένα νόημα για την υπορουτίνα CALC και ακόμη η C δεν έχει ορισθεί στην υπορουτίνα OUT. Ο μηχανισμός της εντολής common λειτουργεί ως εξής: κρατάει κοινές θέσεις μνήμης για τις μεταβλητές που αναγράφονται σ' αυτήν και τις αποδίδει στις εκάστοτε μεταβλητές.

Αν στο κυρίως πρόγραμμα βάλομε COMMON A,B,C,K με A=5.0 , B=-7.99, C=9.1 και K=12, και σε μια οποιαδήποτε υπορουτίνα συναντήσομε COMMON D,G,E,L τότε θα έχομε D=5.0, G=-7.99, C=9.1 και L=12. Βλέπομε λοιπόν ότι δεν έχει σημασία ποια ονόματα θα χρησιμοποιήσομε στην εντολή common, αλλά μόνον οι θέσεις μνήμης στις οποίες θα αναφέρονται τα ονόματα αυτά. Κατά συνέπεια το παραπάνω παράδειγμα για να είναι σωστό θα έπρεπε να γραφτεί ως εξής:

```
     COMMON A,B,C
     CALL INP
     CALL CALC
     CALL OUT
     STOP
     END
     SUBROUTINE INP
     COMMON A,B,C
```

```
      READ(5,1000) A,B
1000  FORMAT(1X,F10.4,1X,F10.4)
      RETURN
      END
      SUBROUTINE CALC
      COMMON A,B,C
      C=A+B
      RETURN
      END
      COMMON A,B,C
      SUBROUTINE OUT
      WRITE (6,2000) C
2000  FORMAT(1X,'Athroisma=',F15.4)
      RETURN
      END
```

Παρατηρήσεις:

- Η εντολή COMMON μπαίνει αμέσως μετά την εντολή DIMENSION (αν υπάρχει) αλλιώς στην αρχή του κυρίως προγράμματος και μετά από κάθε εντολή SUBROUTINE.
- Δεν είναι ανάγκη να χρησιμοποιήσομε την εντολή COMMON στο κυρίως πρόγραμμα ή σε όλες τις υπορουτίνες αν δεν χρησιμοποιούνται εκεί οι μεταβλητές στις οποίες αναφερόμεθα. Δηλαδή στην προκειμένη περίπτωση δεν ήταν απαραίτητο να βάλομε το COMMON στο κυρίως πρόγραμμα καθώς εκεί δεν χρησιμοποιούνται οι μεταβλητές που αναγράφονται εις το COMMON.
- Σε περίπτωση πινάκων πχ A(50) μια ενδεχόμενη εντολή COMMON A θα δέσμευε 50 θέσεις μνήμης και η 32η θέση θα αντιστοιχούσε στο στοιχείο A με δείκτη 32.
- Συνίσταται να χρησιμοποιείται το ίδιο ακριβώς COMMON αν αυτό είναι δυνατό προς αποφυγή λαθών και επιπλοκών στο πρόγραμμα.

Παράδειγμα:

Να διαβαστεί πίνακας από υπορουτίνα και να εκτυπωθεί στο κυρίως πρόγραμμα (μονοδιάστατος με 30 ακέραιους αριθμούς)

```
      DIMENSION I(30)
```

```
      COMMON I
      CALL INP
      N=1
   20 WRITE(6,1000) I(N)
 1000 FORMAT(1X,I10)
      IF (N.EQ.30) GOTO 10
      N=N+1
              GOTO 20
   10 STOP
      END
      SUBROUTINE INP
      DIMENSION I(30)
      COMMON I
      N=1
   30 READ(5,500) I(N)
  500 FORMAT(5X,I10)
      IF (N.EQ.30) RETURN
      N=N+1
      GOTO 30
      END
```

Είναι προφανές ότι θα δεσμευτούν 30 κοινές θέσεις μνήμης κατά την εκτέλεση της εντολής COMMON γι' αυτό και αυτή ακολουθεί **υποχρεωτικά** την εντολή DIMENSION και δεν προηγείται αυτής.

2.8 Βρόχοι DO

2.8.1 Εισαγωγικά στοιχεία

Πολλές φορές χρειάστηκε να κάνομε την ίδια εργασία πολλές φορές. Για παράδειγμα στο προηγούμενο πρόγραμμα χρειαστήκαμε να εκτυπώσομε και τα 30 στοιχεία της λίστας. Αυτό το επιτύχαμε θέτοντας ένα δείκτη (το Ν στην προκειμένη περίπτωση) ίσο με την τιμή 1. Μετά ελέγχαμε αν το Ν είναι ίσο με τη μέγιστη τιμή (εδώ 30) αλλιώς προσθέταμε την μονάδα στο Ν και συνεχίζαμε τη διαδικασία. Αν και ίσως φαίνεται βολικός, στην

πραγματικότητα ο τρόπος αυτός είναι τουλάχιστον άβολος, οπότε τέθηκε το ερώτημα πώς αλλιώς μπορούσαμε να εκτελέσομε πολλές φορές την ίδια διαδικασία (βρόχος). Την απάντηση μας τη δίνει η θεωρία των βρόχων και μάλιστα των βρόχων DO (κάνε) όπως έχει σήμερα επικρατήσει να λέγονται από το πρώτο συνθετικό των.

2.8.2 Η εντολή DO xxxx A=m_1,m_2,m_3.

Αποτελεί την αρχή του βρόχου. Όπου xxxx μπαίνει ο αριθμός της εντολής CONTINUE που σημαίνει το τέλος του βρόχου. Τώρα το A έχει πλέον αντικαταστήσει αυτό που καλούσαμε δείκτη (βλ. πιο πάνω το αναφερόμενο ως N). Το m_1 αποτελεί μια αριθμητική τιμή που σημαίνει την αφετηρία μετρήσεως. Εμείς μέχρι τώρα - χωρίς αυτό να αποτελεί κανόνα - δίναμε N=1 δηλαδή m_1=1. Το m_2 είναι η τελευταία τιμή που θα λάβει ο δείκτης A. Σημειώνομε ότι στο προηγούμενο παράδειγμα της 2.8.1 το m_2 θα είχε την τιμή 30.Τέλος υπάρχει το m_3 το οποίο προστίθεται σε κάθε ανακύκλωση του βρόχου στο m_1 μέχρι αυτό να γίνει ίσο με το m_2 ή μέχρι να το ξεπεράσει. Αν δε παραλειφθεί τότε θα ισούται αυτόματα με τη μονάδα (τις περισσότερες φορές παραλείπεται χάριν απλότητας).

```
      DO 100 N=1,50,1
      <ΒΡΟΧΟΣ 50 ΕΠΑΝΑΛΗΨΕΩΝ>
100   CONTINUE
      DO 200 I=2,40
      <ΒΡΟΧΟΣ 39 ΕΠΑΝΑΛΗΨΕΩΝ>
200   CONTINUE
      DO 300 K=50,60,10
      <ΒΡΟΧΟΣ 1 ΕΠΑΝΑΛΗΨΗΣ>
300   CONTINUE
      DO 400 K=50,60,0
      <ΑΥΤΟΣ Ο ΒΡΟΧΟΣ ΔΕΝ ΕΧΕΙ ΤΕΛΟΣ>
400   CONTINUE
      DO 500 K=50,100,25
      <ΒΡΟΧΟΣ 2 ΕΠΑΝΑΛΗΨΕΩΝ>
500   CONTINUE
```

Το μόνο που έχομε να παρατηρήσομε είναι στο 4ο παράδειγμα ότι m_3 =0 οπότε ο βρόχος προφανώς δε θα έχει τέλος (infinite - ατέρμονος).

2.8.3 Η εντολή CONTINUE

Δεν είναι εκτελέσιμη αλλά απλώς δηλώνει το τέλος του βρόχου.

2.8.4 Υπερτοποθέτησις Βρόχων

Τις περισσότερες φορές τα προβλήματα που έχομε να επιλύσομε είναι αρκετά συνθετότερα από τα παραδείγματα που παρουσιάσαμε μέχρι τώρα και ήταν ιδιαίτερα απλά. Χρειαζόμαστε λοιπόν να διαβάσομε τα στοιχεία μιας διδιάστατης λίστας (αργότερα θα δούμε ότι αυτό είναι ακόμη ευκολότερα) από κάρτες. Πρόκειται μια ένα μητρώο που μπορεί να αποθηκεύει πραγματικούς αριθμούς. Αρκεί να τοποθετήσομε ένα βρόχο μέσα σ' έναν άλλο όπως:

```
      DO 100 N=1,100,1
      DO 200 K=1,100,1
      READ(5,1000) A(K,N)
1000 FORMAT(1X,F10.4)
 200 CONTINUE
 100 CONTINUE
```

Αυτό που μπορεί να φαίνεται απλό είναι εν τούτοις αρκετά πολύπλοκο. Καταρχήν να παρατηρήσομε ότι έτσι όπως δόθηκε το παράδειγμα, τα στοιχεία του πίνακα θα διαβάζονται κατά στήλες. Επίσης υποθέσαμε ότι το μητρώον είναι 100x100. Για να τοποθετήσομε ένα βρόχο μέσα σ' έναν άλλο πρέπει να γνωρίζομε τα εξής:

Είναι σφάλμα να φτιάξομε έναν κενό βρόχο ή να μεταβάλλομε τις τιμές που δηλώνονται στην εντολή DO.Τα ακόλουθα δηλαδή είναι λάθη:

```
      DO 100 N=1,50,1
100 CONTINUE
```

```
      DO 100 N=I,J,K
      J=J+1
      K=5
100   CONTINUE
```

Μπορούμε να βάλομε αντί για σταθερές τιμές μεταβλητές (κατάλληλες) στην εντολή DO χωρίς όμως όπως είπαμε να μεταβάλλονται καθ' οιονδήποτε τρόπο:

```
      I=5
      J=10
      K=2
      DO 100 N=I,J,K
      <ΒΡΟΧΟΣ 3 ΕΠΑΝΑΛΗΨΕΩΝ>
100   CONTINUE
```

Δεν επιτρέπεται όταν είμαστε μέσα σε ένα βρόχο να διαβάσομε πάλι την εντολή DO:

```
 50   DO 100 N=1,5,1
      ……
      GOTO 50
100   CONTINUE
```

Αυτό δηλαδή είναι ένα μεγάλο λάθος που γίνεται αρκετά συχνά.

Δεν επιτρέπεται να μπούμε σε ένα βρόχο χωρίς αρχικά να εκτελεστεί η εντολή DO, είναι όμως σωστό (και λίαν χρήσιμο) να βγαίνουμε από αυτόν πριν τελειώσουν όλες οι επαναλήψεις του καθ' οιονδήποτε τρόπο, παράδειγμα:

```
 50   ……
      DO 100 NC=1,20,1
      ……
      GOTO 50
      ……
100   CONTINUE
```

```
      DO 200 L=1,20
      ……
      GOTO 10
      ……
  200 CONTINUE
   10 WRITE(6,1000) I
 1000 FORMAT(1X,I4)
      STOP
      END
```

Τα δυο παραπάνω είναι **σωστά** παραδείγματα ενώ τα δυο επόμενα **λάθος**:

```
   10 GOTO 50
      DO 100 N=1,5
      ……
   50 WRITE(6,1000) I
 1000 FORMAT(1X,I4)
      ……
  100 CONTINUE
      STOP
      END


      ……
      DO 50 MAX=1,L,7
      ……
   30 READ(5,1000) I
 1000 FORMAT(1X,I4)
      ……
   50 CONTINUE
      GOTO 30
      STOP
      END
```

Αυτά τα δυο τελευταία παραδείγματα είναι σαφώς λανθασμένα καθότι εισέρχονται σε βρόχο χωρίς προηγουμένως να διαβαστεί η εντολή DO.

Δυο βρόχοι που υπερτοποθετούνται πρέπει **απαραίτητα να περιέχει ο ένας τον άλλο και οτιδήποτε άλλο είναι τραγικό λάθος**.

```
      DO 100 N=1,50
      DO 200 L=1,50
      ……
200 CONTINUE
      ……
100 CONTINUE
```

Σημείωση: προφανώς δεν είναι δυνατή η χρησιμοποίηση πχ του N ως δείκτη για τους δυο αυτούς βρόχους. Εξάλλου ότι έχει ειπωθεί για περίπτωση δυο βρόχων ισχύει ομοίως και για περισσότερους.

Αν μεταξύ των δυο CONTINUE δεν παρεμβάλλεται εκτελέσιμη εντολή(ες) τότε **είναι δυνατόν να χρησιμοποιηθεί το ίδιο CONTINUE και για τους δυο (ή περισσότερους) βρόχους θεωρώντας ότι ο ένας περιέχει τον άλλο**.

```
      DO 100 I=1,50,1
      ……
      DO 200 J=1,20,2
      ……
      DO 200 M=1,10,2
      ……
200 CONTINUE
100 CONTINUE
```

Θα αποτελούσε όμως **μεγάλο λάθος** μας να πούμε:

```
      DO 200 N=1,7
      DO 300 L=7,9,1
      ……
```

```
200 CONTINUE
300 CONTINUE
    STOP
    END
```

Ή να χρησιμοποιήσομε ίδιους δείκτες πχ:

```
    DO 200 I=1,5
    DO 200 I=5,7,1
    ……
200 CONTINUE
```

Περισσότερες πρακτικές πληροφορίες μπορούν να βρεθούν στις ασκήσεις του κεφ. 3.

2.9 Προχωρημένα Θέματα

2.9.1 Εισαγωγή

Για λόγους πληρότητας του παρόντος συγγράμματος πρέπει να γίνει μια αναφορά σε πιο προχωρημένες δυνατότητες της Fortran. Θα ασχοληθούμε με δυο τελευταία θέματα. Το πρώτο είναι η μεταφορά με όρους και το δεύτερο ο έλεγχος του εκτυπωτή για εμφάνιση κομψότερων αποτελεσμάτων. Εξάλλου η Fortran δε φημίζεται για τις εκτυπωτικές της δυνατότητες. Ο αναγνώστης θα βρει εφαρμογές για τα περιγραφόμενα θέματα στην παράγραφο 2.9, στο κεφάλαιο 4, και ειδικότερα στη Β ενότητά του.

2.9.2 GOTO (xx,yy,zz,…) I

Η εντολή αυτή είναι σαν ένας συνδυασμός τόσων εντολών IF-GOTO όσοι οι αριθμοί εντολών εντός της παρένθεσης. Δηλαδή αντί να πούμε:

```
    IF (I.EQ.1) GOTO 100
    IF (I.EQ.2) GOTO 200
```

```
      IF (I.EQ.3) GOTO 300
      STOP
  100 ……
      STOP
  200 ……
      STOP
  300 ……
      STOP
      END
```

Χρησιμοποιώντας την εντολή αυτή έχομε:

```
      GOTO (100,200,300) I
```

όπου βέβαια το I ως ακέραιος παίρνει τιμές από 1 ως 3. Αν πχ I=1 τότε θα είναι ισοδύναμη η εντολή αυτή με την GOTO 100. Χρειάζεται όμως προσοχή στα εξής:

- Αν το I ξεπερνάει το 3 στην προκειμένη περίπτωση, δυο μπορούν να συμβούν, ή να αγνοηθεί εντελώς η εντολή ή να βγει μήνυμα λάθους (εξαρτάται από τον compiler).
- Προφανώς πρέπει να εξετάζομε ακέραιο αριθμό πχ I,K και όχι πραγματικό πχ A ή B.

Καλό είναι να ελέγχεται ο δείκτης I πριν εκτελεσθεί η εντολή.

```
      READ(5,1000) I
 1000 FORMAT(1X,I1)
      IF (I.LT.0.OR.I.GT.4) GOTO 40
      GOTO (100,200,300,400) I
   40 STOP
  100 ……
      STOP
  400 ……
      STOP
      END
```

2.9.3 Ο πρώτος χαρακτήρας της έκφρασης FORMAT

Στην παράγραφο 2.1.2.7 είπαμε κάπως αυθαίρετα να χρησιμοποιούμε το 1Χ στην αρχή κάθε format διότι ο πρώτος χαρακτήρας δεν εκτυπώνεται αλλά καθορίζει που θα εκτυπωθούν οι υπόλοιποι. Στον παρακάτω πίνακα δίδονται όλοι οι χαρακτήρες ελέγχου του εκτυπωτή.

Χαρακτήρας	**Λειτουργία**
vX	Η επόμενη γραμμή θα τυπωθεί ακριβώς κάτω από αυτή που τυπώθηκε. Έτσι για v=1 προκύπτει το 1Χ που είπαμε στην αρχή.
0 (μηδέν)	Θα προηγηθεί μια κενή γραμμή αυτής που εκτυπώνεται. Παρατηρήστε την ομοιότητα με το slash που περιγράψαμε.
1	Η νέα γραμμή θα τοποθετηθεί σε καινούργια σελίδα. Ιδιαίτερα χρήσιμο στην αρχή της εκτύπωσης ή σε μεγάλες εκτυπώσεις.

Προφανώς για τους χαρακτήρες ελέγχου θα χρησιμοποιηθεί το H format δηλαδή:

```
1000 FORMAT(1HO,'Kenh grammh',I10)
```

θα τυπώσει τον ακέραιο αφήνοντας μια κενή γραμμή από αυτήν που εκτυπώθηκε πριν.

```
1200 FORMAT(1H1,'Nea selida',I10)
```

ενώ εδώ θα αλλάξει σελίδα και θα εκτυπώσει τον ακέραιο με δέκα ψηφία. Για το δε κενό, αυτό εισάγεται αμέσως χωρίς τη χρήση του H format δηλαδή έχομε:

```
1500 FORMAT(1X,'Apla allagh grammhs',I10)
```

οπότε απλά αλλάζει ο εκτυπωτής γραμμή.

ΚΕΦΑΛΑΙΟ 3: Παραδείγματα

Άσκηση 3.1

Η μεταβλητή Χ μπορεί να πάρει τις τιμές 1,2 και 3.Αν όταν Χ=3 έχω Υ=0 και όταν Χ=1 ή 2 έχω Υ=3 να γραφεί το αντίστοιχο πρόγραμμα σε Fortran 77.

Λύση:

Καταρχήν το πρόγραμμά μας πρέπει με κάποιο τρόπο να "διαβάζει" τον αριθμό Χ. Έτσι χρησιμοποιούμε την εντολή READ και την αντίστοιχη FORMAT και διαβάζουμε τον αριθμό Χ, τον οποίο θεωρούμε πραγματικό:

```
      READ(5,1000) X
1000 FORMAT(1X,F5.2)
```

Στην συνέχεια συγκρίνουμε τον αριθμό Χ διαδοχικά με τα 1. , 2. , 3. και για κάθε επιμέρους σύγκριση που κάνουμε, θέτουμε την μεταβλητή Υ ίση με την τιμή που υποδεικνύει η εκφώνηση της άσκησης.

```
      IF (X.EQ.3.) Y=0.
      IF (X.EQ.1.) Y=3.
      IF (X.EQ.2.) Y=3.
```

Τέλος, αφού αντιμετωπίσαμε με επιτυχία το πρόβλημα πρέπει να εκτυπώσουμε την τιμή του Υ. Αυτό γίνεται με την εντολή:

```
      WRITE(6,1000) Y
```

Παρατηρείστε ότι χρησιμοποιούμε την ίδια FORMAT που χρησιμοποιήσαμε στην αρχή του προγράμματός μας, ανεξάρτητα αν αυτή έχει ήδη χρησιμοποιηθεί ή βρίσκεται σε

άλλο σημείο του προγράμματός μας. Τέλος χρησιμοποιούμε τις εντολές STOP,END για να κλείσουμε το πρόγραμμά μας.

Συνολικά το πρόγραμμά μας έχει ως εξής:

```
      READ(5,1000) X
1000  FORMAT(1X,F5.2)
      IF (X.EQ.3.) Y=0.
      IF (X.EQ.1.) Y=3.
      IF (X.EQ.2.) Y=3.
      WRITE(6,1000) Y
      STOP
      END
```

Άσκηση 3.2

Να γραφεί το πρόγραμμα που δίνει για τη μεταβλητή X τις εξής τιμές: X=Y όταν Y<1, X=Y^2 όταν Y>1 και X=0 όταν Y=1.

Λύση:

Καταρχήν το πρόγραμμά μας πρέπει να διαβάζει την μεταβλητή Y. Αυτό επιτυγχάνεται με τις εντολές:

```
      READ(5,1000) Y
1000  FORMAT(1X,F5.2)
```

Προφανώς θεωρούμε ότι η Y είναι πραγματικός, οπότε στην αντίστοιχη FORMAT χρησιμοποιούμε την μορφοποίηση F5.2

Στην συνέχεια προχωρούμε στον έλεγχο της τιμής της Y ,ώστε να διακλαδίσουμε το πρόγραμμά μας και να αποδώσουμε κάθε φορά την σωστή τιμή στην μεταβλητή X, σύμφωνα με τις υποδείξεις της ασκήσεως. Έτσι γράφουμε διαδοχικά "φίλτρα" σύγκρισης

(εντολές IF) και σε κάθε μία τοποθετούμε την αντίστοιχη καταχώρηση τις τιμής στη μεταβλητή X.

```
      IF (Y.EQ.1.) X=0.
```

Η παραπάνω εντολή σημαίνει:" Εάν το Y είναι ίσο με 1. (Μην ξεχνάτε την τελεία, ο Y είναι πραγματικός) τότε το X παίρνει την τιμή 0. ".Τι συμβαίνει εάν το Y δεν είναι 1. ;Πολύ απλά , η εκτέλεση του προγράμματος συνεχίζεται στην επόμενη γραμμή, που είναι και το επόμενο "φίλτρο".

```
      IF (Y.LT.1.) X=Y
```

Η λειτουργία είναι όμοια με αυτή που περιγράφηκε παραπάνω. Τέλος για να καλύψουμε όλες τις περιπτώσεις για το Y, σύμφωνα με τις υποδείξεις της άσκησης, γράφομε:

```
      IF (Y.GT.1.) X=Y**2.
```

Μετά και από αυτή την εντολή IF έχουμε καλύψει όλες τις περιπτώσεις για το Y. Τέλος προχωρούμε στην εκτύπωση της μεταβλητής X:

```
      WRITE(5,1000) X
```

και κλείνουμε το πρόγραμμά μας:

```
      STOP
      END
```

Συνολικά το πρόγραμμά μας έχει ως εξής:

```
      READ(5,1000) Y
 1000 FORMAT(1X,F5.2)
      IF (Y.EQ.1.) X=0.
      IF (Y.LT.1.) X=Y
      IF (Y.GT.1.) X=Y**2.
```

```
      WRITE(5,1000) X
      STOP
      END
```

Παρατηρείστε ότι στην έκφραση X=Y**2. η τελεία δεν είναι απαραίτητη ,αφού πρόκειται για περίπτωση όπου μπορούμε να αναμίξουμε πραγματικούς και ακέραιους.(Η ύψωση πραγματικού σε ακέραια δύναμη επιτρέπεται.)

Άσκηση 3.3

Δίνονται δύο μονοδιάστατα μητρώα Α,Β με 50 στοιχεία το καθένα. Φτιάξτε ένα πρόγραμμα το οποίο να δημιουργεί ένα νέο μητρώο Q με 50 στοιχεία, του οποίου το κάθε στοιχείο θα είναι το άθροισμα των αντίστοιχων στοιχείων των Α,Β.

Λύση:

Καταρχάς θα πρέπει να δημιουργήσουμε το μητρώο (πίνακας) Q στον οποίο αναφέρεται το πρόβλημα. Θεωρούμε ότι τα μητρώα Α,Β υπάρχουν ήδη και έχουν συμπληρωθεί με δεδομένα. Απομένει λοιπόν να φτιάξομε το τμήμα εκείνο του προγράμματος στο οποίο θα αποδίδουμε στα στοιχεία του Q τις κατάλληλες τιμές. Προφανώς αυτό θα γίνει με έναν βρόχο 50 επαναλήψεων, στον οποίο το στοιχείο J του πίνακα Α θα προστίθεται με το στοιχείο J του πίνακα Β , και το άθροισμα θα καταχωρείται στο στοιχείο J του πίνακα Q.
Καταρχήν δημιουργούμε τον πίνακα Q:

```
      DIMENSION Q(50)
```

Στη συνέχεια γράφουμε τον βρόχο επανάληψης κατά τα γνωστά:

```
      DO 100 J=1,50,1
      Q(J)=A(J)+B(J)
  100 CONTINUE
```

Έτσι αντιμετωπίσαμε το πρόβλημα. Αν θεωρήσουμε ότι τα μητρώα Α,Β δεν υπήρχαν, θα πρέπει αφενός να δημιουργήσουμε τα μητρώα Α,Β και στην συνέχεια να εισάγουμε τις τιμές στα στοιχεία τους, με την βοήθεια βρόχου επανάληψης. Είναι δυνατόν να

χρησιμοποιήσουμε πάλι το βρόχο J που περιγράφηκε παραπάνω. Τέλος είναι δυνατόν μέσα στον ίδιο βρόχο να εκτελούμε και την εκτύπωση του στοιχείου Q(J),χρησιμοποιώντας την κατάλληλη FORMAT.

Έτσι το τελικό πρόγραμμα θα έχει ως εξής:

```
      DIMENSION A(50),B(50),Q(50)
      DO 100 J=1,50,1
      READ(5,1000) A(J),B(J)
 1000 FORMAT(F10.4,2X,F10.4)
      Q(J)=A(J)+B(J)
      WRITE(6,2000) Q(J)
 2000 FORMAT(F14.4)
  100 CONTINUE
      STOP
      END
```

Άσκηση 3.4

Δίνεται το μητρώο AR(20,100).Γράψτε ένα πρόγραμμα το οποίο θα υπολογίζει το άθροισμα όλων των στοιχείων του AR.

Λύση:

Καταρχάς θα δημιουργήσουμε το μητρώο AR(20,1000).Αυτό γίνεται με την εντολή DIMENSION κατά τα γνωστά:

```
      DIMENSION AR(20,100)
```

Στη συνέχεια επιλέγουμε μια μεταβλητή (έστω S),με την οποία θα υπολογίσουμε το συνολικό άθροισμα. Η μεταβλητή αυτή θα αυξάνεται συνεχώς καθώς ένα - ένα τα στοιχεία του πίνακα AR θα αθροίζονται στην προηγούμενη του τιμή. Προφανώς θα θέσουμε **(εκτός του βρόχου επανάληψης)** την μεταβλητή S ίση με μηδέν, ενώ μέσα στο βρόχο η S θα ισούται με τον εαυτό της συν το στοιχείο που διαβάστηκε στον συγκεκριμένο κύκλο επανάληψης. Κατά την πρώτη επανάληψη ,η S θα πάρει την τιμή μηδέν συν την τιμή του πρώτου επεξεργαζόμενου στοιχείου (εδώ το AR(1,1))και θα ισούται με AR(1,1). Κατά την

δεύτερη επανάληψη η S θα πάρει την τιμή AR(1,1) συν την τιμή AR(1,2) κοκ. Τελικά, μετά το πέρας των επαναλήψεων , η S θα ισούται με το ζητούμενο άθροισμα. Επειδή το μητρώο AR είναι δύο διαστάσεων, η ανάγνωση των στοιχείων θα γίνει με διπλό βρόχο επανάληψης **(μην ξεχνάμε ότι ο ένας θα πρέπει να περιέχει πλήρως τον άλλο)** .Με τον πρώτο βρόχο θα αυξάνουμε τον μετρητή της πρώτης διάστασης του AR ενώ με τον άλλο βρόχο θα αυξάνουμε τον μετρητή της δεύτερης διάστασης, ώστε τελικά να "σαρώσουμε" όλα τα στοιχεία του AR.

Για την εισαγωγή των στοιχείων του AR μπορούμε να χρησιμοποιήσουμε τους ίδιους δύο βρόχους που περιγράφηκαν παραπάνω. Πρέπει να προσέξουμε στη σειρά της εισαγωγής των στοιχείων του AR: πρώτα συμπληρώνεται η "γραμμή" 1 από το 1 έως το 120 με αύξουσα σειρά, στην συνέχεια εισάγουμε την γραμμή 2 κοκ. Τέλος πρέπει να προσέξουμε ώστε η εισαγωγή των στοιχείων να προηγείται του υπολογισμού.

Κατά τον μηδενισμό της S δεν πρέπει να ξεχνάμε την τελεία μετά το μηδέν. Η μεταβλητή αυτή είναι πραγματική, διότι το μητρώο AR περιέχει πραγματικά στοιχεία (μην ξεχνάμε ότι ένα μητρώο περιέχει ακεραίους εάν το όνομά του αρχίζει από I.J,K,L,M,N,. Σε αντίθετη περίπτωση περιέχει πραγματικούς),οπότε και το άθροισμα αυτών θα είναι πραγματικός.

Τελικά το πρόγραμμά μας έχει ως εξής:

```
      DIMENSION AR(20,100)
      S=0.
      DO 200 N=1,20,1
      DO 100 L=1,100,1
C     ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΤΟΥΣ
      READ(5,1000) AR(N,L)
 1000 FORMAT(F10.4)
      S=S+AR(N,L)
  100 CONTINUE
  200 CONTINUE
C     ΕΚΤΥΠΩΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
      WRITE(6,2000) S
 2000 FORMAT(F14.4)
```

```
      STOP
      END
```

Άσκηση 3.5

Γράψτε ένα πρόγραμμα που να βρίσκει και να εκτυπώνει το μέσο όρο N αριθμών και τη διαφορά κάθε αριθμού από το μέσο όρο. Οι αριθμοί έχουν διατρηθεί 5 ανά κάρτα με format 5F8.3.Δίνεται N<=1000.

Λύση:

Η λύση του προβλήματός μας έχει απλή συλλογιστική: θα πρέπει να αντιμετωπίσουμε τον "ειδικό" τρόπο με τον οποίο θα δοθούν τα δεδομένα , σύμφωνα με την άσκηση, και στην συνέχεια να προχωρήσουμε στην καθαυτό λύση του προβλήματος.

Τα στοιχεία που θα χρειαστούμε για το πρώτο τμήμα της άσκησης είναι το άθροισμα των αριθμών - δεδομένων καθώς και το πλήθος τους.

Δημιουργώ έναν πίνακα μονοδιάστατο με ικανή διάσταση στον οποίο αποθηκεύονται οι αριθμοί C έτσι ώστε αφενός να μη χρειαστεί να ξαναχρησιμοποιηθούν οι κάρτες, αφετέρου να μπορώ να χειριστώ καλύτερα τα δεδομένα:

```
      DIMENSION AN(1000)
```

Στη συνέχεια τοποθετώ τα στοιχεία στον πίνακα AN χρησιμοποιώντας έναν μετρητή (N) του οποίου δίνω αρχική τιμή μηδέν και τον οποίο αυξάνω κατά 5 κάθε φορά που διαβάζεται μια κάρτα. Το στοιχείο A είναι το πρώτο που διαβάζεται κάθε φορά και σε αυτό αντιστοιχούμε το AN(N +1).Το B είναι το δεύτερο στοιχείο κάθε κάρτας και το αποδίδουμε στο στοιχείο AN(N + 2) κοκ. Φτάνουμε στο E στοιχείο που είναι ο πέμπτος κατά σειρά αριθμός που είναι διατρημένος στην κάρτα και τον καταχωρούμε στο στοιχείο AN(N + 5) του βοηθητικού πίνακά μας. Στη συνέχεια αυξάνουμε το N κατά 5.Είναι λοιπόν φανερό ότι το στοιχείο A της δεύτερης κάρτας θα καταχωρηθεί στο AN(5+1) = AN(6) κοκ. και έτσι λειτουργεί η είσοδος των δεδομένων.

Ταυτόχρονα με την εισαγωγή των δεδομένων, κι αφού ήδη δημιουργήσαμε βρόχο που να διαβάζει όλα τα στοιχεία είναι βολικό να αντιμετωπίσουμε την μια πτυχή του προβλήματος: το άθροισμα των αριθμών. Έτσι χρησιμοποιούμε την πραγματική μεταβλητή SUM την οποία αρχικά μηδενίζουμε (προφανώς εκτός του βρόχου επανάληψης).Μέσα στο βρόχο επανάληψης θέτουμε την SUM ίση με τον εαυτό της (για να κρατήσουμε το προηγούμενο άθροισμα) συν τα στοιχεία που διαβάστηκαν στην συγκεκριμένη κάρτα δεδομένων (και συνεπώς στην συγκεκριμένη επανάληψη του βρόχου).

Για να καταλάβουμε ότι τελείωσαν τα δεδομένα απαιτούμε μια κάρτα να έχει διατρημένους και τους 5 αριθμούς ίσους με το μηδέν. Με την βοήθεια λοιπόν του ελέγχου:

```
      IF (A.EQ.0..AND.B.EQ.0..AND.C.EQ.0..AND.D.EQ.0..AND.
     *E.EQ.0.) GOTO 200
```

γίνεται η έξοδος από τον βρόχο επανάληψης, με τον οποίο γίνεται η εισαγωγή των δεδομένων. Ο έλεγχος αυτός θα πρέπει να γίνεται στην αρχή ώστε να μην επεξεργαστούμε τα μηδενικά ως δεδομένα και μετά να καταλάβουμε την σημασία τους! Μην ξεχνάτε την τελεία: οι αριθμοί είναι πραγματικοί!

Έχουμε λοιπόν:

```
      N=0
      SUM=0
      DO 100 NC=1,1000,5
      READ(5,1000) A,B,C,D,E
1000  FORMAT(5F8.3)
      IF(A.EQ.0..AND.B.EQ.0..AND.C.EQ.0..AND.D.EQ.0..
     *AND.E.EQ.0.)GOTO 200
      SUM=SUM+A+B+C+D+E
      AN(N+1)=A
      AN(N+2)=B
      AN(N+3)=C
      AN(N+4)=D
      AN(N+5)=E
      N=N+5
```

```
100 CONTINUE
```

Η έξοδος από το βρόχο επανάληψης μας οδηγεί στην γραμμή με αριθμό 200.Έχοντας ήδη αντιμετωπίσει την μία πτυχή του προβλήματος, αντιμετωπίζουμε πολύ εύκολα την άλλη: το πλήθος των αριθμών είναι ίσο με την μεταβλητή N που χρησιμοποιήσαμε στο βρόχο επανάληψης!

Το μόνο που μας μένει είναι να διαιρέσουμε άθροισμα με πλήθος για να βρούμε τον μέσο όρο. Προσοχή! το N είναι ακέραιος και απλή διαίρεση πραγματικού με ακέραιο θα οδηγήσει σε λάθος. Πρώτα μετατρέπουμε τον ακέραιο N σε πραγματικό XN:

```
200 XN=N
```

Τώρα είμαστε έτοιμοι για την διαίρεση. Για να μην δημιουργούμε επιπλέον μεταβλητές χωρίς λόγο μπορούμε να χρησιμοποιήσουμε την SUM ξανά, χωρίς να ξεχνάμε ότι τώρα το SUM δεν σημαίνει άθροισμα αλλά μέσο όρο:

```
  SUM=SUM/XN
```

Προχωράμε στην εκτύπωση του μέσου όρου SUM:

```
  WRITE(6,1500) SUM
1500 FORMAT(1H0,' Mesos Oros = ',F10.3,/)
```

Προχωρώντας στο δεύτερο τμήμα της άσκησης, χρειαζόμαστε τον **μέσο όρο** SUM που βρήκαμε προηγουμένως καθώς και **κάθε ένα από τα στοιχεία δεδομένα** του πίνακα AN. Έτσι υπολογίζουμε με ένα βρόχο N επαναλήψεων την διαφορά του στοιχείου AN(JC) από τον μέσο όρο SUM.Ταυτόχρονα προχωράμε στην εκτύπωση της διαφοράς αυτής.(Σε αντίθετη περίπτωση θα έπρεπε να δημιουργήσουμε πίνακα στον οποίο θα κρατούσαμε τις διαφορές, και στην συνέχεια να προχωρούσαμε με νέο βρόχο στην εκτύπωση των στοιχείων του νέου πίνακα)

```
  DO 300 JC=1,N,1
  AA=SUM-AN(JC)
  WRITE(6,2000) AA
```

```
2000 FORMAT(1X,F9.3)
 300 CONTINUE
```

Τέλος, κλείνουμε το πρόγραμμά μας:

```
     STOP
     END
```

Συνολικά το πρόγραμμά μας έχει ως εξής:

```
     DIMENSION AN(1000)
     N=0
     SUM=0
     DO 100 NC=1,1000,5
     READ(5,1000) A,B,C,D,E
1000 FORMAT(5F8.3)
     IF(A.EQ.0..AND.B.EQ.0..AND.C.EQ.0..AND.D.
    *EQ.0..AND.E.EQ.0.)GOTO 200
     SUM=SUM+A+B+C+D+E
     AN(N+1)=A
     AN(N+2)=B
     AN(N+3)=C
     AN(N+4)=D
     AN(N+5)=E
     N=N+5
 100 CONTINUE
 200 XN=N
     SUM=SUM/XN
     WRITE(6,1500) SUM
1500 FORMAT(1H0,' Mesos Oros = ',F10.3,/)
     DO 300 JC=1,N,1
     AA=SUM-AN(JC)
     WRITE(6,2000) AA
2000 FORMAT(1X,F9.3)
```

```
300 CONTINUE
    STOP
    END
```

Άσκηση 3.6

Δίνεται μια μονοδιάστατη λίστα Υ με 32 στοιχεία και ζητείται να υπολογίσετε την ακόλουθη έκφραση TRAP=1/2(Y_1 +2Y_2 +2Y_3 ...+2Y_{31} +Y_{32})

Λύση:

Αν και είναι δοθείσα η μονοδιάστατη λίστα Υ, είναι προτιμότερο να γράψουμε ένα μικρό τμήμα κώδικα όπου θα καταχωρούμε τις τιμές των στοιχείων της λίστας Υ. Καταρχήν δημιουργούμε τον πίνακα Υ:

```
    DIMENSION Y(32)
```

Στην συνέχεια προχωράμε στην εισαγωγή των δεδομένων:

```
    DO 100 LC=1,32,1
    READ(5,1000) Y(LC)
100 CONTINUE
```

Τώρα που έχουμε τα δεδομένα στον πίνακα Υ, μπορούμε να τα επεξεργαστούμε. Για να διευκολυνθούμε θα προσθέσουμε τα στοιχεία από το Υ(2) έως το Υ(31) που έχουμε τον ίδιο συντελεστή (2) και στην συνέχεια στο άθροισμα αυτό θα προσαρτήσουμε τα στοιχεία Υ(1),Υ(32).Κάνουμε λοιπόν έναν βρόχο επανάληψης με μετρητή NC που μεταβάλλεται από 2 έως 31 και χρησιμοποιώντας την ίδια μεταβλητή TRAP που θα παρουσιάσουμε στο τέλος, υπολογίζουμε το πρώτο άθροισμα:

```
    TRAP=0.
    DO 200 NC=2,31,1
    TRAP=TRAP+2*Y(NC)
200 CONTINUE
```

Στη συνέχεια προσθέτουμε το Y(1) και το Y(32):

```
      TRAP=TRAP+Y(1)+Y(32)
```

και τέλος διαιρούμε το τελικό άθροισμα με το 2 , εκτυπώνουμε το αποτέλεσμα και κλείνουμε το πρόγραμμά μας:

```
      TRAP=TRAP/2.
      WRITE (6,1500) TRAP
 1000 FORMAT (F8.3)
 1500 FORMAT (1H0,1X,F10.3)
      STOP
      END
```

Συνολικά:

```
      DIMENSION Y(32)
      DO 100 LC=1,32,1
      READ(5,1000) Y(LC)
  100 CONTINUE
      TRAP=0.
      DO 200 NC=2,31,1
      TRAP=TRAP+2*Y(NC)
  200 CONTINUE
      TRAP=TRAP+Y(1)+Y(32)
      TRAP=TRAP/2.
      WRITE (6,1500) TRAP
 1000 FORMAT (F8.3)
 1500 FORMAT (1H0,1X,F10.3)
      STOP
      END
```

Άσκηση 3.7

Εάν έχομε μια μονοδιάστατη λίστα τιμών τότε οι πρώτες διαφορές της λίστας σχηματίζονται αφαιρώντας κάθε στοιχείο εκτός από το τελευταίο από το στοιχείο αμέσως μετά από αυτό. Με την περίπτωση μιας μονοδιάστατης λίστας με 50 στοιχεία να υπολογίσετε μιας άλλης D(x) με D(x)=X(i+1)-X(i), όπου i=1,2,...,49.

Λύση:

Καταρχήν δημιουργούμε τους δύο πίνακες του προβλήματος και στην συνέχεια προχωρούμε στην εισαγωγή των δεδομένων της λίστας X, με την βοήθεια βρόχου 50 επαναλήψεων:

```
      DIMENSION D(49),X(50)
      DO 100 NC=1,50,1
      READ(5,1000) X(NC)
1000  FORMAT(F8.3)
 100  CONTINUE
```

Στη συνέχεια προχωρούμε στην λύση του προβλήματος. Με την βοήθεια βρόχου 49 επαναλήψεων καταχωρούμε στο στοιχείο D(LC) την τιμή X(LC+1)-X(LC) .Ταυτόχρονα μπορούμε να εκτυπώνουμε τα στοιχεία του πίνακα D, για να μην αναγκαστούμε να χρησιμοποιήσουμε νέο βρόχο. Μέσα στο βρόχο των 49 επαναλήψεων πρέπει να προσέξουμε ώστε ο υπολογισμός του στοιχείου D(LC) να προηγείται της εκτύπωσής του! Μπορούμε να ξαναχρησιμοποιήσουμε την FORMAT της γραμμής 1000:

```
      DO 200 LC=1,49,1
      D(LC)=X(LC+1)-X(LC)
      WRITE(6,1000) D(LC)
200   CONTINUE
```

Τέλος δεν έχουμε παρά να κλείσουμε το πρόγραμμά μας:

```
      STOP
      END
```

Τελικά:

```
      DIMENSION D(49),X(50)
      DO 100 NC=1,50,1
      READ(5,1000) X(NC)
 1000 FORMAT(F8.3)
  100 CONTINUE
      DO 200 LC=1,49,1
      D(LC)=X(LC+1)-X(LC)
      WRITE(6,1000) D(LC)
  200 CONTINUE
      STOP
      END
```

Άσκηση 3.8

Γράψτε πρόγραμμα που να τυπώνει τιμές του Ν! όπου Ν από 1 μέχρι 20.

Λύση:

Είναι φανερό ότι για την λύση αυτού του προβλήματος θα χρησιμοποιήσουμε κάποια τεχνική παρόμοια με αυτή που κάναμε π.χ. κατά τον υπολογισμό του αθροίσματος των στοιχείων ενός πίνακα. Τότε χρησιμοποιούσαμε μια βοηθητική μεταβλητή τύπου SUM την οποία μηδενίζαμε εκτός του βρόχου και μέσα στο βρόχο την θέταμε ίση με τον εαυτό της συν το στοιχείο που επεξεργαζόμασταν στην συγκεκριμένη επανάληψη του βρόχου. Μετά την "σάρωση" όλων των στοιχείων η μεταβλητή SUM ήταν ίση με το ζητούμενο άθροισμα. Θα μετατρέψουμε την παραπάνω μέθοδο για να έχουμε γινόμενο αριθμών, αφού το Ν! είναι γινόμενο αριθμών.

Σ' αυτή την περίπτωση δεν έχουμε πίνακα, **αφού ο ίδιος ο μετρητής του βρόχου μπορεί να χρησιμοποιηθεί πολύ εύκολα για την λύση του προβλήματος.(πρόκειται για πρόβλημα με ακεραίους)** Έτσι θεσπίζουμε μια βοηθητική μεταβλητή ΑΝ την οποία **θέτουμε ίση με ένα και όχι ίση με μηδέν** (διότι η μονάδα είναι το ουδέτερο στοιχείο του

πολλαπλασιασμού) εκτός του βρόχου και μέσα στο βρόχο θέτουμε την ΑΝ ίση με τον εαυτό της **επί** το στοιχείο που επεξεργαζόμαστε. Προσοχή πρέπει να δώσουμε ώστε να θεσπίσουμε τα σωστά όρια στο μετρητή και να μην οδηγηθούμε σε λάθος. Με κατάλληλο λοιπόν βρόχο μπορούμε να υπολογίσουμε το Ν!

Σημαντικό πρόβλημα είναι **το μέγεθος των αποτελεσμάτων**. Τι σημαίνει αυτό; Απλά, παρόλο που γνωρίζουμε ότι το αποτέλεσμα είναι ακέραιος ,δεν μπορούμε να το χειριστούμε ως ακέραιο διότι είναι πολύ μεγάλο. Έτσι είμαστε αναγκασμένοι να το χειριστούμε ως πραγματικό, αφού τα όρια για τους πραγματικούς είναι πολύ ευρύτερα από αυτά των ακεραίων.

Τέλος, εφόσον τα αποτελέσματα **δεν** καταχωρούνται σε πίνακα, είμαστε υποχρεωμένοι να προχωρήσουμε σε εκτύπωση των ενδιάμεσων παραγοντικών με την βοήθεια της WRITE **μέσα** στον βρόχο (αλλιώς θα χάνονται τα ενδιάμεσα δεδομένα καθώς εκτελείται ο βρόχος). Σε αντίθετη περίπτωση, θα έπρεπε να ορίσουμε πίνακα με πραγματικά στοιχεία και διάσταση 20,να καταχωρούμε τα ενδιάμεσα αποτελέσματα σ' αυτόν κατά την εκτέλεση του βρόχου και μετά την συμπλήρωση των επαναλήψεων, να ορίσουμε νέο βρόχο 20 επαναλήψεων όπου και θα προχωρούσαμε στην εκτύπωση των αποτελεσμάτων. Είναι, θα λέγαμε, επιβεβλημένη η χρησιμοποίηση του πρώτου τρόπου. Κατά τα γνωστά πρέπει η εκτύπωση των αποτελεσμάτων να έπεται του υπολογισμού τους! Σύμφωνα με τα παραπάνω έχουμε:

```
C     ΑΡΧΙΚΟΠΟΙΣΗ ΜΕΤΑΒΛΗΤΩΝ
      AN=1.
C     ARXH BROXOY
      DO 100 NC=1,20,1
C     ΜΕΤΑΤΡΟΠΗ ΤΟΥ ΜΕΤΡΗΤΗ ΣΕ ΠΡΑΓΜΑΤΙΚΟ
      ANC=NC
      AN=AN*ANC
C     ΤΑΥΤΟΧΡΟΝΗ ΕΚΤΥΠΩΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
      WRITE (6,1000) AN
 1000 FORMAT(2X,F20.0)
  100 CONTINUE
      STOP
      END
```

Άσκηση 3.9

Υπολογίστε το άθροισμα των 20 πρώτων όρων των σειρών:

i) 1+1/2+1/3+....+1/20

ii) 1-1/2+1/3-....-1/20

iii) 1-1/2*3+1/3*4-....-1/20*21

iv) 1-1/2!+1/3!-1/4!+...-1/20!

Λύση:

Θα λύσουμε όλα τα σκέλη της άσκησης σε ένα πρόγραμμα. Αφού έχουμε αθροίσματα ,και θα χρησιμοποιήσουμε οπωσδήποτε βρόχους επανάληψης, θέτουμε τις γνωστές βοηθητικές μεταβλητές ίσες με μηδέν:

```
      SUM1=0.
      SUM2=0.
      SUM3=0.
      SUM4=0.
```

Για την πρώτη σειρά, είναι σαφές ότι θα δημιουργούμε μέσα στο βρόχο επανάληψης τα επιμέρους κλάσματα, τα οποία θα προσθέτουμε συνεχώς στην βοηθητική μεταβλητή SUM1.Μετά το τέλος του βρόχου η SUM1 θα είναι το ζητούμενο άθροισμα. Προσοχή θα πρέπει να δίνουμε στο γεγονός ότι σε όλα τα σκέλη της άσκησης οι πράξεις πρέπει να γίνονται με πραγματικούς αριθμούς, και συνεπώς θα πρέπει να μετατρέπουμε τον (ακέραιο) μετρητή σε πραγματικό. Για την πρώτη σειρά θα έχω:

```
C     Πρώτη σειρά
      DO 100 NC=1,20,1
      XNC=NC
      XNC=1./XNC
      SUM1=SUM1+XNC
  100 CONTINUE
```

Είναι καλό να εκτελούμε τις πρώτες π.χ. δύο επαναλήψεις του βρόχου με το μυαλό μας, για να αποφύγουμε εύκολα λάθη. Εδώ ο πρώτος όρος της σειράς (δηλαδή το 1) προκύπτει για NC=1.

Για την δεύτερη σειρά είναι προφανές ότι θα προκύψει από την πρώτη σειρά εάν συμπεριλάβουμε έναν όρο της μορφής (-1)**<κατάλληλος εκθέτης>.
Έτσι:

```
C     Δεύτερη Σειρά.
      DO 200 NC=1,20,1
      XNC=NC
      XNC=1./XNC
      SUM2=SUM2+XNC*(-1)**(NC+1)
  200 CONTINUE
```

Ένα εύκολο λάθος θα ήταν να γράψουμε (-1)**(NC).Παρατηρούμε όμως ότι ο πρώτος όρος είναι θετικός και προκύπτει για NC=1,και έτσι θέτουμε τον εκθέτη ως NC+1. Παρατηρήστε ότι δεν έχουμε πρόβλημα εμπλοκής ακεραίων - πραγματικών, αφού επιτρέπεται η ύψωση πραγματικού σε ακέραια δύναμη.

Για την τρίτη σειρά παρατηρώ ότι το πρώτο στοιχείο (1) δεν προκύπτει κανονικά (σύμφωνα με τα υπόλοιπα).Έτσι θέτω το SUM3 από 0. σε 1. και ξεκινώ την επεξεργασία της σειράς από τον δεύτερο όρο. Μ' αυτό τον τρόπο θα έχω σωστά αποτελέσματα. Έχουμε:

```
C     Τρίτη Σειρά.
      SUM3=1.
      DO 300 LC=2,20,1
      XLC=LC*(LC+1)
      XLC=1./XLC
      SUM3=SUM3+XLC*(-1)**(LC+1)
  300 CONTINUE
```

Για την τέταρτη σειρά δεν έχω πρόβλημα όπως αυτό που παρουσιάστηκε στην σειρά 3,διότι μπορώ να θεωρήσω ότι 1 = 1 / 1!Με διαδικασία που περιγράφτηκε αναλυτικότατα σε προηγούμενο πρόβλημα υπολογίζουμε τα παραγοντικά και τελικά χρησιμοποιούμε την ίδια διαδικασία για να υπολογίσουμε το άθροισμα. Έτσι έχω:

```
C     Τέταρτη Σειρά.
      JC=1
      DO 400 MC=1,20,1
      JC=JC*MC
      XJC=JC
      XJC=1./XJC
      SUM4=SUM4+XJC*(-1)**(MC+1)
  400 CONTINUE
```

Τελικά, αφού διορθώσουμε το SUM3=0. σε SUM3=1. και γράψουμε την ρουτίνα εκτύπωσης των αποτελεσμάτων, συντάσσουμε το τελικό μας πρόγραμμα που έχει ως εξής:

```
      SUM1=0.
      SUM2=0.
      SUM3=1.
      SUM4=0.
C     Πρώτη σειρά
      DO 100 NC=1,20,1
      XNC=NC
      XNC=1./XNC
      SUM1=SUM1+XNC
  100 CONTINUE
C     Δεύτερη Σειρά.
      DO 200 NC=1,20,1
      XNC=NC
      XNC=1./XNC
      SUM2=SUM2+XNC*(-1)**(NC+1)
  200 CONTINUE
C     Τρίτη Σειρά.
```

```
      SUM3=1.
      DO 300 LC=2,20,1
      XLC=LC*(LC+1)
      XLC=1./XLC
      SUM3=SUM3+XLC*(-1)**(LC+1)
  300 CONTINUE
C     Τέταρτη Σειρά.
      JC=1
      DO 400 MC=1,20,1
      JC=JC*MC
      XJC=JC
      XJC=1./XJC
      SUM4=SUM4+XJC*(-1)**(MC+1)
  400 CONTINUE
C     Αποτελέσματα Και Εκτυπώσεις.
      WRITE(6,1000) SUM1,SUM2,SUM3,SUM4
 1000 FORMAT(1H0,' Πρώτη Σειρά   =',F10.3,/,' Δεύτερη Σειρά
     *=',F10.3,/,' Τρίτη Σειρά   =',F10.3,/,' Τέταρτη Σειρά
     *=',F10.3,/)
      STOP
      END
```

Άσκηση 3.10

Στα πλαίσια ερευνητικού προγράμματος για τον έλεγχο της αντοχής σε θλίψη μιας κατασκευής από οπλισμένο σκυρόδεμα, θραύστηκαν κυβικά δοκίμια σκυροδέματος ακμής 15 cm.Η επεξεργασία των πειραματικών αποτελεσμάτων αποφασίστηκε να γίνει με τη βοήθεια Η/Υ και έτσι τα στοιχεία και αποτελέσματα των μετρήσεων σε κάθε πείραμα, διατρήθηκαν σε κάρτες με την ακόλουθη μορφή:

Στήλη 1: Για κύβο από ελαφροσκυρόδεμα έχει διατρηθεί ένα 1
Για κύβο από σκυρόδεμα έχει διατρηθεί ένα 2

Στήλες 2 - 5: Κενές

Στήλες 6 - 10: Η ηλικία κάθε κύβου έχει καταχωρηθεί εδώ σαν ακέραιος θετικός αριθμός.

Στήλες 11 - 15: Κενές
Στήλες 16 - 20:Το βάρος κάθε κύβου έχει διατρηθεί εδώ ως πραγματικός αριθμός κιλών με ακρίβεια τριών δεκαδικών ψηφίων.
Στήλες 21 - 25: Κενές
Στήλες 26 - 30: Το φορτίο θλίψεως του κύβου έχει διατρηθεί εδώ σαν πραγματικός αριθμός με ακρίβεια δυο δεκαδικών ψηφίων.

Γράψτε ένα πρόγραμμα που να διαβάζει τρεις κάρτες δεδομένων θέτοντας τα είδη του σκυροδέματος στα ICONA,ICONB,ICONC, τις ηλικίες στα IAGEA,IAGEB,IAGEC, τα βάρη στα WEITA,WEITB,WEITC και τα φορτία θλίψεως στα COMA,COMB,COMC.Δημιουργείστε 3 κάρτες δεδομένων σύμφωνα με την δεδομένη FORMAT και τυπώστε τις παραπάνω μεταβλητές ως εξής:

ICONA ICONB ICONC
COMA
WEITA
IAGEA

Λύση:

Πρέπει να φτιάξουμε ένα πρόγραμμα ,με το οποίο θα γίνεται η εισαγωγή των παραπάνω δεδομένων από 3 κάρτες και στην συνέχεια να προχωρήσουμε στην εκτύπωση των δεδομένων κατά το υπόδειγμα. Η μόνη δυσκολία κατά την εισαγωγή των δεδομένων είναι ότι πρέπει να προσαρμόσουμε την FORMAT της εντολής READ **ακριβώς** στον τρόπο που έχουν διατρηθεί τα δεδομένα στις κάρτες.

Έτσι η FORMAT μας έχει σαν πρώτο στοιχείο το I1,κι αυτό γιατί στην στήλη 1 διαβάζεται ένας ακέραιος που χαρακτηρίζει το σκυρόδεμα. Στην συνέχεια έχουμε **4 κενά** (και όχι τρία: από το 2 έως το 5 συμπεριλαμβάνοντας τα άκρα έχουμε 4 κενά) και συνεπώς η FORMAT μας θα συνεχίζεται με το στοιχείο 4X. Στις στήλες 6 - 10 αντιστοιχούμε το I5 διότι η ηλικία δίνεται ως ακέραιος, σύμφωνα με την άσκηση.

Στις κενές στήλες 11 - 15 αντιστοιχούμε το 5X.Οι στήλες 16 - 20 είναι πραγματικοί με ακρίβεια τριών δεκαδικών και συνεπώς θα χρησιμοποιήσουμε την μορφοποίηση **[F(συν. αριθμός θέσεων).(αριθμός δεκαδικών θέσεων)].** Εδώ έχουμε συνολικό αριθμό θέσεων

5 εκ των οποίων 3 θέσεις είναι δεκαδικές, και συνεπώς η κατάλληλη μορφοποίηση είναι η (F5.3) .Οι στήλες 21 - 25 είναι κενές και συνεπώς συνεχίζουμε την FORMAT με 5X.Τέλος ,όμοια με τις στήλες 16 - 20,γράφουμε και για τις στήλες 26 - 30 το (F5.2).Έτσι είμαστε έτοιμοι για την εισαγωγή των δεδομένων. Επειδή ο αριθμός των καρτών είναι μικρός είναι προτιμότερο να μην χρησιμοποιήσουμε πίνακες και βρόχους επανάληψης, και να προχωρήσουμε απευθείας στην καταχώρηση των τιμών στις μεταβλητές:

```
      READ(5,1000) ICONA,IAGEA,WEITA,COMA
      READ(5,1000) ICONB,IAGEB,WEITB,COMB
      READ(5,1000) ICONC,IAGEC,WEITC,COMC
1000 FORMAT(I1,4X,I5,5X,F5.3,5X,F5.2)
```

Μετά την εισαγωγή των δεδομένων και με την βοήθεια νέων εντολών FORMAT εκτυπώνουμε τα δεδομένα σύμφωνα με τις υποδείξεις της άσκησης:

```
      WRITE(6,2000) ICONA,ICONB,ICONC
2000 FORMAT(6X,I1,5X,I1,5X,I1)
      WRITE(6,2100) IAGEA,IAGEB,IAGEC
2100 FORMAT(2X,I5,1X,I5,1X,I5)
      WRITE(6,2200) WEITA,WEITB,WEITC
2200 FORMAT(2X,F5.3,1X,F5.3,1X,F5.3)
      WRITE(6,2300) COMA,COMB,COMC
2300 FORMAT(2X,F5.2,1X,F5.2,1X,F5.2)
```

Για κάθε ένα είδος μεταβλητής (π.χ. για τις ηλικίες των δοκιμίων) χρησιμοποιούμε την ίδια μορφοποίηση με αυτή που χρησιμοποιήσαμε κατά την εισαγωγή των δεδομένων. Τέλος κλείνουμε το πρόγραμμά μας. Συνολικά:

```
      READ(5,1000) ICONA,IAGEA,WEITA,COMA
      READ(5,1000) ICONB,IAGEB,WEITB,COMB
      READ(5,1000) ICONC,IAGEC,WEITC,COMC
1000 FORMAT(I1,4X,I5,5X,F5.3,5X,F5.2)
      WRITE(6,2000) ICONA,ICONB,ICONC
2000 FORMAT(6X,I1,5X,I1,5X,I1)
      WRITE(6,2100) IAGEA,IAGEB,IAGEC
```

```
2100 FORMAT(2X,I5,1X,I5,1X,I5)
     WRITE(6,2200) WEITA,WEITB,WEITC
2200 FORMAT(2X,F5.3,1X,F5.3,1X,F5.3)
     WRITE(6,2300) COMA,COMB,COMC
2300 FORMAT(2X,F5.2,1X,F5.2,1X,F5.2)
     STOP
     END
```

Άσκηση 3.11

Υπολογίστε το μέσο φορτίο θλίψεως, τη μέση ηλικία, το μέσο βάρος των δοκιμίων με ακρίβεια δύο δεκαδικών ψηφίων.

Λύση:

Η εργασία αυτή είναι συνέχεια της προηγούμενης. Έτσι δεν χρειάζεται να ξαναγράψουμε την ρουτίνα εισαγωγής δεδομένων, αλλά απλά πρέπει να προσαρτήσουμε την ρουτίνα υπολογισμού και εκτύπωσης των μέσων τιμών. Η ρουτίνα αυτή θα διαιρεί το άθροισμα των αντιστοίχων τιμών των πειραμάτων με το 3. ,και δεν παρουσιάζει καμία δυσκολία, εκτός από το ότι πρέπει να μετατρέψουμε τις ακέραιες τιμές (π.χ. τις ηλικίες των δοκιμίων) σε πραγματικές πριν προχωρήσουμε στην διαίρεση. Έτσι έχω:

```
     XIAGEA=IAGEA
     XIAGEB=IAGEB
     XIAGEC=IAGEC
     COMM=(COMA+COMB+COMC)/3.
     AGEM=(XIAGEA+XIAGEB+XIAGEC)/3.
     WEITM=(WEITA+WEITB+WEITC)/3.
     WRITE(6,3000) COMM,AGEM,WEITM
3000 FORMAT(1X,F6.2,1X,F8.2,1X,F6.2)
```

Παρατηρείστε ότι όλα τα αποτελέσματα είναι πραγματικοί και εκτυπώνονται με δύο δεκαδικά ψηφία, σύμφωνα με τις υποδείξεις της άσκησης. Το παραπάνω τμήμα προγράμματος θα το προσαρτήσουμε στο τέλος του προηγούμενου και πριν τις εντολές STOP - END.Έτσι:

```
      READ(5,1000) ICONA,IAGEA,WEITA,COMA
      READ(5,1000) ICONB,IAGEB,WEITB,COMB
      READ(5,1000) ICONC,IAGEC,WEITC,COMC
 1000 FORMAT(I1,4X,I5,5X,F5.3,5X,F5.2)
      WRITE(6,2000) ICONA,ICONB,ICONC
 2000 FORMAT(6X,I1,5X,I1,5X,I1)
      WRITE(6,2100) IAGEA,IAGEB,IAGEC
 2100 FORMAT(2X,I5,1X,I5,1X,I5)
      WRITE(6,2200) WEITA,WEITB,WEITC
 2200 FORMAT(2X,F5.3,1X,F5.3,1X,F5.3)
      WRITE(6,2300) COMA,COMB,COMC
 2300 FORMAT(2X,F5.2,1X,F5.2,1X,F5.2)
C     Υπολογισμός Μέσων Τιμών
      XIAGEA=IAGEA
      XIAGEB=IAGEB
      XIAGEC=IAGEC
      COMM=(COMA+COMB+COMC)/3.
      AGEM=(XIAGEA+XIAGEB+XIAGEC)/3.
      WEITM=(WEITA+WEITB+WEITC)/3.
      WRITE(6,3000) COMM,AGEM,WEITM
 3000 FORMAT(1X,F6.2,1X,F8.2,1X,F6.2)
      STOP
      END
```

Άσκηση 3.12

Στην στατιστική μελέτη χρειάζεται να βρεθούν και να εκτυπωθούν τα ακόλουθα:

- Αριθμός κύβων από ελαφροσκυρόδεμα.
- Αριθμός κύβων από κοινό σκυρόδεμα.
- Μέσο βάρος, ηλικία, φορτίο θλίψεως των κύβων
- Αριθμός κύβων με φορτίο θλίψεως παραπάνω από 50 tn.

Λύση:

Η εργασία αυτή αποτελεί επέκταση των προηγουμένων, με την διαφορά ότι δεν έχουμε τρεις αλλά άγνωστο αριθμό καρτών και επίσης έχουμε το ερώτημα 4 που δεν αντιμετωπίστηκε στις προηγούμενες εργασίες. Είναι προφανές ότι ο άγνωστος αριθμός των καρτών μας περιορίζει σε χρησιμοποίηση βρόχων επανάληψης. Επειδή δεν μπορούμε να μάθουμε **εξαρχής** τον αριθμό των καρτών, δεν μπορούμε να χρησιμοποιήσουμε βρόχο επανάληψης της μορφής DO.

Έτσι θα φτιάξουμε έναν βρόχο με απλές εντολές αλλαγής ροής, ο οποίος **θα επαναλαμβάνει τον εαυτό του έως ότου διαβαστεί μια ειδική κάρτα, η οποία θα σημαίνει και το τέλος των δεδομένων**. Ο βρόχος αυτός θα ξεκινά με μια γραμμή η οποία θα έχει αριθμό και θα τελειώνει με μια εντολή GOTO <αριθμός γραμμής> η οποία θα ανακυκλώνει το βρόχο.

Στην **αρχή** του βρόχου θα γίνεται έλεγχος εάν η συγκεκριμένη κάρτα που διαβάζεται είναι η ειδική κάρτα που σημαίνει το τέλος των δεδομένων. Είναι βολικό να ορίσουμε ότι η ειδική αυτή κάρτα έχει διατρημένο π.χ. το 3 στην θέση που αντιστοιχεί το 1 για το ελαφροσκυρόδεμα και το 2 για το κοινό σκυρόδεμα.

Για να ξεκινήσουμε, πρέπει πρώτα να διαλέξουμε τα ονόματα των μεταβλητών τα οποία θα χρησιμοποιήσουμε. Έτσι στο πρόγραμμά μας θα χρησιμοποιήσουμε τα:

IELAF είναι ο δείκτης του πλήθους δοκιμίων ελαφροσκυροδέματος
IKSK είναι ο δείκτης του πλήθους δοκιμίων κοινού σκυροδέματος
COMM το μέσο φορτίο θλίψεως
AGEM η μέση ηλικία
WEITM το μέσο βάρος όλων των κύβων
IKMET το πλήθος των κύβων με φορτίο θλίψεως άνω των 50tn

Τις μεταβλητές αυτές θα πρέπει να τις μηδενίσουμε πριν ξεκινήσουμε το βρόχο:

```
IKMET=0
COMM=0.
AGEM=0.
```

```
WEITM=0.
IKSK=0
IELAF=0
```

Ορίζουμε η πρώτη γραμμή του βρόχου να είναι η γραμμή 10,και ξεκινάμε να γράφουμε την διαδικασία η οποία θα επαναλαμβάνεται. Αυτή περιλαμβάνει αρχικά την ανάγνωση των δεδομένων της κάρτας, σύμφωνα με την κατάλληλη FORMAT:

```
  10 READ(5,1000) ICON,IAGE,WEIT,COM
1000 FORMAT(I1,4X,I5,5X,F5.3,5X,F5.2)
```

Πρώτα από όλα, πρέπει να γίνεται ο έλεγχος για την τελευταία ειδική κάρτα. Εάν πρόκειται πράγματι για την τελική κάρτα γίνεται αλλαγή ροής προς την γραμμή π.χ. 20 όπου θα γράψουμε το τελικό τμήμα του προγράμματος. Σε αντίθετη περίπτωση η εκτέλεση του προγράμματος συνεχίζεται κανονικά:

```
IF (ICON.EQ.3) GOTO 20
```

Στη συνέχεια ελέγχουμε πάλι την ICON και εάν είναι 1 τότε αυξάνουμε τον δείκτη του πλήθους των κύβων από ελαφροσκυρόδεμα κατά ένα, ενώ εάν είναι 2 αυξάνουμε τον δείκτη του πλήθους των κύβων από κοινό σκυρόδεμα κατά ένα. Έτσι έχουμε καλύψει όλες τις περιπτώσεις για το ICON:

```
IF (ICON.EQ.1) IELAF=IELAF+1
IF (ICON.EQ.2) IKSK=IKSK+1
```

Στην συνέχεια, θα πρέπει να υπολογίσουμε **το ολικό βάρος των δοκιμίων, το άθροισμα των ηλικιών και το άθροισμα των φορτίων θλίψεως.** Θα μπορούσαμε σε κάθε κύκλο του βρόχου να υπολογίζουμε τις μέσες τιμές των παραπάνω, αλλά κάτι τέτοιο δεν θα ήταν φρόνιμο: Θα προσθέταμε άσκοπούς υπολογισμούς μέσα στο βρόχο, αφού μπορούμε στο τέλος του προγράμματος να τα υπολογίσουμε σύμφωνα με τις τελικές τους τιμές.

Όσο για το συνολικό πλήθος των δοκιμίων που χρειαζόμαστε για τον υπολογισμό των μέσων τιμών, αυτός δεν είναι παρά το άθροισμα του πλήθους των δοκιμίων του

ελαφροσκυροδέματος και του κοινού σκυροδέματος, που ούτως ή άλλως υπολογίζουμε! Για να μην χρησιμοποιούμε πολλές μεταβλητές, μπορούμε να χρησιμοποιήσουμε τις μεταβλητές που παριστάνουν τις μέσες τιμές προσωρινά ως τα αντίστοιχα αθροίσματα. Τελικά θα καταχωρήσουμε σ' αυτές τον εαυτό τους δια τον συνολικό αριθμό των δοκιμίων. Έχουμε λοιπόν:

```
      IAGEM=IAGEM+IAGE
      WEITM=WEITM+WEIT
      COMM=COMM+COM
```

Μετά το πέρας του βρόχου ,οι παραπάνω μεταβλητές θα παριστάνουν το ολικό βάρος των δοκιμίων, το άθροισμα των ηλικιών και το άθροισμα των φορτίων θλίψεως. Στην συνέχεια του βρόχου θα αντιμετωπίσουμε το ερώτημα 4 της άσκησης. Έτσι θα κάνουμε έλεγχο εάν το φορτίο θλίψεως υπερβαίνει τους 50 tn και εάν τούτο είναι αληθές, αυξάνουμε τον αντίστοιχο δείκτη κατά ένα:

```
      IF (COM.GT.50.) IKMET=IKMET+1
```

και ανακυκλώνουμε τον βρόχο:

```
      GO TO 10
```

Τώρα είμαστε έτοιμοι για τους τελικούς υπολογισμούς και την εκτύπωση, αφού οι μεταβλητές μας θα έχουν πάρει τις τελικές τους τιμές όταν θα φτάσει η εκτέλεση του προγράμματος σ' αυτό το σημείο. Δεν ξεχνάμε ότι το αυτό τμήμα του προγράμματος θα ξεκινά με τον αριθμό γραμμής 20, διότι σ' αυτό τον αριθμό γραμμής θα οδηγηθεί η ροή μετά την ανάγνωση της τελικής κάρτας. Υπολογίζουμε τον συνολικό αριθμό των δοκιμίων ΙΑΚ και τον μετατρέπουμε σε πραγματικό για την επικείμενη διαίρεση:

```
20    IAK=IELAF+IKSK
      XIAK=IAK
```

Τέλος, υπολογίζουμε τις μέσες τιμές και αφού εκτυπώσουμε τα αποτελέσματα, κλείνουμε το πρόγραμμά μας.

```
      WEITM=WEITM/XIAK
      COMM=COMM/XIAK
      AGEM=IAGEM/IAK
      WRITE(6,2000) IELAF,IKSK,IKMET
      WRITE(6,2100) WEITM,COMM,AGEM
 2000 FORMAT(1X,I2,1X,I2,1X,I2)
 2100 FORMAT(1X,F8.2,1X,F8.2,1X,F8.2)
      STOP
      END
```

Συνολικά λοιπόν έχουμε:

```
      IKMET=0
      COMM=0.
      AGEM=0.
      WEITM=0.
      IKSK=0
      IELAF=0
   10 READ(5,1000) ICON,IAGE,WEIT,COM
 1000 FORMAT(I1,4X,I5,5X,F5.3,5X,F5.2)
      IF (ICON.EQ.3) GOTO 20
      IF (ICON.EQ.1) IELAF=IELAF+1
      IF (ICON.EQ.2) IKSK=IKSK+1
      IAGEM=IAGEM+IAGE
      WEITM=WEITM+WEIT
      COMM=COMM+COM
      IF (COM.GT.50.) IKMET=IKMET+1
      GO TO 10
   20 IAK=IELAF+IKSK
      XIAK=IAK
      WEITM=WEITM/XIAK
      COMM=COMM/XIAK
      AGEM=IAGEM/IAK
      WRITE(6,2000) IELAF,IKSK,IKMET
```

```
      WRITE(6,2100) WEITM,COMM,AGEM
 2000 FORMAT(1X,I2,1X,I2,1X,I2)
 2100 FORMAT(1X,F8.2,1X,F8.2,1X,F8.2)
      STOP
      END
```

Άσκηση 3.13

Γράψτε μια υπορουτίνα που να υπολογίζει και αποθηκεύει στη θέση result το δεκαδικό μέρος του πηλίκου που παίρνομε διαιρώντας το int1 με το int2. Τα int1 και int2 υπολογίζονται σε κάποια άλλη υπορουτίνα και το result σε μια τρίτη.

Λύση:

Καταρχήν το κύριο μέρος του προγράμματος δεν θα περιέχει τίποτα άλλο από την κλήση των υπορουτινών με την βοήθεια της εντολής CALL.Στην υπορουτίνα INP θα γίνεται η εισαγωγή των int1,int2.Στην υπορουτίνα CALC θα γίνεται ο κυρίως υπολογισμός του δεκαδικού μέρους της διαίρεσης και στην υπορουτίνα OUT θα γίνεται η εκτύπωση του αποτελέσματος. Έτσι το κύριο μέρος θα έχει ως εξής:

```
      CALL INP
      CALL CALC
      CALL OUT
      STOP
      END
```

Γράφουμε την υπορουτίνα INP για την εισαγωγή των δεδομένων. Όλες οι υπορουτίνες θα έχουν κοινές μεταβλητές τις int1,int2,result και αυτό το δηλώνουμε με την εντολή COMMON **σε κάθε υπορουτίνα και με την ίδια σειρά στις μεταβλητές**. Στο τέλος κάθε υπορουτίνας πρέπει να υπάρχει απαραίτητα μια εντολή RETURN:

```
      SUBROUTINE INP
      COMMON INT1,INT2,RESULT
      READ(5,1000) INT1,INT2
```

```
1000 FORMAT(1X,I10,I10)
     RETURN
     END
```

Η υπορουτίνα CALC βασίζεται στο εξής σκεπτικό: Εάν διαιρέσουμε το int1 με το int2 θα πάρουμε το **ακέραιο μέρος της διαίρεσης.** Εάν πραγματοποιήσουμε την ίδια διαίρεση αφού τους μετατρέψουμε σε πραγματικούς, θα πάρουμε την κανονική τιμή της διαίρεσης. Αφαιρώντας τα παραπάνω πηλίκα παίρνουμε τα ζητούμενα δεκαδικά! Για να γίνει περισσότερο κατανοητό αυτό υποθέστε ότι int1=3,int2=2.Τότε **int1/int2=1** ενώ αν πρώτα τους μετατρέψουμε σε πραγματικούς θα έχουμε **rint1/rint2=1.5**.

Αφαιρώντας το 1 από το 1.5 έχουμε το ζητούμενο δεκαδικό μέρος, δηλ το 0.5!Τέλος για να μην εμφανίσουμε αρνητικό δεκαδικό μέρος, κάνουμε έλεγχο εάν το result είναι αρνητικό και αν είναι το πολλαπλασιάζουμε με -1. για να το κάνουμε θετικό. Με βάση αυτά έχω:

```
     SUBROUTINE CALC
     COMMON INT1,INT2,RESULT
     RINT1=INT1
     RINT2=INT2
     IRES=INT1/INT2
     RES=IRES
     RESULT=RINT1/RINT2-RES
     IF (RESULT.LT.0.) RESULT=RESULT*(-1.)
     RETURN
     END
```

Η τελική υπορουτίνα εκτυπώνει το αποτέλεσμα:

```
     SUBROUTINE OUT
     COMMON INT1,INT2,RESULT
     WRITE (6,1100) RESULT
1100 FORMAT(1X,F9.7)
     RETURN
     END
```

Συνολικά έχω:

```
C
C     Κύριο Μέρος Προγράμματος
C
      CALL INP
      CALL CALC
      CALL OUT
      STOP
      END
C
C     Υποροutίνα INP: υπολογισμός int1 & int2.
C
      SUBROUTINE INP
      COMMON INT1,INT2,RESULT
      READ(5,1000) INT1,INT2
 1000 FORMAT(1X,I10,I10)
      RETURN
      END
C
C     Υποροutίνα CALC: υπολογισμός του δεκαδικού μέρους της C     διαίρεσης.
C
      SUBROUTINE CALC
      COMMON INT1,INT2,RESULT
      RINT1=INT1
      RINT2=INT2
      IRES=INT1/INT2
      RES=IRES
      RESULT=RINT1/RINT2-RES
      IF (RESULT.LT.0.) RESULT=RESULT*(-1.)
      RETURN
      END
C
```

```
C     Υποροutίνα OUT: εμφάνιση αποτελέσματος.
C
      SUBROUTINE OUT
      COMMON INT1,INT2,RESULT
      WRITE (6,1100) RESULT
 1100 FORMAT(1X,F9.8)
      RETURN
      END
```

ΚΕΦΑΛΑΙΟ 4: Λυμένες Ασκήσεις

ΕΝΟΤΗΤΑ Α.

Παρατήρηση: Η ενότητα αυτή περιέχει ασκήσεις λιγότερο αναλυτικά λυμένες και πιο δύσκολες από αυτές του κεφαλαίου 4, όμως θα πρέπει να μελετηθούν από όλους.

Άσκηση 4.1

Γράψτε μια υπορουτίνα που να βρίσκει σ' ένα πίνακα μιας διαστάσεως με Ν στοιχεία το μεγαλύτερο ΑΜΑΧ και το μικρότερο ΑΜΙΝ από αυτά.

Λύση:

Για λόγους πληρότητας, θα γράψουμε ένα πρόγραμμα στο οποίο θα γίνεται εισαγωγή των δεδομένων από κάρτα, επεξεργασία των δεδομένων κατά τις υποδείξεις της άσκησης και εκτύπωση των αποτελεσμάτων. Επειδή θα χρησιμοποιήσουμε πίνακα πρέπει πρώτα από όλα να ξέρουμε το πλήθος Ν των στοιχείων του.(Ως διάσταση του πίνακα θα θέσουμε έναν ικανό αριθμό π.χ. 1000) Καταρχήν όλες οι υπορουτίνες που θα γράψουμε θα έχουν κοινές μεταβλητές τις AMAX,AMIN,N,A όπου ΑΜΑΧ το μέγιστο στοιχείο, ΑΜΙΝ το ελάχιστο στοιχείο, Ν το πλήθος των στοιχείων και Α ο πινακάς μας. Αυτό θα δηλώνεται με την κατάλληλη εντολή COMMON σε **κάθε** υπορουτίνα, όπως και η δήλωση της διάστασης του πίνακα. Τέλος δεν πρέπει να ξεχνάμε να κλείνουμε κάθε υπορουτίνα με τις εντολές **RETURN - END.**

Η πρώτη μας υπορουτίνα VOFN θα διαβάζει το πλήθος Ν:

```
      SUBROUTINE VOFN
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      READ(5,1000) N
 1000 FORMAT(1X,I4)
      RETURN
```

```
      END
```

Στην συνέχεια γράφουμε την υπορουτίνα LISTA που διαβάζει (έχοντας ως γνωστό το πλήθος Ν των στοιχείων) τα δεδομένα από κάρτες:

```
      SUBROUTINE LISTA
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      DO 50 NC=1,N,1
      READ(5,1050) A(NC)
 1050 FORMAT(1X,F10.3)
   50 CONTINUE
      RETURN
      END
```

Η εισαγωγή των δεδομένων γίνεται με βρόχο Ν επαναλήψεων και οι τα δεδομένα καταχωρούνται ως στοιχεία του πίνακα Α. Τώρα είμαστε έτοιμοι για να γράψουμε την κυρίως υπορουτίνα (CALC),που θα βρίσκει το μέγιστο και το ελάχιστο στοιχείο της λίστας Α.Το σκεπτικό είναι το εξής: Στην αρχή θέτουμε ως μέγιστο **και** ως ελάχιστο το πρώτο στοιχείο της λίστας (Α(1)).

Στην συνέχεια κάνουμε ένα βρόχο Ν επαναλήψεων που να διαβάζει ένα - ένα τα στοιχεία του πίνακα Α. Αν το στοιχείο Α(NC) που διαβάζεται είναι μεγαλύτερο από το **προσωρινό** μέγιστο AMAX (το οποίο έχει αρχική τιμή Α(1)) τότε το Α(NC) καταχωρείται ως νέο μέγιστο AMAX. Εάν Α(NC)<AMAX τότε προχωράμε στον έλεγχο για το ελάχιστο. Εάν το Α(NC) είναι μικρότερο από το **προσωρινό** ελάχιστο AMIN τότε η τιμή του Α(NC) καταχωρείται ως νέο AMIN. Εάν ούτε αυτή η συνθήκη είναι αληθής τότε είμαστε σίγουροι ότι AMIN < Α(NC) < AMAX οπότε ανακυκλώνουμε το βρόχο και ελέγχουμε το επόμενο στοιχείο.

Είναι προφανές ότι η αλήθεια της πρώτης συνθήκης αποκλείει την αλήθεια της δεύτερης και αντίστροφα (δηλαδή δεν μπορεί ένα στοιχείο να είναι και μεγαλύτερο από το μέγιστο και μικρότερο από το ελάχιστο) Μετά τον έλεγχο όλων των στοιχείων, οι τιμές των AMAX, AMIN θα αντιπροσωπεύουν το ολικό μέγιστο και ελάχιστο αντίστοιχα του πίνακα Α. Με βάση τα παραπάνω έχω:

```
      SUBROUTINE CALC
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      AMAX=A(1)
      AMIN=A(1)
      DO 100 NC=1,N,1
      IF (AMAX.LT.A(NC)) AMAX=A(NC)
      IF (AMIN.GT.A(NC)) AMIN=A(NC)
  100 CONTINUE
      RETURN
      END
```

Τέλος χρειαζόμαστε την υπορουτίνα OUT που θα εκτυπώνει τα αποτελέσματα AMAX, AMIN της επεξεργασίας:

```
      SUBROUTINE OUT
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      WRITE(6,1100) AMIN,AMAX
 1100 FORMAT(1X,'Mikroterh timh=',F10.3,/,' Megalyterh
     *timh=',F10.3)
      RETURN
      END
```

Τώρα γράφουμε το κυρίως πρόγραμμα με το οποίο θα καλούμε τις υπορουτίνες με την κατάλληλη σειρά και θα κλείνουμε το πρόγραμμα με τις εντολές STOP - END. Συνολικά λοιπόν:

```
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      CALL VOFN
      CALL LISTA
      CALL CALC
```

```
      CALL OUT
      STOP
      END

C     Υπορουτίνα VOFN: μας δίνει την τιμή του ν (διαβάζεται
C     από κάρτα)

      SUBROUTINE VOFN
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      READ(5,1000) N
 1000 FORMAT(1X,I4)
      RETURN
      END
C
C     Υπορουτίνα LISTA: διαβάζει τα ν στοιχεία της λίστας Α
C     από κάρτα.

      SUBROUTINE LISTA
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      DO 50 NC=1,N,1
      READ(5,1050) A(NC)
 1050 FORMAT(1X,F10.3)
   50 CONTINUE
      RETURN
      END

C
C     Υπορουτίνα CALC: υπολογίζει τη μεγαλύτερη και
C     μικρότερη τιμή που υπάρχουν στα ν στοιχεία της λίστας.

      SUBROUTINE CALC
      DIMENSION A(1000)
```

```
      COMMON AMAX,AMIN,N,A
      AMAX=A(1)
      AMIN=A(1)
      DO 100 NC=1,N,1
      IF (AMAX.LT.A(NC)) AMAX=A(NC)
      IF (AMIN.GT.A(NC)) AMIN=A(NC)
  100 CONTINUE
      RETURN
      END
C
C     Υπορουτίνα OUT: εμφάνιση αποτελεσμάτων
C
      SUBROUTINE OUT
      DIMENSION A(1000)
      COMMON AMAX,AMIN,N,A
      WRITE(6,1100) AMIN,AMAX
 1100 FORMAT(1X,'Mikroterh timh=',F10.3,/,' Megalyterh
     *timh=',F10.3)
      RETURN
      END
```

Άσκηση 4.2

Για μια λίστα ν θετικών αριθμών βρείτε:

- το μέγιστο στοιχείο της λίστας
- πόσες φορές επαναλαμβάνεται
- θέση της τιμής που εμφανίζεται για πρώτη φορά
- θέση της τιμής που εμφανίζεται για τελευταία φορά

Λύση:

Καταρχήν ορίζουμε τον πίνακα με τον οποίο θα δουλέψουμε, έστω Α(1000), καθώς και τις μεταβλητές που θα χρησιμοποιήσουμε. Έστω AMAX το μέγιστο, IARX ο δείκτης του στοιχείου που εμφανίστηκε πρώτη φορά το AMAX, ITEL ο δείκτης που εμφανίστηκε

τελευταία φορά ,IPL το πλήθος των εμφανίσεων του AMAX. Στη συνέχεια, αφού δώσουμε αρχικές τιμές στις μεταβλητές, διαβάζουμε το πλήθος N των στοιχείων που θα εισαχθούν και κάνουμε έλεγχο ώστε το N να είναι μεγαλύτερο από το 1 και μικρότερο από το 1000:

```
      DIMENSION A(1000)
      AMAX=A(1)
      IARX=1
      ITEL=1
      IPL=1
   10 READ(5,1000) N
 1000 FORMAT(1X,I4)
      IF (N.GT.1000.OR.N.LT.1) GOTO 10
```

Με την βοήθεια βρόχου N επαναλήψεων, εισάγουμε τα δεδομένα, καταχωρώντας απευθείας ως στοιχεία του πίνακα A:

```
   20 DO 50 J=1,N,1
      READ(5,1100) A(J)
 1100 FORMAT(1X,F10.4)
   50 CONTINUE
```

Στην συνέχεια προχωρούμε στον κυρίως βρόχο, που θα είναι και αυτός N επαναλήψεων, με μετρητή έστω το NC.Για να χειριστούμε καλύτερα τις αλλαγές ροής, είναι προτιμότερο ο βρόχος να φτιαχτεί με απλές εντολές GOTO.Ο βρόχος αυτός θα περιέχει τρία "φίλτρα" που θα λειτουργούν ως εξής:

ΦΙΛΤΡΟ 1: Εάν ο δείκτης NC είναι μεγαλύτερος του N τότε πήγαινε στην γραμμή 200(κάτι που σημαίνει το τέλος του βρόχου επανάληψης και την εκτύπωση των αποτελεσμάτων)
Εάν η συνθήκη του φίλτρου 1 είναι ψευδής προχωράμε στο επόμενο φίλτρο:

ΦΙΛΤΡΟ 2:Εάν το A(NC) είναι μεγαλύτερο του προσωρινού μέγιστου τότε πήγαινε στην γραμμή 110.

Εάν η συνθήκη είναι αληθής (A(NC) > AMAX) τότε έχουμε νέο μέγιστο και η ροή αλλάζει προς την γραμμή 110 όπου εκτελούνται οι κατάλληλες εντολές. Οι εντολές αυτές περιλαμβάνουν κατάλληλες καταχωρήσεις μεταβλητών και συγκεκριμένα:

- Καταχώρηση νέου μέγιστου: AMAX=A(NC)
- Πρώτη θέση που εμφανίστηκε το μέγιστο:IARX=NC
- Τελευταία θέση που εμφανίστηκε το μέγιστο: ITEL=NC. (θεωρούμε λοιπόν ότι αυτή είναι η τελευταία φόρα που εμφανίστηκε το AMAX. Εάν ξαναεμφανιστεί το AMAX θα καταχωρήσουμε την νέα τιμή του δείκτη NC,και έτσι θα έχουμε πάντα την **τελευταία** φορά που εμφανίστηκε το AMAX)
- Πλήθος επαναλήψεων επαναφορά στο 1:IPL=1
- Αύξηση του δείκτη NC κατά ένα και επιστροφή στην αρχή του βρόχου (δηλαδή στην γραμμή 100).

Εάν η συνθήκη του δεύτερου φίλτρου είναι ψευδής ,η εκτέλεση του προγράμματος συνεχίζεται από το τρίτο και τελευταίο φίλτρο:

ΦΙΛΤΡΟ 3:Εάν το A(NC) είναι ίσο με το προσωρινό μέγιστο AMAX τότε πήγαινε στην γραμμή 120, Όμοια, η γραμμή 110 όπου συνεχίζεται η ροή εάν η συνθήκη του φίλτρου 3 είναι αληθής, περιλαμβάνει τις κατάλληλες εντολές. Αυτές περιλαμβάνουν:

- "Μεταφορά" του δείκτη που υποδεικνύει την τελευταία θέση που εμφανίστηκε το μέγιστο: ITEL=NC
- Πλήθος επαναλήψεων του μέγιστου αύξηση κατά ένα: IPL=1
- Αύξηση του δείκτη NC κατά ένα και επιστροφή στην αρχή του βρόχου(δηλαδή στην γραμμή 100).

Εάν οι συνθήκες των φίλτρων 2,3 είναι και οι δύο ψευδείς, ο αριθμός που διαβάστηκε στον συγκεκριμένο κύκλο επανάληψης, δεν παρουσιάζει κανένα ενδιαφέρον, αφού είναι μικρότερος από το προσωρινό μέγιστο. Έτσι αυξάνουμε τον δείκτη NC κατά ένα και επιστρέφουμε στην αρχή του βρόχου ελέγχοντας το επόμενο στοιχείο του πίνακα Α. Όλα αυτά μεταφραζόμενα σε κώδικα έχουν ως εξής:

```
      NC=1
100 IF (NC.GT.N) GOTO 200
```

```
      IF (A(NC).GT.AMAX) GOTO 110
      IF (A(NC).EQ.AMAX) GOTO 120
      NC=NC+1
      GOTO 100

C     ΝΕΟ ΜΕΓΙΣΤΟ
  110 AMAX=A(NC)
      IARX=NC
      ITEL=NC
      IPL=1
      NC=NC+1
      GOTO 100

C     ΕΠΑΝΑΛΗΨΗ ΜΕΓΙΣΤΟΥ
  120 IPL=IPL+1
      ITEL=NC
      NC=NC+1
      GOTO 100
```

Είναι προφανές ότι μετά το πέρας του βρόχου, το AMAX θα είναι το **ολικό** μέγιστο, το IARX και το ITEL θα υποδεικνύουν την θέση που βρέθηκε πρώτη και τελευταία φορά το AMAX και το IPL θα δίνει το πλήθος των επαναλήψεων του AMAX. Τελικά προχωρούμε από την γραμμή 200 στην εκτύπωση των αποτελεσμάτων και κλείνουμε το πρόγραμμα:

```
  200 WRITE(6,2000) AMAX,IARX,ITEL,IPL
 2000 FORMAT(1X,'Megisth timh =',F10.4,/,1X,'Arxikh thesh
     *=',I4,/,1X,'Te1likh thesh =',I4,/,1X,'Plhthos
     *emfanisewn =',I4,/,1X)
      STOP
      END
```

Τελικά το πρόγραμμά μας έχει ως εξής:

```
      DIMENSION A(1000)
```

```
      AMAX=A(1)
      IARX=1
      ITEL=1
      IPL=1
C     Ανάγνωση του πλήθους των στοιχείων της λίστας.
   10 READ(5,1000) N
 1000 FORMAT(1X,I4)
      IF (N.GT.1000.OR.N.LT.1) GOTO 10
C     Εισαγωγή στοιχείων λίστας.
   20 DO 50 J=1,N,1
      READ(5,1100) A(J)
 1100 FORMAT(1X,F10.4)
   50 CONTINUE

C     Βρόχος υπολογισμών.

      NC=1
  100 IF (NC.GT.N) GOTO 200
      IF (A(NC).GT.AMAX) GOTO 110
      IF (A(NC).EQ.AMAX) GOTO 120
      NC=NC+1
      GOTO 100
C     ΝΕΟ ΜΕΓΙΣΤΟ
  110 AMAX=A(NC)
      IARX=NC
      ITEL=NC
      IPL=1
      NC=NC+1
      GOTO 100
C     ΕΠΑΝΑΛΗΨΗ ΜΕΓΙΣΤΟΥ
  120 IPL=IPL+1
      ITEL=NC
      NC=NC+1
      GOTO 100
```

```
C     Εκτυπώσεις & Αποτελέσματα υπολογισμών
  200 WRITE(6,2000) AMAX,IARX,ITEL,IPL
 2000 FORMAT(1X,'Megisth timh =',F10.4,/,1X,'Arxikh thesh
     *=',I4,/,1X,'Te1likh thesh =',I4,/,1X,'Plhthos
     *emfanisewn =',I4,/,1X)
      STOP
      END
```

Άσκηση 4.3

Γράψτε ένα διάγραμμα ροής και αντίστοιχο πρόγραμμα που να βάζει τις Ν θετικές τιμές λίστας σε αύξουσα τάξη.

Παρατήρηση: Το πλήθος Ν των στοιχείων τις λίστας βρίσκεται στη μεταβλητή IN και μπορεί να πάρει οποιαδήποτε θετική (μη μηδενική) ακέραια τιμή. Η εντολή όμως dimension πρέπει να έχει επίσης την ίδια τιμή με τη μεταβλητή IN.

Λύση:

Καταρχήν θα δημιουργήσω τον πίνακα - λίστα του οποίου τα στοιχεία θα θέσω σε αύξουσα τάξη. Έστω ότι ο πίνακας αυτός είναι ο Α με διάσταση π.χ. 8. Τότε IN=8 .Στην συνέχεια προχωρούμε στην εισαγωγή των τιμών:

```
      IN=8
      DIMENSION A(8)
      DO 100 NC=1,IN
      READ(5,1000) A(NC)
 1000 FORMAT (1X,F10.3)
  100 CONTINUE
```

Η λογική του προγράμματος θα είναι η εξής: Ας υποθέσω ότι έχω 4 αριθμούς με την σειρά: 5,1,3,2. Ελέγχω αν το δεύτερο στοιχείο είναι μικρότερο από το πρώτο. Αν αληθεύει ανταλλάσσω τις τιμές τους δηλαδή αν 1<5 τότε παίρνω τη σειρά 1,5,3,2.Ομοίως ελέγχω αν το τρίτο στοιχείο είναι μικρότερο από το δεύτερο, δηλαδή αν 5<3 οπότε επειδή αληθεύει το

ανταλλάσσω παίρνοντας τη σειρά 1,3,5,2 συνεχίζω και τελικά παίρνω 1,3,2,5. Αν επαναλάβω τότε παίρνω 1,3,2,5 - 1,2,3,5 και οι υπόλοιπες αλλαγές είναι άνευ ουσίας δηλαδή δε μεταβάλλουν τη λίστα.

Έτσι, η λύση του προβλήματός μας είναι η χρησιμοποίηση διπλού βρόχου, που θα συγκρίνει ένα - ένα τα στοιχεία της λίστας με όλα τα άλλα. Έχω:

```
      DO 300 LC=1,IN
      DO 200 NC=1,IN
      IF (A(LC).GE.A(NC)) GOTO 200
      C=A(NC)
      A(NC)=A(LC)
      A(LC)=C
  200 CONTINUE
  300 CONTINUE
```

Τέλος, αφού η λίστα μας έχει τεθεί σε αύξουσα τάξη, φτιάχνουμε το τμήμα του προγράμματος που θα εκτυπώνει την νέα λίστα:

```
      DO 400 KC=1,IN,1
      WRITE(6,2000) A(KC)
 2000 FORMAT(1X,F10.3)
  400 CONTINUE
```

και κλείνουμε το πρόγραμμά μας:

```
      STOP
      END
```

Συνολικά έχουμε:

```
C
C   Εισαγωγή τιμών.
C
      IN=8
```

```
      DIMENSION A(8)
      DO 100 NC=1,IN
      READ(5,1000) A(NC)
 1000 FORMAT (1X,F10.3)
  100 CONTINUE
C
C   Επεξεργασία τιμών.
C
      DO 300 LC=1,IN
      DO 200 NC=LC,IN
      IF (A(LC).GE.A(NC)) GOTO 200
C     ΕΝΑΛΛΑΓΉ ΤΙΜΩΝ
      C=A(NC)
      A(NC)=A(LC)
      A(LC)=C
  200 CONTINUE
  300 CONTINUE
C
C    Εκτύπωση των αποτελεσμάτων.
C
      DO 400 KC=1,IN,1
      WRITE(6,2000) A(KC)
 2000 FORMAT(1X,F10.3)
  400 CONTINUE
      STOP
      END
```

Άσκηση 4.4

Δίνονται δύο μονοδιάστατα μητρώα IFIR,ISEC με NF,NS στοιχεία αντίστοιχα. Γράψτε ένα πρόγραμμα που να βρίσκει τον μικρότερο από το IFIR και τον μεγαλύτερο από το ISEC,να τυπώνει την παλιά τους τιμή και τον δείκτη τους. Στην συνέχεια να ανταλλάσσει το ψηφίο των δεκάδων του ελάχιστου, με το ψηφίο των χιλιάδων του μέγιστου και να τυπώνει την νέα τους τιμή.

Λύση:

Αφού εισάγουμε τα δεδομένα στους πίνακες IFIR, ISEC και βρούμε κατά τα γνωστά το ελάχιστο του πρώτου και το μέγιστο του δεύτερου, προχωράμε στην εκτύπωση των τιμών αυτών καθώς και των δεικτών των στοιχείων.

Στη συνέχεια προχωρούμε στο κυρίως πρόβλημα: πρέπει με κάποιο τρόπο να "βρούμε" το ψηφίο των δεκάδων του μικρότερου με το ψηφίο των χιλιάδων του μεγαλύτερου. Η συλλογιστική είναι η εξής: Έστω ότι το ελάχιστο στοιχείο IFMIN είναι το 127. Για να βρούμε το ψηφίο των δεκάδων διαιρούμε το 127 με τον ακέραιο 10,οπότε παίρνουμε τον ακέραιο 12 (η διαίρεση δίνει 12.7, όμως με την απόρριψη των ψηφίων παίρνουμε το 12), που τον ονομάζουμε ΜΑ.

Διαιρούμε και τον ΜΑ με τον ακέραιο 10,οπότε όμοια παίρνουμε το ΜΒ=1. Πολλαπλασιάζουμε το ΜΒ με το 10, οπότε παίρνουμε το ΜC=10.Αφαιρώντας τώρα το ΜC από το ΜΑ παίρνουμε το ζητούμενο ψηφίο, δηλαδή το 2!

Για να κατανοηθεί ο τρόπος ας κάνουμε άλλο ένα παράδειγμα: Έστω ότι ο μικρότερος είναι 856.Τοτε ΜΑ=85,ΜΒ=8,ΜC=80,ΜΑ-ΜC=5! Όμοια βρίσκουμε το ψηφίο των χιλιάδων του ISMAX.Έστω ότι το μέγιστο είναι το 14567.Το διαιρώ με το 1000 (γιατί ψάχνω το ψηφίο των χιλιάδων) και παίρνω ΝΑ=14.Διαιρώ το ΝΑ με το 10 και παίρνω ΝΒ=1. Πολλαπλασιάζω το ΝΒ με το 10 και παίρνω 10.Αφαιρώντας το 10 από το 14 παίρνω το ζητούμενο ψηφίο.

Γνωρίζοντας τώρα τα ψηφία των δεκάδων του ελάχιστου (ΜΤ) και των χιλιάδων του μέγιστου (ΝΤ), μπορούμε να τα αντικαταστήσομε ως εξής: Αφαιρούμε από το ελάχιστο ΜΤ δεκάδες και του προσθέτουμε ΝΤ δεκάδες. Έτσι για το ελάχιστο 856 του προηγούμενου παραδείγματος (με ΜΤ=5, ΝΤ=4 για το 14567), αφαιρούμε ΜΤ=5 δεκάδες (προκύπτει το 806) και προσθέτουμε ΝΤ=4 δεκάδες οπότε προκύπτει το ζητούμενο 846.

Όμοια για το μέγιστο, αφαιρούμε ΝΤ χιλιάδες και προσθέτουμε ΜΤ χιλιάδες, οπότε έχω 14567 - (4*1000) + (5*1000) = 15567,το οποίο είναι και ο ζητούμενος αριθμός.

Παρόλο που ο παραπάνω αλγόριθμος φαίνεται ότι λειτουργεί σε όλες τις περιπτώσεις, μια προσεκτικότερη ματιά δείχνει ότι παρουσιάζονται προβλήματα όταν π.χ. το ελάχιστο

είναι αρνητικός και το μέγιστο θετικός. Εξάλλου στην περίπτωση που μας ζητούσαν να τυπώσουμε το ψηφίο π.χ. των δεκάδων που βρήκαμε, και ο αριθμός είναι αρνητικός, το αποτέλεσμα της παραπάνω διαδικασίας θα είναι ένας αρνητικός μονοψήφιος.

Για να αποφύγουμε τέτοιες περιπτώσεις, **στον κυρίως αλγόριθμο εμφανίζουμε τα IFMIN,ISMAX πολλαπλασιασμένα με έναν συντελεστή, αντίστοιχα LF,LS ο οποίος ,για τον κάθε αριθμό παίρνει την τιμή 1 εάν ο αριθμός είναι θετικός (οπότε δεν επηρεάζει την διαδικασία) και την τιμή -1 εάν είναι ο αριθμός αρνητικός. (οπότε μετατρέπεται σε θετικό).** Έτσι κατά την εκτέλεση του κυρίως αλγόριθμου, οι χρησιμοποιούμενοι αριθμοί θα είναι σε κάθε περίπτωση θετικοί, και ο αλγόριθμος θα λειτουργεί σωστά.

Πως όμως θα εμφανίσουμε τους νέους αριθμούς, με ανταλλαγμένα τα στοιχεία, εάν πριν ήταν αρνητικοί; Το αποτέλεσμα θα είναι θετικό. Έτσι πριν ακριβώς από την εκτύπωση ,πολλαπλασιάζουμε ξανά τα αποτελέσματα IFMIN, ISMAX με τους συντελεστές LF,LS οπότε παίρνουμε τα "αρνητικά" αποτελέσματα ,εάν κάποιο από τα IFMIN, ISMAX ήταν αρνητικό. Τελικά έχουμε:

```
      DIMENSION IFIR(1000),ISEC(2000)
C
C     EISAGOGH DEDOMENON
C
      READ(5,500) NF,NS
  500 FORMAT (2(I4))

      DO 100 KA=1,NF,1
      READ(5,1000) IFIR(KA)
  100 CONTINUE

      DO 200 KB=1,NS,1
      READ(5,1000) ISEC(KB)
  200 CONTINUE
 1000 FORMAT (I4)
C
```

```
C     ARXIKOPOIHSH METABLHTON
C
      IFMIN=IFIR(1)
      ISMAX=ISEC(1)
      LA=1
      LB=1
      JF=1
      JS=1
C
C     EYRESH MIROTEROY - MEGALYTEROY DEIKTWN KAI EKTYPWSH
C
    5 IF (LA.GT.NF) GOTO 15
      IF (IFIR(LA).LT.IFMIN) GOTO 10
      LA = LA + 1
      GOTO 5

   10 IFMIN = IFIR(LA)
      JF = LA
      LA = LA + 1
      GOTO 5

   15 IF (LB.GT.NS) GOTO 25
      IF(ISEC(LB).GT.ISMAX) GOTO 20
      LB = LB + 1
      GOTO 15

   20 ISMAX = ISEC(LB)
      JS = LB
      LB = LB + 1
      GOTO 15

   25 WRITE(6,2000) IFMIN,ISMAX,JF,JS
C
C     XEIRISMOS ARNHTIKWN ARI8MWN
```

```
C
      LF = 1
      LS = 1
      IF (IFMIN.LT.0) LF = -1
      IF (ISMAX.LT.0) LS = -1

C
C     ANTALLAGH PSHFIWN KAI EKTYPWSH TELIKWN TIMWN
C
      MA = LF*IFMIN / 10
      MB = MA / 10
      MC = MB * 10
      MT = MA - MC

      NA = LS*ISMAX / 1000
      NB = NA / 10
      NC = NB * 10
      NT = NA - NC

      IFMIN = LF*IFMIN - (MT * 10) + (NT * 10)
      ISMAX = LS*ISMAX - (NT * 1000) + (MT * 1000)

      IFMIN=LF*IFMIN
      ISMAX=LS*ISMAX

      WRITE(6,2000) IFMIN,ISMAX,JF,JS
 2000 FORMAT(4(1X,I4))
      STOP
      END
```

Άσκηση 4.5

Δίνεται ένα μονοδιάστατο μητρώο IP με N ακεραίους αριθμούς όπου N=<1000. Γράψτε ένα πρόγραμμα που να βρίσκει και να εκτυπώνει τον αριθμό K, το άθροισμα ISUM

και το μέσο όρο AVER των στοιχείων του μητρώου που είναι συγχρόνως περιττοί και πολλαπλάσια του 7.

Λύση:

Καταρχήν, ορίζουμε τον πίνακα IP και γράφουμε την ρουτίνα εισαγωγής των δεδομένων. Αρχικά εισάγουμε το N, κάνουμε έλεγχο για την ορθότητα του και στην συνέχεια με βρόχο N επαναλήψεων εισάγουμε τα στοιχεία του μητρώου IP.

Για την λύση του κυρίως προγράμματος, πρέπει να ελέγξουμε ένα - ένα τα στοιχεία του μητρώου εάν είναι ακέραια πολλαπλάσια του 7 και **δεν** είναι ακέραια πολλαπλάσια του 2.**Γενικότερα ένας ακέραιος Κ είναι ακέραιο πολλαπλάσιο του Μ (επίσης ακέραιου) εάν και μόνο αν ισχύει: (Κ / Μ) * Μ = Κ**.

Αυτό συμβαίνει διότι η διαίρεση είναι διαίρεση ακεραίων, οπότε παίρνουμε το ακέραιο μέρος του πηλίκου. Εάν η παραπάνω ισότητα ισχύει, σημαίνει ότι το ακέραιο μέρος του πηλίκου Κ / Μ είναι ίσο με το ίδιο πηλίκο, θεωρώντας τους Κ , Μ πραγματικούς. Πρακτικά αυτό σημαίνει ότι το Κ είναι ακέραιο πολλαπλάσιο του Μ. Σύμφωνα με αυτά έχω:

```
      DIMENSION IP(1000)
C     ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ
   10 READ(5,1000) N
 1000 FORMAT(I4)
      IF(N.GT.1000) GOTO 10
      DO 50 MC=1,N,1
      READ(5,1200) IP(MC)
 1200 FORMAT(I6)
   50 CONTINUE
      K=0
      ISUM=0
      AVER=0.
C     ΚΥΡΙΩΣ ΒΡΟΧΟΣ
      DO 100 I=1,N,1
      IF((IP(I)/2)*2.EQ.IP(I).AND.(IP(I)/7)*7.NE.IP(I)) GOTO
     *100
```

```
      K=K+1
      ISUM=ISUM+IP(I)
  100 CONTINUE
      IF(K.EQ.0) GOTO 110
      XK=K
      XISUM=ISUM
      AVER=XISUM/XK
C     ΕΚΤΥΠΩΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
  110 WRITE(6,2000)K,ISUM
 2000 FORMAT(1X,'SYNOLO ARI8MWM=',I4,1X,'A8ROISMA=',I12)
      WRITE(6,2200) AVER
 2200 FORMAT(1X,'MESOS OROS=',F12.4)
      STOP
      END
```

Άσκηση 4.6

ΘΕΜΑ 1 - ΙΟΥΝΙΟΣ 1996 Δίνεται μονοδιάστατο μητρώο ITEST με N θετικά στοιχεία όπου N<=2000. Γράψτε ένα πρόγραμμα με το αντίστοιχο διάγραμμα ροής το οποίο για κάθε στοιχείο του ITEST θα δημιουργεί και θα εκτυπώνει τον διψήφιο που έχει για ψηφίο δεκάδων το ψηφίο των εκατοντάδων του στοιχειού του ITEST και ψηφίο μονάδων το ψηφίο επίσης των μονάδων του στοιχείου του ITEST.
Π.χ. 1234567 ⇒ 57 , 309 ⇒ 39 , 57 ⇒ 7

Λύση:

Το πρόβλημα είναι παρόμοιο (αν και πιο εύκολο) με το πρόβλημα 4.5. Εδώ δεν έχουμε αρνητικούς αριθμούς, ενώ η γνωστή μέθοδος με την οποία δημιουργούμε τον ζητούμενο διψήφιο λειτουργεί κανονικά ακόμη και αν το στοιχείο του ITEST δεν έχει ψηφίο εκατοντάδων (τότε ο υπολογισμός θα γίνει με ψηφίο εκατοντάδων το 0). Στο τέλος του φυλλαδίου παρατίθεται το διάγραμμα ροής. Το πρόγραμμά μας έχει ως εξής:

```
      DIMENSION ITEST(2000)
C
C     ΕΙΣΑΓΩΓΗ N
```

```
C
   10 READ(5,1000) N
 1000 FORMAT(I3)
      IF(N.LT.1.OR.N.GT.2000) GOTO 10
C
C     ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ & ΕΠΕΞΕΡΓΑΣΙΑ ΤΟΥΣ
C
      DO 100 NA=1,N,1
      READ(5,1500) ITEST(NA)
 1500 FORMAT(I8)
C
C     ΕΥΡΕΣΗ ΨΗΦΙΟΥ ΕΚΑΤΟΝΤΑΔΩΝ
C
      JA=ITEST(NA)/100
      JB=JA/10
      JC=JB*10
      JT=JA-JC
C
C     ΤΟ ΕΙΝΑΙ ΤΟ ΨΗΦΙΟ ΤΩΝ ΕΚΑΤΟΝΤΑΔΩΝ
C     ΟΜΟΙΑ ΒΡΙΣΚΟΥΜΕ ΤΟ ΨΗΦΙΟ ΤΩΝ ΜΟΝΑΔΩΝ
C
      KA=ITEST(NA)/10
      KB=KA*10
      KT=ITEST(NA)-KB
C
C     ΚΤ ΕΙΝΑΙ ΤΟ ΨΗΦΙΟ ΤΩΝ ΜΟΝΑΔΩΝ
C     ΔΗΜΙΟΥΡΓΟΥΜΕ ΤΟΝ ΖΗΤΟΥΜΕΝΟ ΔΙΨΗΦΙΟ & ΕΚΤΥΠΩΝΟΥΜΕ
C
      JZHT=JT*10+KT
      WRITE(6,2000) JZHT
 2000 FORMAT(1X,I2)
  100 CONTINUE
      STOP
      END
```

ΕΝΟΤΗΤΑ Β.

Παρατήρηση: Η ενότητα αυτή περιέχει δύσκολα προγράμματα για εξάσκηση όσων επιθυμούν να ειδικευτούν στο μάθημα.

Άσκηση 4.7

Στα πλαίσια ενός ερευνητικού προγράμματος μετρήθηκαν σε 500 άτομα (άντρες και γυναίκες) το ύψος και το βάρος. Τα αποτελέσματα γράφονται σε αρχείο (κάρτες), μια γραμμή για κάθε άτομο με την εξής μορφή: ITYP(I),H(I),W(I) με FORMAT(I5,F10.0,F10.0) όπου ITYP(I) είναι 7 για άντρες και 8 για γυναίκες και H(I),W(I) το ύψος σε εκατοστά και το βάρος σε κιλά αντίστοιχα. Να συνταχθεί πρόγραμμα που να διαβάζει το αρχείο και να βρίσκει (και εκτυπώνει) τα ακόλουθα:

- τον αριθμό των γυναικών NWOM και των ανδρών NMAN
- το μέσο βάρος όλων των ατόμων AVW
- το μέσο ύψος όλων των ατόμων AVH
- το μέσο βάρος των γυναικών (AVWOM)
- το μέσο βάρος των ανδρών (AVMAN)
- πόσες γυναίκες LWOM έχουν βάρος πάνω από το μέσο βάρος
- πόσοι άντρες LMAN έχουν ύψος κάτω από το μέσο ύψος.

Λύση:

```
      DIMENSION ITYP(500),H(500),W(500)
      READ (5,1000) (ITYP(N),H(N),W(N),N=1,500,1)
 1000 FORMAT(I5,2F10.0)
      DO 100 N=1,500,1
      IF (ITYP(N).EQ.7) GOTO 50
      AVWOM=AVWOM+W(N)
      NWOM=NWOM+1
   50 AVH=AVH+H(N)
      AVW=AVW+W(N)
```

```
 100 CONTINUE
     NMAN=500-NWOM
     AVMAN=AVW-AVWOM
     AVW=AVW/500.
     X=NWOM
     AVWOM=AVWOM/X
     X=NMAN
     AVMAN=AVMAN/X
     AVH=AVH/500.
     DO 200 L=1,500,1
     IF (ITYP(L).EQ.8.AND.W(L).GT.AVWOM) LWOM=LWOM+1
     IF (ITYP(L).EQ.7.AND.H(L).LT.AVH) LMAN=LMAN+1
 200 CONTINUE
     WRITE(6,1100) NWOM,NMAN
1100 FORMAT(1HO,I3,1X,I3)
     WRITE(6,1200) AVH,AVW
1200 FORMAT(1X,F10.4,/,1X,F10.4)
     WRITE(6,1300) AVWOM,AVMAN
1300 FORMAT(2(1X,F10.4))
     WRITE(6,1100) LWOM,LMAN
     STOP
     END
```

Άσκηση 4.8

Σε κυκλική διατομή υποστυλώματος από οπλισμένο σκυρόδεμα πρέπει να τοποθετηθεί ένας αριθμός χαλύβδινων ράβδων περιμετρικά σε ακτίνα R (cm). Δίνεται ότι ο αριθμός των ράβδων πρέπει να είναι τουλάχιστον 6, να έχουν όλες την ίδια διάμετρο και να ισαπέχουν περιμετρικά μεταξύ των. Η ελάχιστη περιμετρική απόσταση των κέντρων είναι 4cm. Διατιθέμενες διατομές ράβδων είναι 14,16,18,20, 22 cm. Συνολικό εμβαδόν ράβδων τουλάχιστο A cm^2 . Να συνταχθεί πρόγραμμα που να κάνει τα ακόλουθα:

- Διάβασμα δεδομένων (R,A) με format της επιλογής σας
- εύρεση του απαιτούμενου αριθμού ράβδων NBAR(I) και το συνολικό τους εμβαδόν AREA(I) για κάθε μια από τις διατιθέμενες διαμέτρους.

- Έλεγχος της ελάχιστης περιμετρικής απόστασης SPACE(I) των διαδοχικών ράβδων
- υπολογισμός της DIF(I)=AREA(I)-A για κάθε μια από τις διαμέτρους
- προσδιορισμός της οικονομικότερης λύσης και εκτύπωση του απαιτούμενου αριθμού των αντίστοιχων ράβδων.

Λύση:

```
      DIMENSION NBAR(5),AREA(5),SPACE(5),DIF(5)
      READ(5,1000) R,A
 1000 FORMAT(2(1X,F10.4))
      PI=3.141593
      N=1
      DO 100 I=14,22,2
      E=PI*(I/2)**2
      NBAR(N)=A/E
      NBAR(N)=NBAR(N)+1
      IF (NBAR(N).LT.6) NBAR(N)=6
      X=NBAR(N)
      AREA(N)=X*E
      N=N+1
  100 CONTINUE
      PER=2*PI*R
      DO 200 I=1,5,1
      X=NBAR(I)
      SPACE(I)=PER/X
      IF (SPACE(I).LT.4.) SPACE(I)=-1.
      DIF(I)=AREA(I)-A
  200 CONTINUE
      AMIN=-1.
      DO 300 J=1,5,1
      IF (SPACE(J).EQ.-1.) GOTO 300
      AMIN=DIF(J)
  300 CONTINUE
      IF (AMIN.EQ.-1.) STOP
```

```
      DO 400 K=1,5,1
      IF (SPACE(K).LT.0.) GOTO 400
      IF (AMIN.GT.DIF(K)) AMIN=DIF(K)
  400 CONTINUE
      WRITE(6,2000) AMIN
 2000 FORMAT(1H0,'Kalyterh lysh=',F10.4)
      STOP
      END
```

Άσκηση 4.9

Δίνεται διδιάστατο μητρώο IC μεγέθους NxN όπου N ≤ 2000. Γράψτε ένα πρόγραμμα που να βρίσκει για τα στοιχεία των δυο κύριων διαγωνίων τα ακόλουθα:

- το άθροισμα των τιμών των IABD
- όσα από αυτά έχουν τιμές που διαιρούνται ακριβώς από το IABD
- τους δείκτες τους
- τους δείκτες του στοιχείου με τη μεγαλύτερη τιμή
- είναι το στοιχείο αυτό πρώτος αριθμός;

Λύση:

```
      DIMENSION IC(200,200),ID(2,200)
      READ(5,1000) N
 1000 FORMAT(I4)
      READ(5,1500) ((IC(I,J),I=1,N,1),J=1,N,1)
 1500 FORMAT(I6)
      IABD=0
      DO 100 M=1,N,1
      IABD=IABD+IC(M,M)
      J=N+1-M
      IABD=IABD+IC(M,J)
  100 CONTINUE
      L=1
      DO 200 M=1,N,1
      IF (IC(M,M)/IABD*IABD.NE.IC(M,M)) GOTO 150
      ID(1,L)=M
```

```
      ID(2,L)=M
      L=L+1
  150 IF(IC(M,N+1-M)/IABD*IABD.NE.IC(M,N+1-M)) GOTO 200
      ID(1,L)=M
      ID(2,L)=N+1-M
      L=L+1
  200 CONTINUE
      WRITE(6,2000) IABD
 2000 FORMAT(1H1,'Athroisma=',I15)
      IF (L.EQ.1) GOTO 400
      DO 300 MC=1,L-1,1
      WRITE(6,2500) ID(1,MC),ID(2,MC)
 2500 FORMAT(1X,'x=',I4,3X,'y=',I4)
  300 CONTINUE
      IMIN=IC(1,1)
      IX=1
      IY=1
      DO 320 MC=1,L-1,1
      IF (IMIN.GT.IC(ID(1,MC),ID(2,MC))) GOTO 320
      IMIN=IC(ID(1,MC),ID(2,MC))
      IX=ID(1,MC)
      IY=ID(2,MC)
  320 CONTINUE
      WRITE(6,3000) IMIN
 3000 FORMAT(1X,'Megalyterh timh=',I10)
      WRITE(6,2500) IX,IY
      SQ=IMIN**.5
      ISQ=SQ
      DO 350 JK=2,SQ,1
      IF (IMIN/JK*JK.EQ.IMIN) GOTO 400
  350 CONTINUE
      WRITE(6,3500) IMIN
 3500 FORMAT(10X,'Nai, einai prwtos o ',I10)
  400 STOP
```

```
      END
```

Άσκηση 4.10

Δίνεται διδιάστατο μητρώο IVEC μεγέθους N * N ,όπου N<=1000 και μερικά από τα στοιχεία του μπορεί να είναι ίσα. Γράψτε ένα πρόγραμμα που να βρίσκει την τιμή του στοιχείου επί των δύο κυρίων διαγωνίων που επαναλαμβάνεται τις περισσότερες φορές και να εκτυπώνει αυτήν την τιμή καθώς και το ζεύγος των δεικτών του στοιχείου που αντιστοιχεί στην πρώτη θέση που συναντάται αυτή η τιμή στις δύο κύριες διαγώνιες, αρχίζοντας από την αριστερή.

Λύση:

Για την λύση του παραπάνω προβλήματος, είναι απαραίτητο να μεταφέρουμε τα στοιχεία των δύο κυρίων διαγωνίων του IVEC σε έναν νέο μονοδιάστατο πίνακα IDIA, έτσι ώστε να μπορούμε να τα επεξεργαστούμε καλύτερα. Ο πίνακας IDIA είναι διαστάσεως 2N. Οι πρώτες N θέσεις καταλαμβάνονται από την "αριστερή" (κύρια) διαγώνιο, ενώ οι θέσεις από N+1 έως 2N καταλαμβάνονται από την "δεξιά" (δευτερεύουσα) διαγώνιο. Στη συνέχεια βρίσκουμε το στοιχείο που επαναλαμβάνεται τις περισσότερες φορές (δουλεύοντας στον νέο πίνακα που φτιάξαμε), και την πρώτη θέση που εμφανίζεται.

Η θέση είναι εύκολο να βρεθεί, με βάση την θέση που βρέθηκε πρώτη φορά το στοιχείο που επαναλαμβάνεται τις περισσότερες φορές στον πίνακα IDIA. Τέλος προχωρούμε στην εκτύπωση των αποτελεσμάτων.

```
DIMENSION IVEC(500,500),IDIA(1000)
   10 READ(5,1000) N
 1000 FORMAT(1X,I4)
      IF(N.GT.500.OR.N.LT.2) GOTO 10
      READ(5,1000) ((IVEC(I,J),J=1,N),I=1,N)
      DO 100 I=1,N,1
      IDIA(I)=IVEC(I,I)
      M=N+1-I
      IDIA(N+I)=IVEC(I,M)
```

```
 100 CONTINUE
     MINDEX=0
     L=2*N
     DO 200 I=1,L,1
     INDEX=0
     DO 300 J=I,L,1
     IF(IDIA(I).EQ.IDIA(J)) INDEX=INDEX+1
 300 CONTINUE
     IF(MINDEX.GE.INDEX) GOTO 200
     MINDEX=INDEX
     K=IDIA(I)
     KR=I
     KC=I
     IF(I.GT.N) GOTO 200
     KR=I-N
     KC=N+1-KR
 200 CONTINUE
     WRITE(6,2000) K,KR,KC
2000 FORMAT(1X,3(I10,5X))
     STOP
     END
```

Άσκηση 4.11

Δίνεται μονοδιάστατο μητρώο ITEST στο οποίο εισάγουμε θετικά ακέραια στοιχεία με format I10, διατρημένα ένα σε κάθε κάρτα. Με την βοήθεια κάρτας της επιλογής σας να γίνεται η λήξη της εισαγωγής.

- Βρείτε και τυπώστε τον μέσο όρο IAVER των στοιχείων, στρογγυλεμένο στην πλησιέστερη μονάδα.
- Βρείτε και τυπώστε τον μέσο όρο ISAVER του πρώτου και του τελευταίου στοιχείου, στρογγυλεμένο προς τα πάνω.
- Για όσα στοιχεία είναι ανάμεσα στα IAVER,ISAVER βρείτε και τυπώστε το πλήθος τους καθώς και τους δείκτες τους

- Από τα παραπάνω στοιχεία, βρείτε και τυπώστε πόσα είναι άρτια καθώς και τους δείκτες τους.
- Να θέσετε τα δεδομένα του ITEST στον πίνακα ISER σε αύξουσα σειρά.

Λύση:

Το πρόβλημα δεν παρουσιάζει ιδιαίτερη δυσκολία. Πρέπει όμως να σημειωθούν τα εξής:

Η στρογγύλευση στην πλησιέστερη μονάδα γίνεται αν βρεθεί το δεκαδικό μέρος της διαίρεσης [άθροισμα]/[πλήθος]. Αν αυτό είναι μεγαλύτερο ή ίσο του 0.5 τότε στην ακέραια διαίρεση (όπου παίρνουμε το ακέραιο μέρος) απλά προσθέτουμε 1. Αν είναι μικρότερο απλά αφήνουμε τον μέσο όρο ίσο με το πηλίκο των ακεραίων [άθροισμα]/[πλήθος].

Η στρογγύλευση προς τα πάνω γίνεται ως εξής: Επειδή η διαίρεση θα γίνει με το 2 (έχω πλήθος αθροιζόμενων στοιχείων δύο), αρκεί να ελέγξω αν το άθροισμα είναι ακέραιο πολλαπλάσιο του 2 (αν είναι άρτιος). Αν είναι, τότε ως μέσος όρος λαμβάνεται το ακέραιο πηλίκο [άθροισμα]/2. Σε οποιαδήποτε άλλη περίπτωση, η (πραγματική) διαίρεση θα αφήνει πηλίκο με δεκαδικό μέρος, οπότε επειδή στρογγυλεύουμε προς τα άνω, αρκεί να προσθέσουμε 1 στο (ακέραιο)πηλίκο της διαίρεσης [άθροισμα]/2.

Για να βρούμε ποια στοιχεία του ITEST είναι ανάμεσα στα IAVER,ISAVER, δημιουργούμε για κάθε στοιχείο το γινόμενο ITES= (στοιχείο - IAVER)* (στοιχείο - ISAVER).Αυτό είναι αυστηρώς αρνητικό μόνο όταν το στοιχείο είναι ανάμεσα στα IAVER, ISAVER **ανεξάρτητα από το ποιο από τα IAVER,ISAVER είναι μεγαλύτερο.**

Τέλος μεταφέρουμε πρώτα τα στοιχεία του ITEST στον ISER, και στην συνέχεια επεξεργαζόμαστε τα στοιχεία του ISER, θέτοντάς τα σε αύξουσα σειρά. Έτσι έχουμε:

```
      DIMENSION ITEST(1000),MA(1000),ISER(1000),MB(1000)

C     ΚΑΤΑΧΩΡΗΣΗ ΣΤΟΙΧΕΙΩΝ ITEST ΚΑΙ ΥΠΟΛΟΓΙΣΜΟΣ ΑΘΡΟΙΣΜΑΤΟΣ

      IAVER=0
      IA=1
```

```
   10 READ(5,1000) ITEST(IA)
 1000 FORMAT(I10)
      IF(ITEST(IA).EQ.0) GOTO 15
      IAVER=IAVER + ITEST(IA)
      IA=IA+1
      GOTO 10

C     Η ΜΗΔΕΝΙΚΗ ΚΑΡΤΑ ΔΕΝ ΜΕΤΡΑΕΙ!

   15 N=IA-1

C     ΥΠΟΛΟΓΙΣΜΟΣ ΔΕΚΑΔΙΚΟΥ ΜΕΡΟΥΣ.ΑΝ ΕΙΝΑΙ ΜΕΓΑΛΥΤΕΡΟ Η ΙΣΟ
C     ΑΠΟ ΤΟ 0.5 ΤΟΤΕ ΣΤΟΝ ΜΕΣΟ ΟΡΟ(ΑΚΕΡΑΙΟ ΜΕΡΟΣ) IAVER
C     ΠΡΟΣΘΕΤΟΥΜΕ 1

      AVER=IAVER
      XN=N
      AVER=AVER/XN
      IAVER=IAVER/N
      TAVER=IAVER
      TEST=AVER-TAVER
      IF(TEST.GE.0.5) IAVER=IAVER+1

C     ΒΡΙΣΚΟΥΜΕ ΤΟ ΑΘΡΟΙΣΜΑ ΠΡΩΤΟΥ ΚΑΙ ΤΕΛΕΥΤΑΙΟΥ ΟΡΟΥ.ΑΝ
C     ΕΙΝΑΙ ΑΡΤΙΟΣ ΤΟΤΕ Η ΔΙΑΙΡΕΣΗ ΜΕ ΤΟ 2 ΔΙΝΕΙ ΑΚΕΡΑΙΟ,ΤΟΝ
C     ΟΠΟΙΟ ΠΕΡΝΟΥΜΕ ΩΣ ΜΕΣΟ ΟΡΟ.ΣΕ ΚΑΘΕ ΑΛΛΗ ΠΕΡΙΠΤΩΣΗ ΣΤΟ
C     ΜΕΣΟ ΟΡΟ (ΑΚΕΡΑΙΟ ΜΕΡΟΣ) ΠΡΟΣΘΕΤΟΥΜΕ 1,ΔΙΟΤΙ ΠΡΕΠΕΙ ΝΑ
C     ΣΤΡΟΓΓΥΛΟΠΟΙΗΣΟΥΜΕ ΠΡΟΣ ΤΑ ΑΝΩ.

      IS=ITEST(1)+ITEST(N)
      ISAVER=IS/2
      IF((IS/2)*2.EQ.IS) GOTO 20
      ISAVER=ISAVER+1
```

```
C     ΕΚΤΥΠΩΣΗ IAVER,ISAVER

   20 WRITE(6,700) IAVER,ISAVER
  700 FORMAT(1X,'IAVER=',I4,3X,'ISAVER=',I4)

C     ΕΝΑ ΣΤΟΙΧΕΙΟ ΤΟΥ ITEST ΕΙΝΑΙ ΑΝΑΜΕΣΑ ΣΤΑ IAVER,ISAVER
C     (ΑΝΕΞΑΡΤΗΤΑ ΑΠΟ ΤΗΝ ΜΕΤΑΞΥ ΤΟΥΣ ΑΝΙΣΟΤΙΚΗ
C     ΣΧΕΣΗ,ΑΝΕΞΑΡΤΗΤΑ ΔΗΛΑΔΗ ΜΕ ΤΟ ΠΟΙΟ ΑΠΟ ΤΑ
C     IAVER,ISAVER ΕΙΝΑΙ ΜΕΓΑΛΥΤΕΡΟ) ΑΝ ΚΑΙ ΜΟΝΟ ΑΝ:
C     (ΣΤΟΙΧΕΙΟ -  IAVER)*(ΣΤΟΙΧΕΙΟ-ISAVER) < 0!!!!
C     ΚΑΤΑΧΩΡΟΥΜΕ ΣΤΟΝ ΠΙΝΑΚΑ ΜΑ ΤΟΥΣ ΔΕΙΚΤΕΣ ΤΩΝ ΣΤΟΙΧΕΙΩΝ
C     ΠΟΥ ΒΡΙΣΚΟΝΤΑΙ ΑΝΑΜΕΣΑ ΣΤΑ IAVER,ISAVER

      IC=0
      DO 25 IB=1,N,1
      ITES1=ITEST(IB)-IAVER
      ITES2=ITEST(IB)-ISAVER
      ITES=ITES1*ITES2
      IF(ITES.GE.0) GOTO 25
      IC=IC+1
      MA(IC)=IB
   25 CONTINUE

C     ΕΚΤΥΠΩΣΗ ΤΗΣ ΠΑΡΑΠΑΝΩ ΔΙΑΔΙΚΑΣΙΑΣ

      WRITE(6,2500) IC
      IF(IC.EQ.0) GOTO 75
      DO 70 ID=1,IC,1
      WRITE(6,2000) MA(ID)
 2000 FORMAT(1X,I3)
 2500 FORMAT(1X,'PLH8OS= ',2X,I3)
   70 CONTINUE

C     ΤΩΡΑ ΕΛΕΓΧΟΥΜΕ ΠΟΙΑ ΑΠΟ ΤΑ ΣΤΟΙΧΕΙΑ ΜΕ ΤΟΥΣ ΔΕΙΚΤΕΣ
```

```
C     ΑΠΟ ΤΟ ΜΑ ΑΝΤΙΣΤΟΙΧΟΥΝ ΣΕ ΑΡΤΙΑ ΣΤΟΙΧΕΙΑ ΤΟΥ ITEST.

   75 IE=0
      DO 80 IF=1,IC,1
      K=ITEST(MA(IF))
      IF((K/2)*2.NE.K) GOTO 80
      IE=IE+1
      MB(IE)=MA(IF)
   80 CONTINUE

C     ΕΚΤΥΠΩΣΗ

      WRITE(6,2500) IE
      IF(IE.EQ.0) GOTO 90
      DO 85 IG=1,IE,1
      WRITE(6,2000) MB(IG)
   85 CONTINUE

C     ΜΕΤΑΦΟΡΑ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ITEST ΣΤΟΝ ISER ΚΑΙ
C     ΤΟΠΟΘΕΤΗΣΗ ΑΥΤΩΝ ΣΕ ΑΥΞΟΥΣΑ ΣΕΙΡΑ.ΕΚΤΥΠΩΣΗ.

   90 DO 95 IH=1,N,1
      ISER(IH)=ITEST(IH)
   95 CONTINUE
      DO 105 II=1,N,1
      DO 100 IJ=II,N,1
      IF(ISER(II).LE.ISER(IJ)) GOTO 100
      IK=ISER(II)
      ISER(II)=ISER(IJ)
      ISER(IJ)=IK
  100 CONTINUE
  105 CONTINUE
      DO 110 IL=1,N,1
      WRITE(6,3000) ISER(IL)
```

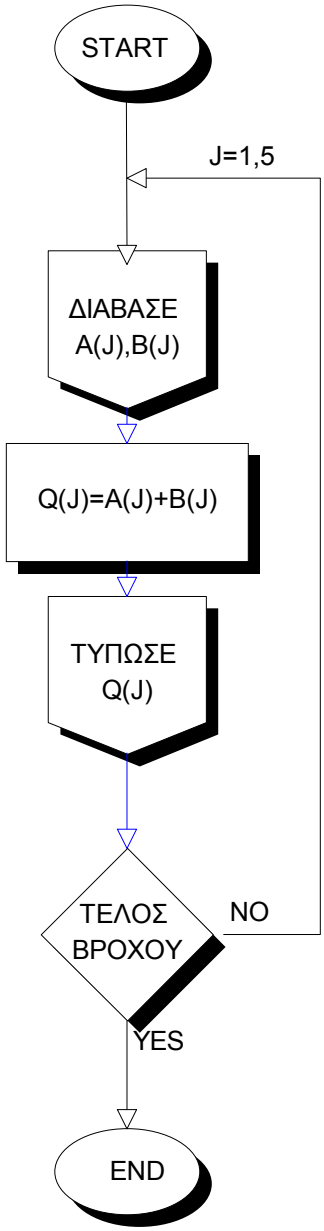
```
3000 FORMAT(1X,I10)
 110 CONTINUE

     STOP
     END
```

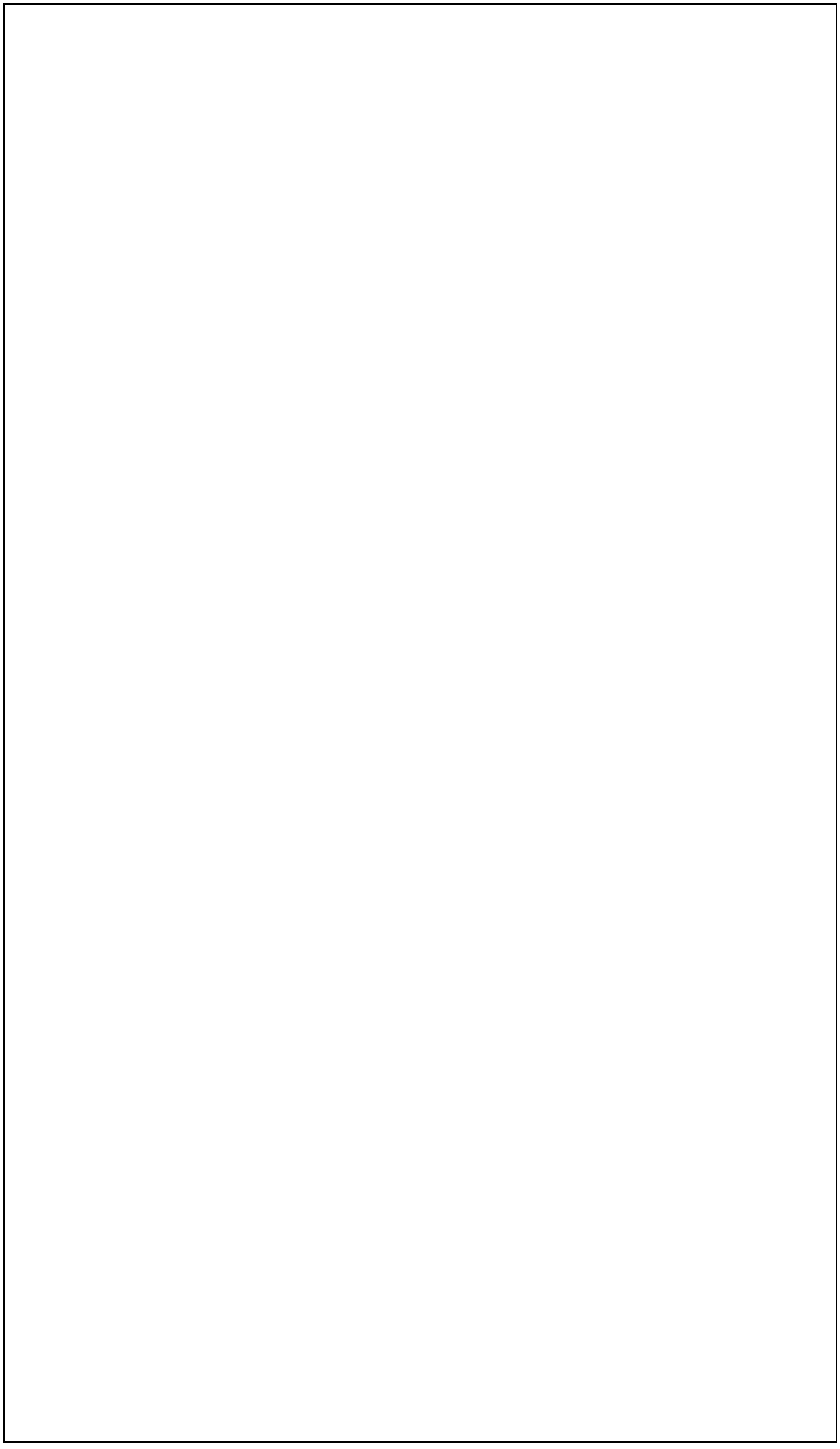
Ο έλεγχος για το ποια από τα στοιχεία του ITEST είναι ανάμεσα στα IAVER, ISAVER και ποια από αυτά είναι άρτιοι θα μπορούσε να γίνει στον ίδιο βρόχο ως εξής:

```
     IC=0
     IE=0
     DO 25 IB=1,N,1
     ITES1=ITEST(IB)-IAVER
     ITES2=ITEST(IB)-ISAVER
     ITES=ITES1*ITES2
     IF(ITES.GE.0) GOTO 25
     IC=IC+1
     MA(IC)=IB
     K=(ITEST(IB)/2)*2
     IF(K.NE.ITEST(IB)) GOTO 25
     IE=IE+1
     MB(IE)=IB
  25 CONTINUE
```

και στην συνέχεια θα προχωρούσαμε απευθείας στην εκτύπωση των αποτελεσμάτων.

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΒΛΗΜΑΤΟΣ 3.3





ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΒΛΗΜΑΤΟΣ 4.7