ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΤΗΣ ΕΠΙΣΤΗΜΗΣ ΤΩΝ ΥΛΙΚΩΝ

ΥΠΟΛΟΓΙΣΤΕΣ Ι
Σημειώσεις εργαστηρίου

Ελευθέριος Λοιδωρίκης
Δημήτριος Γ. Παπαγεωργίου

ΙΩΑΝΝΙΝΑ 2010

Περιεχόμενα

Προς τα τέλη του 1953 ο John Backus πρότεινε στους προϊσταμένους του στην IBM, ότι θα ήταν χρήσιμο μια μικρή ομάδα ερευνητών να αναλάβει να αναπτύξει ένα πιο αποδοτικό και οικονομικό τρόπο προγραμματισμού του υπολογιστή 704 αντί της γλώσσας μηχανής που χρησιμοποιούνταν μέχρι τότε. Η πρόταση έγινε δεκτή και η ομάδα άρχισε δουλειά αμέσως. Στα μέσα του 1954 είχε φτιαχτεί μια αρχική πρόταση για μια γλώσσα προγραμματισμού με σημαντικές δυνατότητες και ευελιξία. Η γλώσσα αυτή ονομάστηκε "the IBM Mathematical FORmula TRANslation system, FORTRAN". Το πρώτο "Εγχειρίδιο Αναφοράς" για τη γλώσσα εκδόθηκε τον Οκτώβριο του 1956 και ο πρώτος μεταφραστής δόθηκε σε πελάτες τον Απρίλιο του 1957. Δώδεκα μήνες μετά ακολούθησε η Fortran II, μια βελτιωμένη έκδοση με αυξημένες δυνατότητες διαγνωστικών και ένα σημαντικό αριθμό επεκτάσεων στη γλώσσα. Παρά τις αρχικές αντιρρήσεις ότι τα μεταφρασμένα προγράμματα δεν ήταν τόσο αποδοτικά όσα αυτά που γράφονταν απευθείας σε γλώσσα μηχανής η γλώσσα προχώρησε και μέχρι το 1960 η IBM είχε εκδώσει μεταφραστές για πολλά από τα μοντέλα υπολογιστών που κατασκεύαζε. Η σημαντικότερη εξέλιξη όμως ήταν ότι και άλλοι κατασκευαστές άρχισαν να γράφουν μεταφραστές Fortran. Μέχρι το 1963 υπήρχαν πάνω από 40 διαφορετικοί μεταφραστές Fortran.

Παρόλα αυτά ένα σημαντικό πρόβλημα ήταν ότι η αρχική Fortran (και η Fortran II) χρησιμοποιούσαν ορισμένα συγκεκριμένα χαρακτηριστικά από το σύνολο εντολών του υπολογιστή 704 και γι αυτό και οι υπόλοιποι μεταφραστές Fortran έτειναν να κάνουν το ίδιο. Επιπλέον την εποχή εκείνη δεν είχαν κατανοηθεί πλήρως τα πλεονεκτήματα μιας γλώσσας ανεξάρτητης από συγκεκριμένους υπολογιστές με αποτέλεσμα να υπάρχουν ασυμβατότητες στους μεταφραστές ακόμη και από τον ίδιο κατασκευαστή. Ως αποτέλεσμα πίεσης από τους χρήστες η IBM το 1961 ξεκίνησε την ανάπτυξη μιας βελτιωμένης Fortran η οποία δεν θα είχε χαρακτηριστικά εξαρτώμενα από συγκεκριμένες μηχανές. Ο μεταφραστής για τη νέα αυτή έκδοση (Fortan IV) κατασκευάστηκε αρχικά για τον IBM 7030 το 1962 και αργότερα για τους IBM 7090/7094. Η πιο σημαντική όμως εξέλιξη ήταν η απόφαση του American Standards Association το Μάιο του 1962 να ορίσει μια επιτροπή που θα καθόριζε ένα πρότυπο της γλώσσας. Η επιτροπή όρισε δύο γλώσσες, τη Fortran βασισμένη στο μεγαλύτερο βαθμό στην Fortran IV της IBM και την Basic Fortran βασισμένη στη Fortran II. Τα πρότυπα αυτά επικυρώθηκαν το Μάρτιο του 1966. Η ύπαρξη ενός επίσημου προτύπου (το οποίο στην πράξη ήταν και διεθνές πρότυπο) σήμαινε ότι υπήρχε μια καλά ορισμένη βάση για την περαιτέρω ανάπτυξη της γλώσσας.

Η δεκαετία του 1970 είδε τους υπολογιστές να εδραιώνονται σε όλους τους τομείς της ανθρώπινης δραστηριότητας. Αυτό μεταξύ άλλων οδήγησε σε ανάπτυξη και πολλών άλλων διαφορετικών γλωσσών προγραμματισμού. Πολλές από αυτές προσανατολίζονταν σε συγκεκριμένες εφαρμογές, αλλά οι περισσότερες ήταν γλώσσες"γενικής χρήσης". Οι πιο γνωστές από αυτές ήταν οι Algol, Basic, Cobol, Pascal και PL/I. Μεταξύ όλης της έρευνας και ανάπτυξης στο πεδίο των γλωσσών προγραμματισμού η Fortran δεν παρέμεινε στάσιμη. Οι κατασκευαστές υπολογιστών κατασκεύασαν μεταφραστές που δέχονταν ένα σημαντικό αριθμό επεκτάσεων από το πρότυπο, ενώ το 1969 το American National Standards Institute όρισε μια επιτροπή για την αναθεώρηση του προτύπου του 1966. Εν μέρει λόγω των πολλών αλλαγών στην φιλοσοφία και πρακτική του προγραμματισμού αυτή την περίοδο, η πρώτη πρόχειρη έκδοση του νέου προτύπου δεν εμφανίστηκε παρά μετά από επτά χρόνια. Κατά τη διάρκεια του 1977 η πρόχειρη αυτή έκδοση έγινε αντικείμενο διεθνούς συζήτησης και μια βελτιωμένη έκδοση έγινε δεκτή ως νέο πρότυπο τον Απρίλιο του 1978. Το νέο αυτό πρότυπο αντικαθιστά τα παλαιότερα και για να ξεχωρίζει από αυτά, αναφέρεται ως Fortran-77.

Κεφάλαιο 1

Πως γράφεται ένα πρόγραμμα Fortran

1.1 Μορφή του προγράμματος

Οι επόμενοι κανόνες συνοψίζουν τον τρόπο γραφής των προγραμμάτων Fortran-77.

1. Οι εντολές του προγράμματος γράφονται στις θέσεις 7 έως και 72 κάθε γραμμής. Συνεπώς πριν γράψουμε μια εντολή πρέπει να αφήσουμε τουλάχιστον 6 κενές θέσεις. Στην πράξη αυτό γίνεται με ένα πάτημα του πλήκτρου TAB. Οτιδήποτε μετά τη θέση 72 κάθε γραμμής, αγνοείται.

2. Σε περίπτωση που μια εντολή υπερβαίνει τη θέση 72, τότε μπορούμε να τη συνεχίσουμε στην επόμενη γραμμή τοποθετώντας ένα οποιοδήποτε σύμβολο στη θέση 6.

3. Οι εντολές Fortran γράφονται με κεφαλαίους ή μικρούς χαρακτήρες χωρίς ο μεταφραστής να ξεχωρίζει τη διαφορά.

4. Στις θέσεις 1 έως και 5 κάθε γραμμής όταν απαιτείται τοποθετούνται "αριθμοί γραμμών". Πρόκειται για ακέραιους αριθμούς από το 1 μέχρι και το 99999 που τοποθετούνται στην αρχή της γραμμής και χρησιμεύουν ως "ετικέτες".

5. Σε οποιοδήποτε σημείο του προγράμματος μπορούν να τοποθετηθούν σχόλια. Κάθε γραμμή που περιέχει σχόλια έχει στην πρώτη θέση το χαρακτήρα `C` ή `*`.

Παράδειγμα:

Το παρακάτω πρόγραμμα θα εμφανίσει στην οθόνη τη λέξη `Hello`.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM FIRST`	Κάθε πρόγραμμα ξεκινάει με την εντολή PROGRAM η οποία ακολουθείται από ένα συμβολικό όνομα που δίνουμε στο πρόγραμμα.
`C Αυτό είναι ένα απλό υπόδειγμα` `C προγράμματος.`	Οι γραμμές αυτές αποτελούν σχόλια που δεν απευθύνονται στον μεταφραστή Fortran, αλλά σε όποιον διαβάσει το πρόγραμμα.
`WRITE (*,*) 'Hello'`	Με την εντολή WRITE εμφανίζεται στην οθόνη ένα μήνυμα. Σημειώστε ότι το μήνυμα πρέπει να βρίσκεται εντός απλών αποστρόφων.
`END`	Κάθε πρόγραμμα τελειώνει με την εντολή END.

1.2 Συμβολικά ονόματα

Το συμβολικό όνομα που γράφουμε δίπλα στην εντολή PROGRAM επιλέγεται από εμάς. Γενικά τα συμβολικά ονόματα επιλέγονται έτσι ώστε να ακολουθούν τους παρακάτω κανόνες:

1. Κάθε συμβολικό όνομα αποτελείται από τουλάχιστον 1 χαρακτήρα. Αν και το πρότυπο της γλώσσας Fortran περιορίζει το μέγιστο μήκος κάθε συμβολικού ονόματος στους 6 χαρακτήρες, οι σύγχρονοι μεταφραστές συνήθως επιτρέπουν πολύ περισσότερους χαρακτήρες.
2. Μπορούν να χρησιμοποιηθούν τα γράμματα του Αγγλικού (όχι του Ελληνικού) αλφαβήτου A..Z και τα αριθμητικά ψηφία 0..9. Οι περισσότεροι μεταφραστές Fortran επιτρέπουν τη χρήση και ορισμένων άλλων χαρακτήρων όπως _.
3. Κεφαλαία και μικρά γράμματα θεωρούνται ίδια. Πχ. τα συμβολικά ονόματα `FIRST`, `First` και `first` θεωρούνται ίδια.
4. Ο πρώτος χαρακτήρας ενός συμβολικού ονόματος πρέπει να είναι γράμμα.

Ορισμένα παραδείγματα συμβολικών ονομάτων φαίνονται στον παρακάτω πίνακα:

Συμβολικό όνομα	*Σωστό ή Λάθος*
SMURF	Σωστό
B3	Σωστό
1EPS	Λάθος διότι ξεκινάει με αριθμό
ΠΡΩΤΟ	Λάθος διότι περιέχει Ελληνικούς χαρακτήρες
GREAT$	Λάθος διότι περιέχει το χαρακτήρα $
GOODDAY	Έχει πάνω από έξι χαρακτήρες, αλλά οι περισσότεροι μεταφραστές το επιτρέπουν

Παράδειγμα:

Για να εμφανιστούν στην οθόνη τα προσωπικά σας στοιχεία με την εξής μορφή:

```
+----------------+
| ΑΠΟΣΤΟΛΟΣ      |
| ΑΠΟΣΤΟΛΟΠΟΥΛΟΣ |
| ΑΜ: 954        |
+----------------+
```

θα πρέπει να χρησιμοποιήσετε τον παρακάτω κώδικα Fortran:

```
      PROGRAM MYNAME
      WRITE (*,*) '+----------------+'
      WRITE (*,*) '| ΑΠΟΣΤΟΛΟΣ      |'
      WRITE (*,*) '| ΑΠΟΣΤΟΛΟΠΟΥΛΟΣ |'
      WRITE (*,*) '| ΑΜ: 954        |'
      WRITE (*,*) '+----------------+'
      END
```

Κεφάλαιο 2

Τύποι δεδομένων, μεταβλητές και είσοδος–έξοδος

Οι παρακάτω τύποι δεδομένων υποστηρίζονται από τη Fortran:

1. Ακέραιοι αριθμοί (INTEGER)
2. Πραγματικοί αριθμοί (REAL)
3. Πραγματικοί αριθμοί διπλής ακρίβειας (DOUBLE PRECISION)
4. Χαρακτήρες (CHARACTER)
5. Μιγαδικοί αριθμοί (COMPLEX)
6. Λογικές τιμές – Αληθές/Ψευδές (LOGICAL)

Από τους ανωτέρω τύπους θα εξετάσουμε μόνο τους τρεις πρώτους.

2.1 Ακέραιοι αριθμοί

Πρόκειται για αριθμούς που έχουν μόνο ακέραιο μέρος και όχι δεκαδικά ψηφία. Για το λόγο αυτό η αναπαράστασή τους στον HY είναι πάντα ακριβής. Η αποθήκευσή τους απαιτεί 4 bytes μνήμης, κατά συνέπεια μπορεί να λάβουν τιμές στην περιοχή -2 147 483 648 έως 2 147 483 647.

Μεταξύ δύο ακεραίων αριθμών μπορούν να γίνουν οι πράξεις πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση και ύψωση σε δύναμη χρησιμοποιώντας τα σύμβολα `+ - * /` και `**` αντίστοιχα.

Όλες οι πράξεις μεταξύ ακεραίων δίνουν ακέραιο αποτέλεσμα. Ειδικότερα, **το αποτέλεσμα της διαίρεσης μεταξύ δύο ακέραιων αριθμών είναι πάντα ακέραιος**, αφού αν προκύψουν δεκαδικά ψηφία αυτά αποκόπτονται. Προσοχή: δεν γίνεται στρογγύλευση του αποτελέσματος, απλώς αποκοπή των δεκαδικών ψηφίων. Αυτή η συγκεκριμένη λειτουργία ονομάζεται *ακέραια διαίρεση*. Ορισμένα παραδείγματα φαίνονται στον παρακάτω πίνακα:

Πράξη	*Αποτέλεσμα*	
`3+5`	8	
`8-4`	4	
`2*6`	12	
`9/3`	3	
`7/2`	3	(Ακέραια διαίρεση)
`25/4`	6	(Ακέραια διαίρεση)
`4**2`	16	

2.2 Πραγματικοί αριθμοί

Ονομάζονται και αριθμοί κινητής υποδιαστολής. Πρόκειται για αριθμούς οι οποίοι έχουν ακέραιο μέρος και δεκαδικά ψηφία. Για την αποθήκευσή τους απαιτούνται 4 bytes (32 bits) μνήμης.
Παραδείγματα πραγματικών αριθμών: 1.27 -761.841 5.0
Σημειώστε ότι ακόμη και αν το δεκαδικό μέρος είναι μηδέν (όπως στον 5.0), ο αριθμός εξακολουθεί να θεωρείται πραγματικός και όχι ακέραιος. Επιπλέον ένας πραγματικός αριθμός μπορεί να γραφεί στην λεγόμενη "επιστημονική αναπαράσταση", δηλαδή χρησιμοποιώντας δυνάμεις του 10. Πχ. ο αριθμός 325.476 μπορεί να γραφεί ως 3.25476E2 που σημαίνει 3.25476 x 10^2. Η επιστημονική αναπαράσταση εξυπηρετεί για την αναπαράσταση πολύ μικρών ή πολύ μεγάλων αριθμών. Ορισμένα παραδείγματα αριθμών στην επιστημονική αναπαράσταση φαίνονται παρακάτω:

Επιστημονική αναπαράσταση	*Αριθμός*
`1.727E5`	172700.0
`-32.476E-2`	-0.32476
`8.45E0`	8.45

Στην πραγματικότητα ένας πραγματικός αριθμός αποθηκεύεται στη μνήμη στην επιστημονική αναπαράσταση (όμως χρησιμοποιώντας δυνάμεις του 2 και όχι του 10). Συγκεκριμένα απαιτείται 1 bit για το πρόσημο του αριθμού, 8 bits για τον εκθέτη (μαζί με το πρόσημό του) και 23 bits για τα σημαντικά ψηφία. Για το λόγο αυτό το πλήθος των σημαντικών ψηφίων που μπορούν να αποθηκευτούν είναι περασμένο (περίπου 7). Για παράδειγμα με την πράξη 2.0/3.0 η οποία στην πραγματικότητα δίνει τον αριθμό 0.6666..., το αποτέλεσμα που θα πάρουμε στον υπολογιστή είναι 0.6666667. Παρατηρείστε ότι το τελευταίο σημαντικό ψηφίο στρογγυλεύεται. Για τον ίδιο λόγο η περιοχή τιμών που μπορεί να πάρει ένας πραγματικός αριθμός είναι περίπου από 10^{-38} έως 10^{38}. Επίσης για τους αρνητικούς από -10^{38} έως -10^{-38}. Όταν από μια πράξη προκύψει αριθμός μεγαλύτερος (κατ' απόλυτη τιμή) από 10^{38} αυτό αποτελεί λογικό λάθος (υπερπλήρωση–overflow) και η λειτουργία του προγράμματός μας σταματά. Παρατηρήστε ότι υπάρχει μια περιοχή μεταξύ -10^{-38} και 10^{-38} όπου οι πραγματικοί αριθμοί δεν μπορούν να πάρουν τιμές πλην του αριθμού 0. Όλοι οι υπόλοιποι αριθμοί σε αυτή την περιοχή δεν μπορούν να αναπαρασταθούν στον υπολογιστή και αν από μια πράξη προκύψει πραγματικός αριθμός σε αυτή την περιοχή η κατάσταση ονομάζεται υποπλήρωση (underflow).

Οι πράξεις που μπορεί να γίνουν μεταξύ δύο πραγματικών αριθμών είναι πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση και ύψωση σε δύναμη χρησιμοποιώντας τα σύμβολα `+ - * /` και `**` αντίστοιχα.

Οποιαδήποτε πράξη μεταξύ δύο πραγματικών αριθμών δίνει ως αποτέλεσμα και πάλι ένα πραγματικό αριθμό.

2.3 Πραγματικοί αριθμοί διπλής ακρίβειας

Πρόκειται για πραγματικούς αριθμούς που καταλαμβάνουν 8 bytes στη μνήμη (διπλάσιο χώρο από τους πραγματικούς απλής ακρίβειας REAL). Από τα 64 bits ενός πραγματικού αριθμού διπλής ακρίβειας, 1 bit καταλαμβάνει το πρόσημο του αριθμού, 11 bits ο εκθέτης και 52 bits τα σημαντικά ψηφία. Σαν συνέπεια η περιοχή αριθμών που αναπαρίσταται με αυτό τον τύπο είναι κατ' απόλυτη τιμή από 10^{-308} έως 10^{308}. Επιπλέον η ακρίβειά τους είναι περίπου 15 σημαντικά ψηφία.

Για να δηλώσουμε ότι για έναν πραγματικό αριθμό θα χρησιμοποιηθεί ο τύπος DOUBLE PRECISION πρέπει υποχρεωτικά να τον γράψουμε στην επιστημονική αναπαράσταση όπου το σύμβολο E αντικαθίσταται από το D. Ορισμένα παραδείγματα φαίνονται στον παρακάτω πίνακα:

Αν γράψουμε	*Σημαίνει*	*Ο τύπος του αριθμού είναι*
2.95E3	2950.0	REAL
-652.12E-4	-0.065212	REAL
3.15D2	315.0	DOUBLE PRECISION
951.43	951.43	REAL
85.91D0	85.91	DOUBLE PRECISION

Οι πράξεις που μπορεί να γίνουν μεταξύ δύο πραγματικών αριθμών διπλής ακρίβειας είναι πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση και ύψωση σε δύναμη χρησιμοποιώντας τα σύμβολα `+ - * /` και `**` αντίστοιχα. Οποιαδήποτε πράξη μεταξύ δύο πραγματικών αριθμών διπλής ακρίβειας δίνει ως αποτέλεσμα και πάλι ένα πραγματικό αριθμό διπλής ακρίβειας.

2.4 Αριθμητικές παραστάσεις

Χρησιμοποιώντας τις πέντε απλές πράξεις (`+ - * / **`) μεταξύ αριθμών, μπορούμε να κατασκευάσουμε πιο σύνθετες παραστάσεις. Πχ `2*3 + 7*(14 - 2**4/2)`

Για να υπολογιστεί το αποτέλεσμα μιας τέτοιας σύνθετης παράστασης χρησιμοποιούμε τους παρακάτω κανόνες:

1. Η σειρά των πράξεων καθορίζεται από την προτεραιότητά τους σύμφωνα με τον παρακάτω πίνακα:

Πράξη	*Προτεραιότητα*
( )	Υψηλή
**	
* ή /	
+ ή -	Χαμηλή

2. Μεταξύ πράξεων με την ίδια προτεραιότητα οι πράξεις γίνονται από αριστερά προς τα δεξιά. Πχ. στην παράσταση 2*6/3 όπου ο πολλαπλασιασμός και η διαίρεση έχουν ίδια προτεραιότητα πρώτα θα γίνει η πράξη 2*6 και κατόπιν το αποτέλεσμα θα διαιρεθεί με το 3.

3. Ο ανωτέρω κανόνας δεν ισχύει στην περίπτωση διαδοχικών υψώσεων σε δύναμη όπου οι πράξεις γίνονται από δεξιά προς τα αριστερά. Πχ. στην παράσταση 2**3**4 πρώτα θα γίνει η πράξη 3**4 και το αποτέλεσμα θα χρησιμοποιηθεί ως εκθέτης του 2. Δηλαδή η παράσταση αυτή σημαίνει $2^{(3^4)}$ και όχι $(2^3)^4$.

4. Ζεύγη παρενθέσεων μπορούν να χρησιμοποιηθούν για να αλλάξουν τη σειρά των πράξεων. Πχ. στην παράσταση (3+5)*4 παρότι ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα έναντι της πρόσθεσης, θα γίνει πρώτα η πράξη που βρίσκεται εντός των παρενθέσεων και κατόπιν το αποτέλεσμα θα πολλαπλασιαστεί με 4.

5. Όπως ήδη έχει αναφερθεί πράξεις μεταξύ ακεραίων δίνουν πάντα ακέραιο αποτέλεσμα, πράξεις μεταξύ πραγματικών δίνουν πραγματικό αποτέλεσμα, ενώ πράξεις μεταξύ πραγματικών αριθμών διπλής ακρίβειας δίνουν πάντα ως αποτέλεσμα αριθμό διπλής ακρίβειας. Στην περίπτωση που σε μια πράξη υπάρχουν δύο διαφορετικοί τύποι δεδομένων τότε πριν γίνει η πράξη ο κατώτερος τύπος προάγεται στον ανώτερο, ενώ το αποτέλεσμα είναι πάντα του ανώτερου τύπου. Ο τύπος DOUBLE PRECISION θεωρείται ανώτερος του REAL, ο οποίος με τη σειρά του θεωρείται ανώτερος του INTEGER.

   Παράδειγμα:
   Για να υπολογιστεί η παράσταση 4+7.2 ο ακέραιος αριθμός 4 προάγεται στον πραγματικό 4.0 ενώ το αποτέλεσμα (11.2) είναι πραγματικός αριθμός. Επίσης για να υπολογιστεί η παράσταση 8/3.2 ο ακέραιος αριθμός 8 προάγεται στον πραγματικό αριθμό 8.0 ενώ το αποτέλεσμα (2.5) είναι πραγματικός αριθμός.

Παράδειγμα:

Θεωρήστε την παράσταση: `2*3 + 7*(14 - 2**4/2)`. Η σειρά των πράξεων έχει ως εξής:

	Πράξη	*Μερικό αποτέλεσμα*
	Αρχική παράσταση	`2*3 + 7*(14 - 2**4/2)`
1	Πρώτα θα γίνουν οι πράξεις εντός των παρενθέσεων. Συγκεκριμένα η ύψωση σε δύναμη έχει μεγαλύτερη προτεραιότητα, άρα η πράξη 2**4 θα γίνει πρώτη	`2*3 + 7*(14 - 16/2)`
2	Κατόπιν θα γίνει η διαίρεση 16/2	`2*3 + 7*(14 - 8)`
3	Ακολουθεί η αφαίρεση 14-8, οπότε ολοκληρώνονται οι πράξεις εντός των παρενθέσεων.	`2*3 + 7*6`
4	Από τις πράξεις που απομένουν ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα. Επειδή υπάρχουν δύο πολλαπλασιασμοί, θα γίνει πρώτα αυτός που είναι πιο αριστερά, δηλαδή 2*3	`6 + 7*6`

5	Κατόπιν γίνεται ο επόμενος πολλαπλασιασμός 7*6	6 + 42
6	Τέλος γίνεται η πρόσθεση δίνοντας το τελικό αποτέλεσμα	48

2.5 Εσωτερικές συναρτήσεις

Η Fortran έχει ενσωματωμένο ένα σημαντικό αριθμό από συναρτήσεις γνωστές ως *εσωτερικές συναρτήσεις*. Οι συναρτήσεις αυτές είναι διαθέσιμες σε όποιο πρόγραμμα τις χρειαστεί και δεν χρειάζεται να δηλωθούν ή να οριστούν με κάποιο τρόπο. Οι περισσότερες συναρτήσεις έχουν ένα γενικό όνομα, δηλαδή **μπορούν να χρησιμοποιηθούν με ορίσματα διαφορετικού τύπου, επιστρέφοντας διαφορετικού τύπου αποτελέσματα**. Υπάρχει και ένας μικρότερος αριθμός συναρτήσεων οι οποίες δέχονται ορίσματα ενός μόνο τύπου.

Τέλος για λόγους συμβατότητας με παλαιότερες εκδόσεις υπάρχουν και τα ειδικά ονόματα των γενικών συναρτήσεων (δεν περιγράφονται εδώ). Έτσι για παράδειγμα η γενική συνάρτηση ABS που μπορεί να χρησιμοποιηθεί με ορίσματα τύπου INTEGER, REAL και DOUBLE PRECISION έχει τα ειδικά ονόματα IABS, ABS, και DABS αντίστοιχα.

Στον παρακάτω πίνακα παρουσιάζονται οι συνηθέστερα χρησιμοποιούμενες συναρτήσεις. Ο τύπος των ορισμάτων καθορίζεται από τα επόμενα σύμβολα ως εξής: I = INTEGER, R = REAL, D = DOUBLE PRECISION.

Συνάρτηση	*Τύπος αποτελέσματος*	*Τι επιστρέφει*
ABS(IRD)	Όπως το όρισμα	Απόλυτη τιμή |IRD|
ACOS(RD)	Όπως το όρισμα	Τόξο συνημιτόνου, $\arccos(RD)$
ASIN(RD)	Όπως το όρισμα	Τόξο ημιτόνου, $\arcsin(RD)$
ATAN(RD)	Όπως το όρισμα	Τόξο εφαπτομένης, $\arctan(RD)$
COS(RD)	Όπως το όρισμα	$cos(RD)$
DBLE(IR)	DOUBLE PRECISION	Μετατροπή του ορίσματος σε διπλή ακρίβεια
EXP(RD)	Όπως το όρισμα	e^{RD}
INT(RD)	INTEGER	Ακέραιο μέρος του ορίσματος
LOG(RD)	Όπως το όρισμα	Φυσικός λογάριθμος $\ln(RD)$
LOG10(RD)	Όπως το όρισμα	Δεκαδικός λογάριθμος $\log_{10}(RD)$
MAX(IRD1,IRD2,...)	Όπως το όρισμα	Η μεγαλύτερη τιμή από τις IRD1, IRD2, ...
MIN(IRD1,IRD2,...)	Όπως το όρισμα	Η μικρότερη τιμή από τις IRD1, IRD2, ...
MOD(IRD1,IRD2)	Όπως το όρισμα	Υπόλοιπο διαίρεσης του IRD1 από τον IRD2
NINT(RD)	INTEGER	Ο πλησιέστερος ακέραιος
REAL(ID)	REAL	Μετατροπή του ορίσματος σε απλή ακρίβεια
SIN(RD)	Όπως το όρισμα	$\sin(RD)$
SQRT(RD)	Όπως το όρισμα	$\sqrt{RD}$
TAN(RD)	Όπως το όρισμα	$\tan(RD)$

Ορισμένες παρατηρήσεις για τις συναρτήσεις και τον τρόπο που χρησιμοποιούνται:

1. Αρκετές συναρτήσεις δεν δέχονται ως όρισμα ακέραιο αριθμό. Έτσι για να βρούμε την τετραγωνική ρίζα του 2 πρέπει να γράψουμε SQRT(2.) και όχι SQRT(2) που αποτελεί συντακτικό λάθος.

2. Οι τριγωνομετρικές συναρτήσεις SIN, COS και TAN δέχονται τα ορίσματά τους σε ακτίνια και όχι μοίρες. Έτσι αν γράψουμε SIN(20.) θα υπολογιστεί το ημίτονο των 20 ακτινίων και όχι των 20 μοιρών.

3. Επίσης οι αντίστροφες τριγωνομετρικές συναρτήσεις ASIN, ACOS και ATAN επιστρέφουν το αποτέλεσμά τους σε ακτίνια και όχι μοίρες.

4. Τα ορίσματα κάθε συνάρτησης πρέπει να είναι εντός του πεδίου ορισμού της συνάρτησης. Για παράδειγμα δεν μπορούμε να ζητήσουμε την τετραγωνική ρίζα αρνητικού αριθμού, πράγμα που αποτελεί λογικό σφάλμα. Σε μια τέτοια περίπτωση το πρόγραμμα θα σταματήσει στο σημείο αυτό εμφανίζοντας ένα μήνυμα λάθους.

5. Εάν χρειάζεται να υπολογίσουμε τη βάση των φυσικών λογαρίθμων e (≈ 2.71828) γράφουμε EXP(1.).

6. Για να υπολογίσουμε τον αριθμό π (≈ 3.14159) γράφουμε ACOS(-1.).

2.6 Παραδείγματα αριθμητικών παραστάσεων

Ο παρακάτω πίνακας δείχνει ορισμένες αριθμητικές παραστάσεις και το αποτέλεσμά τους.

Παράσταση	*Αποτέλεσμα*
`1 + 8/3*(2+4/2**2)`	7
`2*(6**3/3-(5/6+2)**2+4)`	144
`(4/(7-4)*2+1)**(2/3*4+1)`	3

Παρακάτω φαίνονται ορισμένες αλγεβρικές παραστάσεις και ο τρόπος κωδικοποίησής τους σε Fortran. Σημειώστε ότι ανάλογα με την αλγεβρική παράσταση μπορεί να υπάρχει παραπάνω από ένας τρόπος για να γραφεί σε Fortran.

Αλγεβρική παράσταση	*Παράσταση σε Fortran*
3^{2+6}	`3**(2+6)`
$3^{-4} \cdot 8$	`3**(-4)*8`
$\left(\frac{25-7}{9}\right)^2$	`((25-7)/9)**2`
$\ln 2 - \frac{\sqrt{8}}{9}$	`LOG(2.)-SQRT(8.)/9`
$6[2 + (8\lvert\sqrt{11} - 6^2\rvert + 1)]$	`6*(2+(8*ABS(SQRT(11.)-6**2)+1))`
$\sqrt{10 - e^2} + \pi$	`SQRT(10-EXP(2.))+ACOS(-1.)`
$\sqrt[3]{8}$	`8**(1.0/3.0)`

2.7 Μεταβλητές

Μεταβλητή είναι ένα συμβολικό όνομα που δίνεται σε κάποια θέση μνήμης του υπολογιστή για να αποθηκευτεί ένας αριθμός (ή άλλος τύπος δεδομένων). Τα ονόματα των μεταβλητών επιλέγονται από τον προγραμματιστή, και πρέπει να ακολουθούν τους κανόνες για τα συμβολικά ονόματα της παραγράφου 1.2.

Δεδομένου ότι η Fortran μπορεί να χειριστεί διάφορους τύπους δεδομένων κάθε μεταβλητή έχει ένα συγκεκριμένο τύπο, δηλαδή σε κάθε μεταβλητή μπορεί να αποθηκευτεί ένας μόνο τύπος δεδομένων. Η Fortran χρησιμοποιεί ένα προκαθορισμένο κανόνα σύμφωνα με τον οποίο θεωρεί ως ακέραιες μεταβλητές όσες το όνομά τους ξεκινάει με τα γράμματα I, J, K, L, M και N, ενώ όλες οι υπόλοιπες θεωρούνται πραγματικές απλής ακρίβειας. Εάν θέλουμε να έχουμε μεταβλητές άλλων τύπων αυτές πρέπει να δηλωθούν αμέσως μετά την εντολή PROGRAM. Στο παρόν κείμενο δεν θα χρησιμοποιήσουμε τον προκαθορισμένο κανόνα της Fortran, αλλά θα δηλώνουμε όσες μεταβλητές χρειαζόμαστε. Οι δηλώσεις των μεταβλητών γίνονται στην αρχή του προγράμματος, πριν από οποιαδήποτε εκτελέσιμη εντολή.

Παράδειγμα: Σε ένα πρόγραμμα απαιτούνται οι ακέραιες μεταβλητές N, KSYM και PANEL, οι πραγματικές μεταβλητές T και XMIN και οι μεταβλητές διπλής ακρίβειας FUN και R. Το αντίστοιχο απόσπασμα προγράμματος θα ήταν ως εξής:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM EXAMPL`	
`IMPLICIT NONE`	Καταργεί τον προκαθορισμένο κανόνα της Fortran για τους τύπους των μεταβλητών
`INTEGER N, KSYM, PANEL`	Δηλώνονται οι ακέραιες μεταβλητές
`REAL T, XMIN`	Δηλώνονται οι πραγματικές μεταβλητές
`DOUBLE PRECISION FUN, R`	Δηλώνονται οι μεταβλητές διπλής ακρίβειας
⋮	

Σημειώστε ότι η εντολή IMPLICIT NONE η οποία καταργεί τον προκαθορισμένο κανόνα της Fortran για τους τύπους των μεταβλητών, δεν περιλαμβάνεται στο πρότυπο της Fortran-77, αλλά αποτελεί μια συνηθισμένη επέκτασή της την οποία υλοποιούν σχεδόν όλοι οι σύγχρονοι μεταφραστές.

Χρησιμοποιώντας μεταβλητές μπορούμε να κάνουμε αριθμητικές πράξεις και να κατασκευάσουμε σύνθετες παραστάσεις ακριβώς όπως περιγράφηκε στις προηγούμενες παραγράφους.

2.8 Εντολή ανάθεσης τιμής

Για να δώσουμε τιμή σε μεταβλητή χρησιμοποιούμε την εντολή ανάθεσης τιμής. Η εντολή αυτή έχει τη μορφή:

$$\textit{μεταβλητή} = \textit{αριθμητική παράσταση}$$

Στο αριστερό μέρος του = βρίσκεται πάντα το όνομα μιας μεταβλητής (που θα πρέπει ήδη να έχουμε δηλώσει στην αρχή του προγράμματος). Στο δεξί μέρος της εντολής βρίσκεται πάντα μια αριθμητική παράσταση.

Η εντολή λειτουργεί ως εξής: Πρώτα υπολογίζεται η τιμή της αριθμητικής παράστασης στο δεξί μέρος και κατόπιν το αποτέλεσμα που θα προκύψει ανατίθεται ως τιμή της μεταβλητής που βρίσκεται στο αριστερό μέρος της εντολής.

Παράδειγμα:

```
SUM = A+B
```

Εδώ πρώτα θα υπολογιστεί το άθροισμα A+B και κατόπιν το αποτέλεσμα θα ανατεθεί ως τιμή της μεταβλητής SUM.

Άλλα παραδείγματα:

```
K = 1
D = SQRT(B**2-4*A*C)
ERM = (XM-XS)/2
M = M+1
```

Ειδικότερα στο τελευταίο παράδειγμα υπολογίζεται πρώτα η παράσταση M+1 και κατόπιν το αποτέλεσμα ανατίθεται και πάλι ως τιμή της μεταβλητής M αντικαθιστώντας την προηγούμενη τιμή. Έτσι η εντολή αυτή έχει ως αποτέλεσμα να αυξηθεί κατά 1 η τιμή της μεταβλητής M.

2.9 Εντολές εισόδου–εξόδου

2.9.1 Εντολή WRITE

Όπως είδαμε για να εμφανιστεί ένα μήνυμα στην οθόνη χρησιμοποιούμε την εντολή WRITE ως εξής:

`WRITE (*,*) '`*μήνυμα*`'`

Με την ανωτέρω εντολή εμφανίζεται στην οθόνη οτιδήποτε βρίσκεται εντός των απλών αποστρόφων. Όμως με την εντολή WRITE μπορούμε επίσης να εμφανίσουμε στην οθόνη την τιμή κάποιας μεταβλητής. Στην περίπτωση αυτή η εντολή WRITE έχει τη μορφή:

`WRITE (*,*)` *μεταβλητή*

Προσέξτε ότι η *μεταβλητή* δεν τοποθετείται εντός αποστρόφων. Πολλαπλές μεταβλητές και μηνύματα μπορούν να εμφανιστούν στην οθόνη από μια μόνο εντολή WRITE. Πχ.

```
WRITE (*,*) 'Το μήκος είναι ',S,' ενώ το εμβαδόν είναι ',E
```

Τα μηνύματα ή οι μεταβλητές γράφονται δίπλα από την εντολή WRITE χωρισμένα με κόμματα. Ότι βρίσκεται εντός αποστρόφων θεωρείται μήνυμα και εμφανίζεται στην οθόνη ως έχει (συμπεριλαμβανομένων και τυχόν κενών), ενώ τα υπόλοιπα θεωρούνται μεταβλητές (ή γενικότερα αριθμητικές παραστάσεις) και εμφανίζεται η αντίστοιχη τιμή.

Παραδείγματα

```
WRITE (*,*) X, Y, Z
WRITE (*,*) PI*R**2
WRITE (*,*) 'Η ταχύτητα είναι ', A1*V1+A2*V2
```

Με τις ανωτέρω εντολές WRITE ο τρόπος που εμφανίζεται ο κάθε αριθμός στην οθόνη είναι προκαθορισμένος και δυστυχώς διαφέρει από μεταφραστή σε μεταφραστή. Για παράδειγμα ο αριθμός 8.53 θα εμφανιστεί ως 8.53 από κάποιους μεταφραστές ή ως 8.530000 από κάποιους άλλους. Για να ελέγξουμε τον τρόπο εμφάνισης των αριθμών χρησιμοποιούμε την εντολή FORMAT. Η εντολή FORMAT δεν είναι εκτελέσιμη, αλλά με τη χρήση της μπορούμε να αλλάξουμε τον τρόπο που εμφανίζονται οι αριθμοί (δηλαδή πόσους χαρακτήρες θα καταλάβει ο αριθμός σε μια γραμμή της οθόνης, πόσα δεκαδικά ψηφία θα εμφανιστούν κα.). Η εντολή FORMAT έχει τη μορφή:

ΑΓ `FORMAT (` *π1*`,` *π2*`,` *π3*`, ...)`

ΑΓ είναι ένας αριθμός γραμμής, δηλαδή ένας ακέραιος αριθμός από το 1 έως το 99999, μοναδικός εντός του προγράμματος, που χρησιμεύει σαν ετικέτα στην εντολή FORMAT. Ο αριθμός αυτός δείχνει στην εντολή WRITE ότι πρέπει να χρησιμοποιηθεί η αντίστοιχη εντολή μορφοποίησης και γράφεται στη θέση του δεύτερου αστερίσκου της εντολής WRITE. Τα σύμβολα *π1*, *π2*, *π3*, ... ονομάζονται *πεδία μορφοποίησης* και καθορίζουν τον τρόπο μορφοποίησης των αριθμών.

Παράδειγμα:

Για να εμφανίσουμε την οθόνη μία ακέραια μεταβλητή μπορούμε να χρησιμοποιήσουμε τις εντολές:

```
      WRITE (*,10) K
10    FORMAT (I6)
```

Επιλέξαμε (αυθαίρετα) τον αριθμό 10 ως αριθμό γραμμής, ενώ το πεδίο διαμόρφωσης `I6` δείχνει ότι θα εμφανιστεί στην οθόνη ένας ακέραιος (η μεταβλητή K) καταλαμβάνοντας 6 θέσεις. Ο αριθμός εμφανίζεται με δεξιά στοίχιση, δηλαδή αν το πλήθος των ψηφίων του αριθμού είναι μικρότερο από 6, μένουν κενά μπροστά από τον αριθμό.

Ορισμένα από τα πιο συχνά χρησιμοποιούμενα πεδία μορφοποίησης φαίνονται στον παρακάτω πίνακα:

Πεδίο μορφοποίησης	*Τι εμφανίζεται*
`In`	Ακέραιος αριθμός που καταλαμβάνει `n` θέσεις.
`Fw.d`	Πραγματικός αριθμός που καταλαμβάνει `n` θέσεις εκ των οποίων `d` είναι για δεκαδικά ψηφία.
`Ew.d`	Πραγματικός αριθμός στην επιστημονική αναπαράσταση που καταλαμβάνει `n` θέσεις εκ των οποίων `d` είναι για τα σημαντικά ψηφία.
`nX`	Μένουν `n` κενές θέσεις.

Παράδειγμα:

```
      WRITE (*,20) X
20    FORMAT (F20.12)
```

Εδώ η πραγματική μεταβλητή X θα εμφανιστεί στην οθόνη καταλαμβάνοντας συνολικά 20 θέσεις, εκ των οποίων οι 12 θα είναι δεκαδικά ψηφία. Εάν ο αριθμός έχει λιγότερα δεκαδικά ψηφία, αυτά θα συμπληρωθούν με μηδενικά, ενώ αν δεν έχει αρκετά ακέραια ψηφία έτσι ώστε να καταλάβει το συνολικό εύρος των 20 θέσεων, θα μείνουν κενά μπροστά από τον αριθμό.

Παράδειγμα:

```
      WRITE (*,30) X
30    FORMAT (E20.8)
```

Εδώ η πραγματική μεταβλητή X θα εμφανιστεί στην οθόνη στην επιστημονική αναπαράσταση, καταλαμβάνοντας συνολικά 20 θέσεις, εκ των οποίων οι 8 θα είναι δεκαδικά ψηφία.

Παράδειγμα:

```
      WRITE (*,30) K
30    FORMAT (8X,I6)
```

Εδώ η ακέραια μεταβλητή (K) θα εμφανιστεί στην οθόνη καταλαμβάνοντας 6 θέσεις αφού όμως πρώτα μείνουν 8 κενές θέσεις.

Παράδειγμα:

Θεωρήστε το παρακάτω απόσπασμα προγράμματος:

```
      REAL  X, Y
      INTEGER K, M
          ⋮
      K = 176
      M = -32
      X = 81.4
      Y = 0.0027
      WRITE (*,10) K
      WRITE (*,10) M
      WRITE (*,20) X
      WRITE (*,30) Y
      WRITE (*,40) K, X, Y
10    FORMAT (I6)
20    FORMAT (F10.4)
30    FORMAT (E15.4) Y
40    FORMAT (1X, I4, 1X, F6.2, E12.4 )
```

Τα αποτελέσματα που θα εμφανιστούν στην οθόνη φαίνονται παρακάτω (η πρώτη γραμμή δεν εμφανίζεται αλλά χρησιμεύει για την αρίθμηση των θέσεων):

```
 12345678901234567890123456789012345678901234  <== Θέσεις στην οθόνη.
    176
    -32
    81.4000
      0.2700E-02
   176   81.40   0.2700E-02
```

2.9.2 Εντολή READ

Με την εντολή READ μπορούμε να εισάγουμε από το πληκτρολόγιο την τιμή μιας μεταβλητής κατά την εκτέλεση ενός προγράμματος. Η εντολή συντάσσεται ως εξής:

`READ (*,*)` *μ1*, *μ2*, *μ3*, ...

όπου *μ1*, *μ2*, *μ3*, ... είναι μεταβλητές.

Παράδειγμα:

```
READ (*,*) X, Y
```

Η εντολή λειτουργεί ως εξής: Μόλις το πρόγραμμα φτάσει στην εντολή READ σταματά και περιμένει από το χρήστη να πληκτρολογήσει δύο αριθμούς. Ότι πληκτρολογήσει ο χρήστης ανατίθεται ως τιμή στις αντίστοιχες μεταβλητές (X,Y) και το πρόγραμμα συνεχίζει. Είναι σύνηθες η εντολή READ να χρησιμοποιείται μαζί με μια εντολή WRITE που εμφανίζει στην οθόνη κάποιο μήνυμα σχετικά με το τι πρέπει να πληκτρολογήσει ο χρήστης.

Παράδειγμα:

```
WRITE (*,*) 'Εισάγετε τις συντεταγμένες X και Y'
READ (*,*) X, Y
```

Εάν ο χρήστης πληκτρολογήσει λιγότερα νούμερα και πατήσει enter, το πρόγραμμα εξακολουθεί να περιμένει τα υπόλοιπα. Εάν ο χρήστης εισάγει περισσότερα νούμερα από όσα απαιτούνται τότε το πρόγραμμα διαβάζει αυτά που χρειάζεται και τα υπόλοιπα αγνοούνται.

2.10 Παραδείγματα

2.10.1 Εμβαδόν κύκλου

Η ακτίνα ενός κύκλου είναι 25cm. Κατασκευάστε πρόγραμμα το οποίο να υπολογίζει το εμβαδόν και την περίμετρό του. Χρησιμοποιείστε μεταβλητές διπλής ακρίβειας.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM CIRCLE`	
`IMPLICIT NONE`	Καταργεί τον προκαθορισμένο κανόνα της Fortran για τους τύπους των μεταβλητών.
`DOUBLE PRECISION R, EMB, PER`	Δηλώνονται οι απαραίτητες μεταβλητές. R είναι η ακτίνα του κύκλου, EMB το εμβαδόν και PER η περίμετρος του κύκλου.
`DOUBLE PRECISION PI`	Δηλώνεται η μεταβλητή PI στην οποία θα αποθηκευτεί η τιμή του αριθμού π.
`C Το πρόγραμμα αυτό υπολογίζει το`	Σχόλιο.
`C εμβαδόν και την περίμετρο κύκλου.`	Σχόλιο.
`R = 25.`	Ανατίθεται τιμή στην ακτίνα του κύκλου.
`PI = ACOS(-1.)`	Υπολογίζεται η τιμή του αριθμού π.
`EMB = PI*R**2`	Υπολογίζεται το εμβαδόν και ανατίθεται στη μεταβλητή EMB.
`PER = 2*PI*R`	Υπολογίζεται η περίμετρος και ανατίθεται στη μεταβλητή PER.
`WRITE (*,*) 'Εμβαδόν = ', EMB`	Εμφανίζεται το εμβαδόν στην οθόνη.
`WRITE (*,*) 'Περίμετρος = ', PER`	Εμφανίζεται η περίμετρος στην οθόνη.
`END`	

2.10.2 Ευθύγραμμη κίνηση

Ένα σώμα εκτελεί διαδοχικά δύο είδη κίνησης. Αρχικά εκτελεί ευθύγραμμη ομαλή κίνηση με σταθερή ταχύτητα v για χρόνο t_1. Κατόπιν εκτελεί ευθύγραμμη επιταχυνόμενη κίνηση με επιτάχυνση a για χρόνο t_2. Κατασκευάστε πρόγραμμα που θα υπολογίζει το συνολικό διάστημα που διένυσε. *Σημείωση:* Κατά την ομαλή κίνηση το διάστημα που διένυσε είναι $s_1 = vt_1$. Κατά την επιταχυνόμενη κίνηση το διάστημα που διένυσε είναι $s_2 = vt_2 + \frac{1}{2}at_2^2$.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM MOVE`	
`IMPLICIT NONE`	
`DOUBLE PRECISION V, A, T1, T2`	V είναι η αρχική ταχύτητα, A η επιτάχυνση και T1, T2 οι χρόνοι για τα δύο είδη κίνησης.
`DOUBLE PRECISION S1, S2, S`	S1 το διάστημα κατά την ευθύγραμμη κίνηση, S2 κατά την επιταχυνόμενη και S το ολικό διάστημα.

`WRITE (*,*) 'Εισάγετε τα V,A,T1,T2'`	Εμφάνιση μηνύματος στην οθόνη.
`READ (*,*) V, A, T1, T2`	Εισάγονται από το πληκτρολόγιο οι τιμές των μεταβλητών V, A, T1 και T2.
`S1 = V*T1`	Υπολογίζεται το διάστημα κατά την ομαλή κίνηση.
`S2 = V*T2 + A*T2**2/2`	Υπολογίζεται το διάστημα κατά την επιταχυνόμενη κίνηση.
`S = S1+S2`	Υπολογίζεται το ολικό διάστημα.
`WRITE (*,*) 'Διάστημα = ', S`	Εμφανίζεται στην οθόνη το ολικό διάστημα.
`END`	

Κεφάλαιο 3

Εντολή IF

3.1 Πως συντάσσεται

Η εντολή IF επιτρέπει να γίνονται συγκρίσεις μεταξύ μεταβλητών και ανάλογα με το αποτέλεσμα να εκτελείται ή όχι κάποιο τμήμα προγράμματος. Η εντολή συντάσσεται ως εξής:

```
IF ( σύγκριση ) THEN
   εντολή1
   εντολή2
     ⋮
END IF
```

Η μορφή αυτή της εντολής ονομάζεται τμηματικό IF (block if). Η *σύγκριση* είναι μια σύγκριση μεταξύ δύο μεταβλητών ή γενικότερα μεταξύ δύο παραστάσεων, ενώ *εντολή1*, *εντολή2*, ... είναι οποιαδήποτε εκτελέσιμη εντολή Fortran. Οι δυνατές συγκρίσεις που μπορεί να γίνουν είναι οι παρακάτω:

Σύγκριση	*Σημαίνει*
`A .EQ. B`	$A = B$
`A .NE. B`	$A \neq B$
`A .LT. B`	$A < B$
`A .LE. B`	$A \leq B$
`A .GT. B`	$A > B$
`A .GE. B`	$A \geq B$

Το αποτέλεσμα κάθε σύγκρισης είναι είτε αληθές (ισχύει) είτε ψευδές (δεν ισχύει).

Η ανωτέρω εντολή IF λειτουργεί ως εξής: Αρχικά πραγματοποιείται η *σύγκριση*. Αν το αποτέλεσμά της είναι αληθές τότε οι *εντολή1*, *εντολή2*, ... που βρίσκονται εντός του τμήματος IF–ENDIF εκτελούνται διαδοχικά μέχρι να φτάσουμε στο ENDIF. Κατόπιν ο εκτελούνται οι υπόλοιπες εντολές του προγράμματος που βρίσκονται κάτω από το ENDIF. Αν το αποτέλεσμα της σύγκρισης είναι ψευδές,

οι *εντολή1*, *εντολή2*, ...δεν εκτελούνται (παρακάμπτονται) και η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που βρίσκεται κάτω από το ENDIF. Σε ορισμένες απλές περιπτώσεις όπου μόνο μία εντολή βρίσκεται εντός του IF–ENDIF, μπορούμε να χρησιμοποιήσουμε την απλοποιημένη σύνταξη:

```
IF ( σύγκριση ) εντολή
```

Η μορφή αυτή ονομάζεται λογικό IF και είναι ισοδύναμη με:

```
IF ( σύγκριση ) THEN
   εντολή
END IF
```

Στην περίπτωση που θέλουμε να εκτελέσουμε ένα τμήμα κώδικα όταν η *σύγκριση* είναι αληθής και ένα άλλο όταν είναι ψευδής, τότε η εντολή IF έχει την μορφή:

```
IF ( σύγκριση ) THEN
   εντολή
   εντολή
    ⋮
ELSE
   εντολή
   εντολή
    ⋮
END IF
```

Με αυτή τη σύνταξη όταν η *σύγκριση* είναι αληθής εκτελούνται οι εντολές που βρίσκονται εντός του τμήματος IF–ELSE, ενώ όταν η *σύγκριση* είναι ψευδής εκτελούνται οι εντολές εντός του τμήματος ELSE–ENDIF. Τέλος η πιο γενική μορφή της εντολής IF, όπου μπορούν να γίνουν πολλαπλές συγκρίσεις είναι η παρακάτω:

```
IF ( σύγκριση1 ) THEN
   εντολή
   εντολή
    ⋮
ELSE IF ( σύγκριση2 ) THEN
   εντολή
   εντολή
    ⋮
ELSE IF ( σύγκριση3 ) THEN
   εντολή
   εντολή
    ⋮
ELSE
   εντολή
   εντολή
    ⋮
```

```
END IF
```

Με αυτή τη σύνταξη, πρώτα ελέγχεται η *σύγκριση1* και αν είναι αληθής εκτελούνται οι εντολές που την ακολουθούν, μέχρι το επόμενο ELSE IF. Κατόπιν το πρόγραμμα συνεχίζει με τις εντολές που ακολουθούν το END IF. Αν η *σύγκριση1* είναι ψευδής τότε ελέγχεται η *σύγκριση2* και αν βρεθεί αληθής εκτελούνται οι εντολές που την ακολουθούν μέχρι το επόμενο ELSE IF. Γενικά μπορεί να υπάρχουν πολλά τμήματα ELSE IF. Μόλις βρεθεί μια σύγκριση με αληθές αποτέλεσμα εκτελούνται οι εντολές του αντίστοιχου τμήματος. Αν καμία ELSE IF δεν δώσει αληθές αποτέλεσμα, τότε εκτελούνται οι εντολές που βρίσκονται στο τμήμα ELSE.

Ορισμένες παρατηρήσεις για την εντολή IF.

1. Μια εντολή IF μπορεί να έχει πολλαπλά τμήματα ELSE IF, όμως μόνο ένα τμήμα ELSE.
2. Σε μια εντολή IF εκτελούνται οι εντολές ενός μόνο τμήματος. Μετά την εκτέλεση των εντολών του τμήματος το πρόγραμμα συνεχίζει με τις εντολές που ακολουθούν το ENDIF.
3. Το τμήμα ELSE ή τμήματα ELSE IF δεν είναι υποχρεωτικό να υπάρχουν.
4. Οι εντολές που βρίσκονται εντός της εντολής IF γράφονται λίγο πιο δεξιά, αφήνοντας μερικά κενά ή πατώντας το πλήκτρο TAB. Αυτό δεν απαιτείται από το συντακτικό της γλώσσας, αλλά είναι χρήσιμο ώστε ο κώδικας να είναι ευανάγνωστος.

3.2 Παραδείγματα

3.2.1 Εύρεση του μικρότερου από δύο αριθμούς

Δεδομένων δύο αριθμών A και B να βρεθεί ποιος είναι ο μικρότερος.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION A, B, M`	Δήλωση μεταβλητών. A και B είναι οι μεταβλητές προς σύγκριση και M είναι η μικρότερη εκ των δύο.
⋮	
`IF (A.LT.B) THEN`	Ελέγχουμε αν η A είναι μικρότερη από τη B.
`M = A`	Αν η ανωτέρω σύγκριση είναι αληθής θέτουμε ως μικρότερη την A.
`ELSE`	Εδώ φτάνει το πρόγραμμα αν η σύγκριση είναι ψευδής (δηλαδή η A **δεν** είναι μικρότερη από τη B).
`M = B`	Θέτουμε ως μικρότερη τη B.
`END IF`	Τέλος της εντολής IF.
`WRITE (*,*) 'Ο μικρότερος είναι', M`	Εμφάνιση στην οθόνη του μικρότερου αριθμού.

Σημειώστε ότι στο ανωτέρω απόσπασμα προγράμματος χρησιμοποιήσαμε μεταβλητές διπλής ακρίβειας, όμως η ίδια τεχνική εφαρμόζεται με μεταβλητές και των άλλων τύπων.

3.2.2 Εύρεση του μικρότερου από τρεις αριθμούς

Δεδομένων τριών αριθμών A, B και C να βρεθεί ποιος είναι ο μικρότερος.

Εδώ εφαρμόζουμε την τεχνική του προηγούμενου παραδείγματος δύο φορές. Συγκεκριμένα βρίσκουμε τη μικρότερη από τις A και B και την ονομάζουμε M2. Κατόπιν βρίσκουμε τη μικρότερη από τις M2 και C.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION A, B, C, M2, M`	Δήλωση μεταβλητών. A, B και C είναι οι μεταβλητές προς σύγκριση. M2 είναι η μικρότερη εκ των A και B. M είναι η μικρότερη από τις A, B, C.
⋮	
`IF (A.LT.B) THEN`	Με αυτή την εντολή IF βρίσκουμε τη μικρότερη από τις A, B και την αποθηκεύουμε στη μεταβλητή M2.
`  M2 = A`	
`ELSE`	
`  M2 = B`	
`END IF`	
`IF (C.LT.M2) THEN`	Με αυτή την εντολή IF βρίσκουμε τη μικρότερη από τις M2, C και την αποθηκεύουμε στη μεταβλητή M.
`  M = C`	
`ELSE`	
`  M = M2`	
`END IF`	
`WRITE (*,*) 'Ο μικρότερος είναι', M`	Εμφανίζουμε το μικρότερο αριθμό στην οθόνη.

3.2.3 Εξίσωση δευτέρου βαθμού

Το παρακάτω πρόγραμμα βρίσκει τις πραγματικές λύσεις της εξίσωσης δευτέρου βαθμού

$$Ax^2 + Bx + C$$

όταν δίνονται οι συντελεστές A, B, C.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM EQUAT2`	
`IMPLICIT NONE`	
`DOUBLE PRECISION A, B, C`	Δήλωση μεταβλητών. A, B και C είναι οι συντελεστές της εξίσωσης.

```
DOUBLE PRECISION D, X1, X2          Δήλωση μεταβλητών. D είναι η διακρί-
                                    νουσα της εξίσωσης. Οι μεταβλητές X1,
                                    X2 είναι οι λύσεις της εξίσωσης.

WRITE (*,*) 'Εισάγετε τα A,B,C'
READ (*,*) A, B, C                  Εισάγονται τα A, B, C από το πληκτρολό-
                                    γιο.
IF (A.NE.0) THEN                    Ελέγχεται αν ο συντελεστής A είναι διάφο-
                                    ρος του μηδενός.
    D = B**2-4*A*C                  Υπολογίζεται η διακρίνουσα.
    IF (D.GT.0) THEN                Ελέγχεται αν η διακρίνουσα είναι θετική.
        X1 = (-B+SQRT(D))/(2*A)     Υπολογίζεται η πρώτη λύση.
        X2 = (-B-SQRT(D))/(2*A)     Υπολογίζεται η δεύτερη λύση.
        WRITE(*,*)'Δύο λύσεις',X1,X2 Εμφανίζονται στην οθόνη οι δύο λύσεις.
    ELSE IF (D.EQ.0) THEN           Ελέγχεται αν η διακρίνουσα είναι ίση με
                                    μηδέν.
        X1 = -B/(2*A)               Υπολογίζεται η λύση.
        WRITE(*,*)'Μία διπλή λύση',X1 Εμφανίζεται στην οθόνη η λύση.
    ELSE                            Εδώ φτάνει το πρόγραμμα όταν η διακρί-
                                    νουσα είναι αρνητική.
        WRITE(*,*)'Καμία λύση'
    END IF                          Τέλος του IF που ελέγχει τη διακρίνουσα.
ELSE                                Εδώ έρχεται το πρόγραμμα όταν ο συντε-
                                    λεστής A είναι μηδέν. Αυτό σημαίνει ότι η
                                    εξίσωση είναι πρώτου βαθμού.
    IF (B.NE.0) THEN                Ελέγχεται ο συντελεστής B αν είναι διάφο-
                                    ρος του μηδενός.
        X1 = -C/B                   Υπολογίζεται η λύση.
        WRITE(*,*)'Μία λύση',X1     Εμφανίζεται στην οθόνη η λύση.
    ELSE                            Εδώ φτάνει το πρόγραμμα αν το B είναι ίσο
                                    με μηδέν.
        WRITE(*,*)'Καμία λύση'
    END IF                          Τέλος του IF που ελέγχει το συντελεστή
                                    B.
END IF                              Τέλος του IF που ελέγχει το συντελεστή
                                    A.
END
```

3.3 Σύνθετες λογικές παραστάσεις

Δύο ή περισσότερες συγκρίσεις μπορούν να συνδυαστούν έτσι ώστε να κατασκευαστεί μια πιο σύνθετη λογική παράσταση με αποτέλεσμα που μπορεί να είναι αληθές ή ψευδές. Για το σκοπό αυτό χρησιμοποιούνται οι *λογικοί τελεστές* .AND., .OR. και .NOT. Οι δύο πρώτοι χρησιμοποιούνται με

δύο λογικές παραστάσεις ως:

Λ1 .AND. *Λ2*

Λ1 .OR. *Λ2*

ενώ ο τελεστής .NOT. με μία λογική παράσταση ως:

.NOT. *Λ*

Παραδείγματα:

```
X.GT.0 .AND. Y.GT.0
A.NE.0 .OR. B.NE.0
.NOT. K.EQ.2
A.EQ.0 .AND. B.EQ.0 .OR. X.EQ.1 .AND. Y.EQ.2
```

Το αποτέλεσμα των τελεστών .AND. και .OR. καθορίζεται από τον παρακάτω πίνακα (*Λ1* και *Λ2* είναι λογικές παραστάσεις).

Λ1	*Λ2*	*Λ1* .AND. *Λ2*	*Λ1* .OR. *Λ2*
Αληθής	Αληθής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής

Το αποτέλεσμα του τελεστή .NOT. καθορίζεται από τον παρακάτω πίνακα (*Λ* είναι μια λογική παράσταση).

Λ	.NOT. *Λ*
Αληθής	Ψευδής
Ψευδής	Αληθής

Σε μια σύνθετη λογική έκφραση η σειρά των πράξεων καθορίζεται από την προτεραιότητα του κάθε τελεστή:

Λογικός τελεστής	*Προτεραιότητα*
( )	Υψηλή
.NOT.	
.AND.	
.OR.	Χαμηλή

Για τελεστές με την ίδια προτεραιότητα οι πράξεις γίνονται από αριστερά προς τα δεξιά.

Παράδειγμα:

Το κέντρο ενός τετραγώνου πλευράς A βρίσκεται στο κέντρο των αξόνων. Δίνεται ένα σημείο με συντεταγμένες X,Y. Ελέγξτε αν το σημείο βρίσκεται εντός ή εκτός του τετραγώνου.

Εφόσον το κέντρο του τετραγώνου βρίσκεται στο κέντρο των αξόνων, τότε αυτό εκτείνεται οριζοντίως από -A/2 έως A/2, και καθέτως επίσης από -A/2 έως A/2. Αρκεί λοιπόν να ελέγξουμε αν η συντεταγμένη X του σημείου που θα δοθεί είναι εντός του διαστήματος -A/2 έως A/2 και το ίδιο για τη συντεταγμένη Y.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION A, X, Y, M`	Δηλώσεις μεταβλητών. A είναι η πλευρά του τετραγώνου. X, Y είναι οι συντεταγμένες του σημείου, και M είναι το ήμισυ της πλευράς.
⋮	
`M = A/2`	Υπολογίζεται το ήμισυ της πλευράς.
`IF (X.GE.-M .AND. X.LE.M .AND.`	Ελέγχεται αν η συντεταγμένη X είναι εντός του διαστήματος [-A/2,A/2] και ταυτόχρονα το ίδιο και η συντεταγμένη Y. Σημειώστε ότι επειδή χρησιμοποιούνται οι τελεστές `.GE.` και `.LE.` και όχι οι `.GT.` και `.LT.` τα σημεία που είναι πάνω στην περιφέρεια θεωρούνται εντός του τετραγώνου.
`&  Y.GE.-M .AND. Y.LE.M) THEN`	Η γραμμή αυτή αποτελεί συνέχεια της προηγούμενης αφού στη θέση 6 έχει το σύμβολο `&`.
`  WRITE(*,*)'Εντός τετραγώνου'`	
`ELSE`	Στο σημείο αυτό έρχεται το πρόγραμμα αν η λογική παράσταση στην εντολή IF είναι ψευδής.
`  WRITE(*,*)'Εκτός τετραγώνου'`	
`END IF`	

Παράδειγμα:

Στο προηγούμενο παράδειγμα θεωρείστε τον κύκλο που είναι εγγεγραμμένος στο τετράγωνο (έχει διάμετρο A) και τον κύκλο που είναι περιγεγραμμένος στο τετράγωνο (έχει διάμετρο $A\sqrt{2}$. Μπορούμε να ορίσουμε τις εξής τέσσερις περιοχές.

1. Η περιοχή εντός του εγγεγραμμένου κύκλου.
2. Η περιοχή μεταξύ τετραγώνου και εγγεγραμμένου κύκλου.
3. Η περιοχή μεταξύ τετραγώνου και περιγεγραμμένου κύκλου.
4. Η περιοχή εκτός του περιγεγραμμένου κύκλου.

Δοθέντος ένας σημείου με συντεταγμένες X,Y, βρείτε σε ποια από τις τέσσερις περιοχές βρίσκεται. Θα χρησιμοποιήσουμε μια εντολή IF με πολλαπλούς ελέγχους.

```
DOUBLE PRECISION A, X, Y, M
   ⋮
M = A/2
IF (X**2+Y**2.LE.M) THEN
    WRITE (*,*) 'Εντός του εγγεγραμμένου κύκλου'
ELSE IF (X.GE.-M .AND. X.LE.M .AND. Y.GE.-M .AND. Y.LE.M) THEN
    WRITE (*,*) 'Εντός του τετραγώνου'
ELSE IF (X**2+Y**2.LE.M*SQRT(2.)) THEN
    WRITE (*,*) 'Εντός του περιγεγραμμένου κύκλου'
ELSE
    WRITE (*,*) 'Εκτός'
END IF
```

Παράδειγμα:

Δίνεται κύκλος ακτίνας R και δύο σημεία με συντεταγμένες (X1,Y1) και (X2,Y2). Αν και τα δύο σημεία είναι εντός του κύκλου ή και τα δύο σημεία είναι εκτός του κύκλου βρείτε τη μεταξύ τους απόσταση. Αν ένα σημείο είναι εντός και το άλλο εκτός βρείτε ποιο από τα δύο απέχει λιγότερο από την περιφέρεια του κύκλου και ποια είναι η απόσταση αυτή.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM CIRC2`	
`IMPLICIT NONE`	
`DOUBLE PRECISION R`	Δήλωση μεταβλητών. R είναι η ακτίνα του κύκλου.
`DOUBLE PRECISION X1, Y1, X2, Y2`	Δήλωση μεταβλητών. X1, Y1, X2, Y2 είναι οι συντεταγμένες των δύο σημείων.
`DOUBLE PRECISION A1, A2, D`	Δήλωση μεταβλητών. A1, A2 είναι οι αποστάσεις των σημείων από το κέντρο των αξόνων και D η απόσταση των δύο σημείων μεταξύ τους.
`DOUBLE PRECISION P1, P2, P`	Δήλωση μεταβλητών. P1, P2 είναι είναι οι αποστάσεις των δύο σημείων από την περιφέρεια του κύκλου. P είναι η μικρότερη εκ των δύο αυτών αποστάσεων.
`WRITE(*,*)'Εισάγετε την ακτίνα R'`	
`READ (*,*) R`	Εισαγωγή της ακτίνας R από το πληκτρολόγιο.
`WRITE(*,*)'Εισάγετε τα X1,Y1,X2,Y2'`	
`READ (*,*) X1, Y1, X2, Y2`	Εισαγωγή των συντεταγμένων X1, Y1, X2, Y2 από το πληκτρολόγιο.
`A1 = SQRT(X1**2+Y1**2)`	Υπολογισμός της απόστασης του πρώτου σημείου από το κέντρο των αξόνων.

```
      A2 = SQRT(X2**2+Y2**2)
```
Υπολογισμός της απόστασης του δεύτερου σημείου από το κέντρο των αξόνων.

```
      IF (A1.LE.R .AND. A2.LE.R .OR.
```
Έλεγχος αν και τα δύο σημεία είναι εντός του κύκλου ή και τα δύο σημεία είναι εκτός του κύκλου. Παρατηρείστε ότι επειδή στην εντολή IF χρησιμοποιείται ο τελεστής .LE., τα σημεία που βρίσκονται πάνω στην περιφέρεια, θεωρούνται εντός του κύκλου.

```
     &    A1.GT.R .AND. A2.GT.R) THEN
```
Η γραμμή αυτή αποτελεί συνέχεια της προηγούμενης αφού στη θέση 6 έχει το σύμβολο &.

```
          D = SQRT((X1-X2)**2+(Y1-Y2)**2)
```
Υπολογίζεται η απόσταση μεταξύ των δύο σημείων.

```
          WRITE (*,*) D
```
Εμφανίζεται στην οθόνη η απόσταση.

```
      ELSE
```
Εδώ έρχεται το πρόγραμμα όταν η λογική παράσταση στην εντολή IF είναι ψευδής.

```
          P1 = ABS(R-A1)
```
Υπολογίζεται η απόσταση του πρώτου σημείου από την περιφέρεια.

```
          P2 = ABS(R-A2)
```
Υπολογίζεται η απόσταση του δεύτερου σημείου από την περιφέρεια.

```
          IF (P1.LT.P2) THEN
```
Με αυτή την εντολή IF βρίσκουμε τη μικρότερη εκ των δύο αποστάσεων P1, P2 και την αποθηκεύουμε στη μεταβλητή P.

```
              P = P1
          ELSE
              P = P2
          END IF
```
Εδώ τελειώνει η εντολή IF που βρίσκει τη μικρότερη απόσταση.

```
          WRITE (*,*) P
```
Εμφανίζεται στην οθόνη η μικρότερη απόσταση P.

```
      END IF
```
Εδώ τελειώνει η αρχική εντολή IF.

```
      END
```

Κεφάλαιο 4

Εντολή DO

4.1 Πως συντάσσεται

Με την εντολή DO έχουμε τη δυνατότητα να επαναλάβουμε την εκτέλεση ενός τμήματος προγράμματος όσες φορές θέλουμε. Η εντολή συντάσσεται ως εξής:

```
DO μεταβλητή = αρχική τιμή, τελική τιμή, βήμα
   εντολή1
   εντολή2
     ⋮
END DO
```

Παράδειγμα:

```
DO K = 1,9,2
   WRITE (*,*) K
END DO
```

Η εντολή λειτουργεί ως εξής: Αρχικά η *μεταβλητή* (K) λαμβάνει την *αρχική τιμή* (1). Κατόπιν εκτελούνται διαδοχικά όλες οι εντολές που βρίσκονται εντός του τμήματος DO–ENDDO. Στο ανωτέρω παράδειγμα θα εκτελεστεί η εντολή WRITE και θα εμφανιστεί η τιμή της μεταβλητής K στην οθόνη. Η εκτέλεση των εντολών που βρίσκονται εντός του τμήματος DO–ENDDO ονομάζεται *επανάληψη*. Στη συνέχεια το πρόγραμμα επιστρέφει (προς τα πίσω) στην εντολή DO, όπου η μεταβλητή αυξάνεται όσο προσδιορίζεται από το *βήμα* (2). Κατά συνέπεια η μεταβλητή K θα λάβει την τιμή 3. Αν η *μεταβλητή* δεν ξεπεράσει την *τελική τιμή* (9), τότε πραγματοποιείται ακόμη μία επανάληψη. Στο ανωτέρω παράδειγμα αυτό θα έχει σαν αποτέλεσμα να εμφανιστεί και πάλι η τιμή της μεταβλητής K στην οθόνη (3). Διαφορετικά οι επαναλήψεις τερματίζονται, δηλαδή το πρόγραμμα συνεχίζει με τις εντολές που βρίσκονται κάτω από το ENDDO. Στο ανωτέρω παράδειγμα θα πραγματοποιηθούν συνολικά 5 επαναλήψεις και θα εμφανιστούν στην οθόνη οι αριθμοί: 1, 3, 5, 7 και 9.

Ορισμένες παρατηρήσεις που αφορούν τη λειτουργία της εντολής DO:

1. Η *μεταβλητή* που ονομάζεται και *δείκτης* της εντολής DO πρέπει να είναι μια ακέραια μεταβλητή. Υπάρχει δυνατότητα για πραγματικές μεταβλητές στην εντολή DO αλλά δεν θα την

χρησιμοποιήσουμε.

2. Η τιμή της μεταβλητής αυξάνεται αυτόματα από την εντολή DO και δεν επιτρέπεται η αλλαγή της με εντολές του τύπου K=... ή READ (*,*) K ή με άλλο τρόπο.

3. Το *βήμα* μπορεί να παραλειφθεί από την εντολή DO, οπότε εννοείται η τιμή 1.

4. Το βήμα μπορεί να λάβει και αρνητικές τιμές, όχι όμως 0.

5. Ανάλογα με την *αρχική τιμή*, *τελική τιμή* και το *βήμα*, μπορεί να μην πραγματοποιηθούν καθόλου επαναλήψεις.
   Παράδειγμα: DO M=15,8,2
   Σε αυτή την περίπτωση οι εντολές εντός του τμήματος DO–ENDDO παρακάμπτονται και το πρόγραμμα συνεχίζει με τις εντολές που ακολουθούν το ENDDO.

Παράδειγμα:

Ο παρακάτω πίνακας δείχνει ορισμένες εντολές DO, πόσες επαναλήψεις θα πραγματοποιήσουν και ποιες είναι οι διαδοχικές τιμές που θα λάβουν οι αντίστοιχες μεταβλητές.

Εντολή DO	*Πλήθος επαναλήψεων*	*Τιμές μεταβλητής*
`DO L=1,7,2`	4	1, 3, 5, 7
`DO L=10,20,3`	4	10, 13, 16, 19
`DO K=-2,2`	5	-2, -1, 0, 1 ,2
`DO N=1,10,25`	1	1
`DO K=-8,-10,-2`	2	-8, -10
`DO M=7,4,-1`	4	7, 6, 5, 4
`DO K=7,1,4`	0	
`DO M=25,20`	0	
`DO J=-5,-10,1`	0	

4.2 Παραδείγματα

4.2.1 Πίνακας τριγωνομετρικών αριθμών

Κατασκευάστε πρόγραμμα που θα τυπώνει πίνακα με τους τριγωνομετρικούς αριθμούς ημίτονο, συνημίτονο και εφαπτομένη για μια περιοχή γωνιών από M1 έως M2 μοίρες, ανά μία μοίρα.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM TRIGON`	
`IMPLICIT NONE`	
`INTEGER M1, M2, M`	Δήλωση ακέραιων μεταβλητών. Οι M1, M2 προσδιορίζουν τα όρια του πίνακα και η M χρησιμοποιείται ως δείκτης της εντολής DO.
`DOUBLE PRECISION PI, R`	Δήλωση μεταβλητών διπλής ακρίβειας. PI είναι ο αριθμός π και R είναι η γωνία σε ακτίνια.
`DOUBLE PRECISION S, C, T`	Δήλωση μεταβλητών διπλής ακρίβειας. S, C και T είναι το ημίτονο, συνημίτονο και εφαπτομένη της γωνίας, αντίστοιχα.
`WRITE (*,*) 'Εισάγετε τα M1, M2'`	
`READ (*,*) M1, M2`	Εισαγωγή των M1, M2 από το πληκτρολόγιο.
`PI = ACOS(-1.0D0)`	Υπολογισμός του αριθμού π.
`DO M=M1,M2`	Επανάληψη για όλες τις γωνίες από M1 έως M2 μοίρες.
`R = M*PI/180`	Μετατροπή των μοιρών σε ακτίνια.
`S = SIN(R)`	Υπολογισμός του ημιτόνου.
`C = COS(R)`	Υπολογισμός του συνημιτόνου.
`T = TAN(R)`	Υπολογισμός της εφαπτομένης.
`WRITE (*,10) M, S, C, T`	Εμφάνιση των αποτελεσμάτων στην οθόνη χρησιμοποιώντας την εντολή μορφοποίησης με αριθμό 10.
`END DO`	
`10 FORMAT(I6,3X,F11.8,3X,F11.8,3X,F11.8)`	Εντολή μορφοποίησης των αποτελεσμάτων.
`END`	

Ορισμένα σχόλια για το ανωτέρω πρόγραμμα:

1. Οι τριγωνομετρικές συναρτήσεις SIN, COS και TAN δέχονται το όρισμα τους σε ακτίνια. Δεδομένου ότι ο χρήστης εισάγει μοίρες, θα πρέπει πριν κληθούν οι συναρτήσεις αυτές να προηγηθεί μετατροπή της γωνίας σε ακτίνια.

2. Η εντολή μορφοποίησης FORMAT με αριθμό 10, προσδιορίζει ότι θα τυπωθεί ένας ακέραιος

αριθμός σε 6 θέσεις (I6), και τρεις πραγματικοί αριθμοί σε 11 θέσεις ο καθένας με 8 δεκαδικά ψηφία (F11.8). Μεταξύ των αριθμών μένουν τρεις κενές θέσεις (3X).

3. Ενδεικτικά το αποτέλεσμα για M1=0 και M2=5 είναι:

```
0     0.00000000     1.00000000     0.00000000
1     0.01745241     0.99984770     0.01745506
2     0.03489950     0.99939083     0.03492077
3     0.05233596     0.99862953     0.05240778
4     0.06975647     0.99756405     0.06992681
5     0.08715574     0.99619470     0.08748866
```

4.2.2 Αθροίσματα και γινόμενα με σταθερούς όρους

Γράψτε τμήμα προγράμματος που για δεδομένο N θα υπολογίζει το κάτωθι άθροισμα.

$$1 + 2 + 3 + \cdots + N$$

Γενικά για να υπολογίσουμε αθροίσματα χρησιμοποιούμε μια μεταβλητή που ονομάζεται "αθροιστής" και στην οποία σταδιακά συσσωρεύουμε όλους τους όρους του αθροίσματος. Στις μεταβλητές που χρησιμοποιούνται ως αθροιστές απαιτείται να αναθέσουμε μια αρχική τιμή (μηδέν, ή κάποια άλλη ανάλογα με το πρόβλημα). Η συσσώρευση στον αθροιστή γίνεται με εντολές του τύπου:

`S = S +` *αριθμητική παράσταση*

Στην εντολή αυτή πρώτα θα γίνουν οι πράξεις στα δεξιά του συμβόλου =, δηλαδή θα προστεθεί η *αριθμητική παράσταση* στον αθροιστή `S`. Κατόπιν το αποτέλεσμα τίθεται ως νέα τιμή του `S`. Στο συγκεκριμένο παράδειγμα το αποτέλεσμα της άθροισης είναι αναλυτικά γνωστό και ίσο με $N(N+1)/2$.

Κώδικας Fortran	*Σχολιασμός*
`INTEGER N, K, S`	Δήλωση ακέραιων μεταβλητών. N είναι το πλήθος των όρων του αθροίσματος, K είναι ο δείκτης της εντολής DO και S ο αθροιστής.
⋮	
`S = 0`	Ο αθροιστής λαμβάνει αρχική τιμή.
`DO K=1,N`	Επανάληψη N φορές.
`S = S+K`	Στο άθροισμα προστίθεται η τιμή της μεταβλητής K.
`END DO`	

Γράψτε τμήμα προγράμματος που για δεδομένο N θα υπολογίζει το κάτωθι άθροισμα.

$$1^2 + 2^2 + 3^2 + \cdots + N^2$$

Το αποτέλεσμα της άθροισης είναι αναλυτικά γνωστό και ίσο με $N(N+1)(2N+1)/6$.

Κώδικας Fortran	*Σχολιασμός*
`INTEGER N, K, S`	Δήλωση ακέραιων μεταβλητών. N είναι το πλήθος των όρων του αθροίσματος, K είναι ο δείκτης της εντολής DO και S ο αθροιστής.
⋮	
`S = 0`	Ο αθροιστής λαμβάνει αρχική τιμή.
`DO K=1,N`	Επανάληψη N φορές.
`S = S+K**2`	Στο άθροισμα προστίθεται το K**2.
`END DO`	

Γράψτε τμήμα προγράμματος που για δεδομένο N θα υπολογίζει το κάτωθι γινόμενο γνωστό και ως N–παραγοντικό (συμβολίζεται N!).

$$N! = 1 \cdot 2 \cdot 3 \cdots N$$

Κώδικας Fortran	*Σχολιασμός*
`INTEGER N, K, P`	Δήλωση ακέραιων μεταβλητών. N είναι το πλήθος των όρων του γινομένου, K είναι ο δείκτης της εντολής DO και P το γινόμενο.
⋮	
`P = 1`	Το γινόμενο λαμβάνει αρχική τιμή.
`DO K=2,N`	Επανάληψη για όλους τους όρους του γινομένου πλην του αρχικού.
`P = P*K`	Το γινόμενο πολλαπλασιάζεται με την τιμή της μεταβλητής K.
`END DO`	

Γράψτε τμήμα προγράμματος που για δεδομένο N θα αθροίζει την κάτωθι σειρά:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots \frac{1}{N}$$

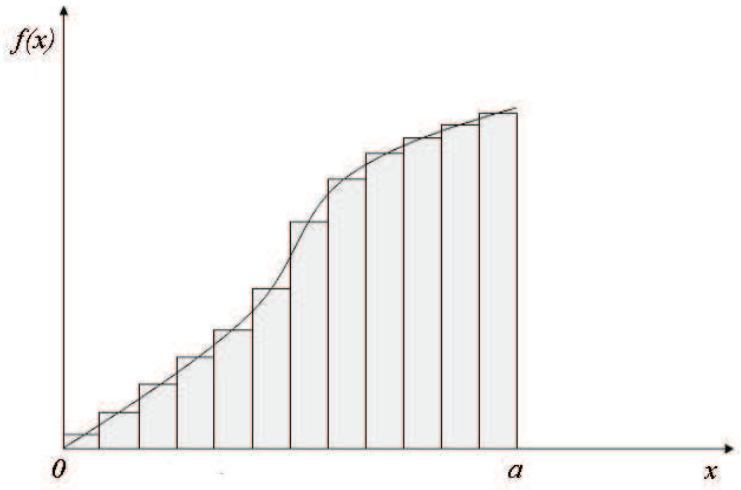
Η σειρά αυτή αποτελεί μια προσέγγιση του $\pi/4$. Η ιδιαιτερότητα που έχει σε σχέση με τα προηγούμενα παραδείγματα είναι ότι τα πρόσημα των όρων εναλλάσσονται Για το λόγο αυτό ορίζουμε μια μεταβλητή προσήμου (P). Η μεταβλητή αυτή λαμβάνει αρχικά την τιμή 1 επειδή το πρόσημο του πρώτου όρου είναι θετικό. Κατόπιν μόλις ο πρώτος όρος προστεθεί στο άθροισμα η μεταβλητή P αλλάζει πρόσημο (P = -P) και γίνεται -1. Έτσι στη δεύτερη επανάληψη ο όρος 1/3 αφαιρείται από το άθροισμα κοκ. Σημειώστε ότι η σειρά συγκλίνει πολύ αργά και για να πετύχουμε σωστά 4 σημαντικά ψηφία απαιτείται περίπου N=10000.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION P, S`	Δήλωση μεταβλητών διπλής ακρίβειας. P είναι το πρόσημο και S ο αθροιστής.
`INTEGER N, K`	Δήλωση ακέραιων μεταβλητών. N είναι ο παρονομαστής του τελευταίου όρου του αθροίσματος και K ο δείκτης της εντολής DO.
⋮	
`S = 0.0`	Αρχική τιμή του αθροιστή.
`P = 1.0`	Αρχική τιμή του προσήμου.
`DO K=1,N,2`	Επανάληψη για όλους τους όρους. Προσέξτε ότι το βήμα είναι 2.
`S = S + P/K`	Ο κάθε όρος προστίθεται στο άθροισμα πολλαπλασιασμένος με το πρόσημο.
`P = -P`	Αλλαγή προσήμου για τον επόμενο όρο του αθροίσματος.
`END DO`	

4.2.3 Ανατοκισμός

Ένα αρχικό κεφάλαιο K τοκίζεται στο τέλος κάθε μήνα με επιτόκιο E %. Αμέσως μετά ακολουθεί ανάληψη ποσού A αν υπάρχει επαρκές υπόλοιπο. Ποιο ποσό μένει μετά από N μήνες ;

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM BANK`	
`IMPLICIT NONE`	
`DOUBLE PRECISION K, E, A, T`	Δήλωση μεταβλητών διπλής ακρίβειας. K είναι το κεφάλαιο, E είναι το επιτόκιο(%), A το ποσό της ανάληψης και T οι τόκοι κάθε μήνα.
`INTEGER N, I`	Δήλωση ακέραιων μεταβλητών. N είναι το πλήθος των μηνών και το I χρησιμοποιείται ως δείκτης των μηνών στην εντολή DO.
`WRITE (*,*) 'Εισάγετε τα K,E,A,N'`	
`READ (*,*) K, E, A, N`	Εισαγωγή των K,E,A και N από το πληκτρολόγιο.
`DO I=1,N`	Επανάληψη για όλους τους μήνες.
`T = K * E/100`	Υπολογισμός των τόκων του μήνα.
`K = K+T`	Προσθήκη των τόκων στο κεφάλαιο.
`IF (K.GE.A) K = K-A`	Έλεγχος αν υπάρχει διαθέσιμο υπόλοιπο. Αν ναι γίνεται ανάληψη ποσού A.
`END DO`	
`WRITE (*,*) K`	Εμφάνιση των αποτελεσμάτων.
`END`	



Σχήμα 4.1: Υπολογισμός ολοκληρώματος με τον κανόνα του παραλληλογράμμου.

4.2.4 Υπολογισμός ολοκληρώματος

Δεδομένης μιας γνωστής συνάρτησης $f(x)$, κατασκευάστε πρόγραμμα που θα υπολογίζει προσεγγιστικά το ολοκλήρωμα

$$I = \int_0^a f(x)dx$$

όταν δίνεται το άνω όριο ολοκλήρωσης a, χρησιμοποιώντας τον κανόνα του παραλληλογράμμου.

Το ανωτέρω ολοκλήρωμα είναι το εμβαδόν της επιφάνειας που περικλείεται από την $f(x)$ και τον άξονα x (σχ. 4.1). Ο κανόνας του παραλληλογράμμου έχει ως εξής: Χωρίζουμε το διάστημα $[0, a]$ σε N ίσα υποδιαστήματα μήκους $h = a/N$ το καθένα. Αν τα υποδιαστήματα είναι αρκετά μικρά μπορούμε σε ικανοποιητικό βαθμό να προσεγγίσουμε το εμβαδόν του καθενός από ένα παραλληλόγραμμο που η οριζόντια πλευρά του έχει μήκος h, ενώ το ύψος της κατακόρυφης πλευράς είναι η τιμή της συνάρτησης $f(x)$ υπολογισμένη στο μέσο του διαστήματος. Το μέσο του πρώτου υποδιαστήματος είναι $h/2$, το μέσου του δεύτερου $h+h/2$, του τρίτου $2h+h/2$ και γενικά το μέσο του υποδιαστήματος υπ' αριθμόν i είναι $(i-1)h+h/2$ ή αλλιώς $ih-h/2$. Κατά συνέπεια το εμβαδόν του υποδιαστήματος υπ' αριθμόν i είναι $hf(ih-h/2)$. Αρκεί λοιπόν να αθροίσουμε τα N επιμέρους εμβαδά.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM INTEG`	
`IMPLICIT NONE`	
`DOUBLE PRECISION A, H, S, X, F`	Δηλώνονται οι μεταβλητές διπλής ακρίβειας. A είναι το άνω όριο ολοκλήρωσης, H το μήκος του κάθε υποδιαστήματος, S η τιμή του ολοκληρώματος (αθροιστής), X το μέσο του κάθε υποδιαστήματος και F η τιμή της συνάρτησης στο μέσο του υποδιαστήματος.

`INTEGER N, I`	Δηλώνονται οι ακέραιες μεταβλητές. N είναι το πλήθος των υποδιαστημάτων και η μεταβλητή I χρησιμοποιείται ως δείκτης στην εντολή DO.
`WRITE (*,*) 'Εισάγετε τα A,N'`	
`READ (*,*) A, N`	Εισάγονται τα A, N από το πληκτρολόγιο.
`H = A/N`	Υπολογίζεται το μήκος του κάθε υποδιαστήματος.
`S = 0.`	Δίνεται αρχική τιμή στον αθροιστή.
`DO I=1,N`	Επανάληψη για όλα τα υποδιαστήματα.
`X = I*H-H/2`	Υπολογίζεται το μέσο του υποδιαστήματος I.
`F = SIN(X)`	Υπολογίζεται η τιμή της συνάρτησης στο μέσο του υποδιαστήματος. Εδώ χρησιμοποιούμε τη συνάρτηση SIN(X).
`S = S+H*F`	Υπολογίζεται το εμβαδόν του κάθε υποδιαστήματος και αθροίζεται.
`END DO`	
`WRITE (*,*) 'Ολοκλήρωμα=',S`	Εμφανίζεται το αποτέλεσμα.
`END`	

Σημειώστε ότι για τη συνάρτηση $\sin x$ που χρησιμοποιήσαμε το αντίστοιχο ολοκλήρωμα είναι αναλυτικά γνωστό:

$$I = \int_0^a \sin x \; dx = -\cos a + cos\ 0 = -\cos a + 1$$

Κατά συνέπεια εάν θέλουμε να ελέγξουμε την ορθότητα του προγράμματός μας μπορούμε στο τέλος να τυπώσουμε την ποσότητα -COS(A)+1 για επιβεβαίωση. Φυσικά η επιβεβαίωση αυτή δεν είναι δυνατή για οποιαδήποτε συνάρτηση.

4.2.5 Αναδρομικές σχέσεις

Με ένα προηγούμενο όρο

Γράψτε πρόγραμμα που θα υπολογίζει το N–στό όρο της ακολουθίας:

$$a_k = 2 + \frac{1}{a_{k-1}}$$

με πρώτο όρο $a_1 = 1$. Η ακολουθία αυτή είναι γνωστό ότι συγκλίνει γρήγορα στην τιμή $1 + \sqrt{2}$.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM AKOL`	
`IMPLICIT NONE`	
`DOUBLE PRECISION A`	Στην μεταβλητή A αποθηκεύεται ο κάθε όρος της ακολουθίας.
`INTEGER N, I`	Η μεταβλητή N είναι το πλήθος των όρων της ακολουθίας ενώ η I χρησιμοποιείται ως δείκτης στην εντολή DO.
`WRITE (*,*) 'Ποιόν όρο θέλετε ;'`	
`READ (*,*) N`	Εισαγωγή του πλήθους των όρων από το πληκτρολόγιο.
`A = 1`	Ο πρώτος όρος της ακολουθίας.
`DO I=2,N`	Επανάληψη για τους υπόλοιπους όρους (από τον δεύτερο και μετά).
`      A = 2+1/A`	Υπολογισμός του επόμενου όρου. Το αποτέλεσμα αποθηκεύεται και πάλι στη μεταβλητή A.
`END DO`	
`WRITE (*,*) A`	Εμφάνιση του αποτελέσματος στην οθόνη.
`END`	

Με δύο προηγούμενους όρους. Ακολουθία Fibonacci

Κατασκευάστε πρόγραμμα που θα εμφανίζει στην οθόνη την ακολουθία αριθμών Fibonacci που δίνονται από τη σχέση:

$$F_n = F_{n-1} + F_{n-2}$$

με αρχικές τιμές $F_0 = 0$ και $F_1 = 1$. Παρατηρείστε ότι κάθε όρος της ακολουθίας εξαρτάται από τους δύο αμέσως προηγούμενους όρους.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM FIBO`	
`IMPLICIT NONE`	
`INTEGER FA, FB, FC`	Δήλωση ακέραιων μεταβλητών. FC είναι ο τρέχων όρος της ακολουθίας, FB ο προηγούμενος και FA ο προ–προηγούμενος.
`INTEGER N, I`	Δήλωση ακέραιων μεταβλητών. N είναι ο τελευταίος όρος της ακολουθίας και I είναι ο δείκτης της εντολής DO.
`WRITE (*,*) 'Τελευταίος όρος;'`	
`READ (*,*) N`	
`FA = 0`	Αρχική τιμή στον πρώτο όρο (F0).
`FB = 1`	Αρχική τιμή στον δεύτερο όρο (F1).
`WRITE (*,*) FA`	Εμφάνιση στην οθόνη του πρώτου όρου.
`WRITE (*,*) FB`	Εμφάνιση στην οθόνη του δεύτερου όρου.
`DO I=2,N`	Επανάληψη για όλους τους υπόλοιπους όρους της ακολουθίας μέχρι το N–στο.
`  FC = FA+FB`	Υπολογισμός του όρου υπ' αριθμόν I.
`  WRITE (*,*) FC`	Εμφάνιση στην οθόνη.
`  FA = FB`	Ο προ–προηγούμενος όρος παίρνει την τιμή του προηγούμενου.
`  FB = FC`	Ο προηγούμενος όρος παίρνει την τιμή του τρέχοντα όρου.
`END DO`	
`END`	

4.2.6 Σειρές με μεταβλητούς όρους

Γράψτε τμήμα προγράμματος που για δεδομένο N θα αθροίζει την κάτωθι σειρά:

$$1 + x + x^2 + x^3 + \cdots + x^{N-1}$$

Το ανωτέρω άθροισμα έχει αναλυτικά γνωστό αποτέλεσμα ίσο με $(1 - x^N)/(1 - x)$.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM XSUM`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, S`	Δήλωση μεταβλητών διπλής ακρίβειας. S είναι ο αθροιστής.
`INTEGER N, K`	Δήλωση ακέραιων μεταβλητών. N είναι το πλήθος των όρων του αθροίσματος και η K χρησιμεύει σαν δείκτης της εντολής DO.
`WRITE (*,*) 'Εισάγετε τα X, N'`	
`READ (*,*) X, N`	Εισαγωγή των X, N από το πληκτρολόγιο.
`S = 1`	Αρχική τιμή του αθροίσματος.
`DO K=1,N-1`	Επανάληψη για κάθε ένα όρο του αθροίσματος.
`  S = S+X**K`	Προστίθεται ο όρος X**K στο άθροισμα.
`END DO`	
`WRITE (*,*) S`	Εμφάνιση του αποτελέσματος.
`END`	

4.2.7 Σειρές Taylor

Εκθετικό

Κατασκευάστε πρόγραμμα το οποίο θα αθροίζει τη σειρά:

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^N}{N!}$$

Το ανωτέρω άθροισμα αποτελεί μια προσέγγιση του e^x.

Παρατηρήστε ότι σε κάθε όρο του αθροίσματος $x^k/k!$ υπάρχει στον παρονομαστή το $k!$. Συνεπώς χρειαζόμαστε δύο εντολές DO. Η μια θα διατρέχει όλους τους όρους του αθροίσματος και η δεύτερη θα βρίσκεται εντός της πρώτης και θα υπολογίζει το αντίστοιχο παραγοντικό για κάθε όρο.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM EXPON1`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, S, P`	Δήλωση μεταβλητών διπλής ακρίβειας. S είναι ο αθροιστής και η P χρησιμοποιείται στον υπολογισμό του παραγοντικού.
`INTEGER N, I, K`	Δήλωση ακέραιων μεταβλητών. Οι μεταβλητές I, K χρησιμεύουν ως δείκτες των εντολών DO.
`WRITE (*,*) 'Εισάγετε τα X, N'`	
`READ (*,*) X, N`	Εισαγωγή των X, N από το πληκτρολόγιο.
`S = 1`	Αρχική τιμή του αθροιστή.
`DO K=1,N`	Επανάληψη για όλους τους όρους του αθροίσματος.
`  P = 1.`	Αρχική τιμή για το παραγοντικό.
`  DO I=2,K`	Επανάληψη για τον υπολογισμό του K!
`    P = P*I`	
`  END DO`	
`  S = S + X**K/P`	Προστίθεται στο άθροισμα ο όρος X**K/K!
`END DO`	
`WRITE (*,*) S`	Εμφάνιση του αποτελέσματος.
`END`	

Ας δούμε πόσες αριθμητικές πράξεις θα κάνει το παραπάνω πρόγραμμα για μια δεδομένη τιμή του N. Παρατηρήστε ότι το εξωτερικό DO σε κάθε επανάληψη υπολογίζει το K! και επιπλέον κάνει 3 πράξεις (στην εντολή `S = S + X**K/P`). Το εσωτερικό DO (που υπολογίζει το παραγοντικό) για μια δεδομένη τιμή του K κάνει K-1 πολλαπλασιασμούς. Ο παρακάτω πίνακας συνοψίζει

K	*Πράξεις στο εσωτερικό DO*	*Πράξεις στο εξωτερικό DO*
1	0	3
2	1	3
3	2	3
⋮	⋮	3
N	N-1	3

Το άθροισμα της δεύτερης στήλης $(0+1+2+\cdots+N-1)$ είναι $N(N-1)/2$, ενώ το άθροισμα της τρίτης στήλης είναι 3N. Συνολικά λοιπόν χρειάζονται $N(N-1)/2+3N = (N^2+5N)/2$ αριθμητικές πράξεις.

Ένας άλλος τρόπος για να αθροίσουμε τη σειρά βασίζεται στην παρατήρηση ότι κάθε όρος της σειράς μπορεί να γραφεί σαν συνάρτηση του προηγούμενου, δηλαδή:

$$\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!}\ \frac{x}{k}$$

Έτσι μπορούμε να κατασκευάσουμε τον κάθε όρο $x^k/k!$ από τον προηγούμενό του, χωρίς να χρειάζεται να υπολογίζουμε το $k!$ κάθε φορά.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM EXPON2`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, S, T`	Δήλωση μεταβλητών διπλής ακρίβειας. S είναι ο αθροιστής και T ο κάθε όρος του αθροίσματος.
`INTEGER N, K`	Δήλωση ακέραιων μεταβλητών. Η μεταβλητή K χρησιμεύει ως δείκτης της εντολής DO.
`WRITE (*,*) 'Εισάγετε τα X, N'`	
`READ (*,*) X, N`	Εισαγωγή των X, N από το πληκτρολόγιο.
`S = 1`	Αρχική τιμή του αθροιστή.
`T = 1`	Η μεταβλητή T λαμβάνει την τιμή του πρώτου όρου του αθροίσματος.
`DO K=1,N`	Επανάληψη για όλους τους όρους του αθροίσματος.
`  T = T*X/K`	Η μεταβλητή T παίρνει την τιμή του επόμενου όρου.
`  S = S + T`	Προσθήκη στο άθροισμα.
`END DO`	
`WRITE (*,*) S`	Εμφάνιση του αποτελέσματος.
`END`	

Παρατηρήστε ότι απαιτείται μόνο μια εντολή DO, εντός της οποίας γίνονται 3 πράξεις σε κάθε επανάληψη οπότε το συνολικό πλήθος πράξεων είναι 3N.

Συνημίτονο

Κατασκευάστε πρόγραμμα το οποίο θα αθροίζει τη σειρά Taylor του συνημιτόνου:

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \frac{x^N}{N!}$$

Θα εφαρμόσουμε την τεχνική του προηγούμενου παραδείγματος. Παρατηρούμε ότι ο όρος $x^k/k!$ μπορεί να γραφεί σαν συνάρτηση του προηγούμενου όρου:

$$\frac{x^k}{k!} = \frac{x^{k-2}}{(k-2)!} \frac{x^2}{(k-1)k}$$

Για παράδειγμα ο όρος $x^4/4!$ μπορεί να γραφεί με βάση τον προηγούμενο ως:

$$\frac{x^4}{4!} = \frac{x^2}{2!} \frac{x^2}{3 \cdot 4}$$

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM COSINE`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, S, T`	Δήλωση μεταβλητών διπλής ακρίβειας. S είναι ο αθροιστής και T ο κάθε όρος του αθροίσματος.
`INTEGER N, K`	Δήλωση ακέραιων μεταβλητών. Η μεταβλητή K χρησιμεύει ως δείκτης της εντολής DO.
`WRITE (*,*) 'Εισάγετε τα X, N'`	
`READ (*,*) X, N`	Εισαγωγή των X, N από το πληκτρολόγιο.
`S = 1`	Αρχική τιμή του αθροιστή.
`T = 1`	Η μεταβλητή T λαμβάνει την τιμή του πρώτου όρου του αθροίσματος.
`DO K=2,N,2`	Επανάληψη για όλους τους όρους του αθροίσματος. Το βήμα είναι 2 αφού στο άθροισμα υπάρχουν μόνο άρτιοι όροι.
`T = -T*X**2/(K*(K-1))`	Η μεταβλητή T παίρνει την τιμή του επόμενου όρου.
`S = S + T`	Προσθήκη στο άθροισμα.
`END DO`	
`WRITE (*,*) S`	Εμφάνιση του αποτελέσματος.
`END`	

Ημίτονο

Κατασκευάστε πρόγραμμα το οποίο θα αθροίζει τη σειρά Taylor του ημιτόνου:

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots \frac{x^N}{N!}$$

Όμοια όπως και πριν γράφουμε τον κάθε όρο του αθροίσματος σαν συνάρτηση του προηγούμενου.

```
      PROGRAM SINE
      IMPLICIT NONE
      DOUBLE PRECISION X, S, T
      INTEGER N, K
      WRITE (*,*) 'Εισάγετε τα X, N'
      READ (*,*) X, N
      S = X
      T = X
      DO K=3,N,2
            T = -T*X**2/(K*(K-1))
            S = S + T
      END DO
      WRITE (*,*) S
      END
```

Κεφάλαιο 5

Μονοδιάστατοι πίνακες

Σε πολλά προβλήματα μπορεί να χρειαζόμαστε πολλές μεταβλητές παρόμοιου τύπου και σημασίας. Για παράδειγμα, έστω ότι θέλουμε να ορίσουμε την τιμή μιας συνάρτησης σε N διαφορετικά σημεία. Θα μπορούσαμε να δηλώσουμε N διαφορετικές μεταβλητές για να αποθηκεύσουμε τις N διαφορετικές τιμές της συνάρτησης (φανταστείτε πόσο δύσκολο είναι αυτό για μεγάλα N, πχ για N=1000), ή πιο εύκολα να ορίσουμε μια μεταβλητή με N τιμές, δηλαδή μια μεταβλητή με δείκτες. Η δυνατότητα αυτή μας δίνεται στην FORTRAN με τους πίνακες.

Απλή μεταβλητή:	ένα όνομα, μια θέση στην μνήμη
Πίνακας (μεταβλητή με δείκτες):	ένα όνομα, πολλές διαδοχικές θέσεις στην μνήμη

Μπορούμε να δηλώσουμε τον πίνακα με έναν ή και περισσότερους δείκτες. Εάν τον δηλώσουμε με έναν δείκτη τότε είναι μονοδιάστατος. Εάν χρησιμοποιούμε M δείκτες, τότε ο πίνακας είναι M διαστάσεων. Ο μέγιστος αριθμός διαστάσεων είναι 7.

5.1 Δήλωση

Όλα τα στοιχεία ενός πίνακα είναι υποχρεωτικά του ίδιου τύπου. Συνεπακόλουθα, ο πίνακας δηλώνεται όπως και οι απλές μεταβλητές ανάλογα με το είδος των στοιχείων του: INTEGER, DOUBLE PRECISION, κτλ. Η μόνη διαφορά είναι ότι πρέπει να δηλώσουμε τις διαστάσεις και το μέγεθός του. Για παράδειγμα ένας μονοδιάστατος πίνακας A με 100 στοιχεία διπλής ακρίβειας δηλώνεται ως:

```
DOUBLE PRECISION A(100)
```

- Όπως και στις απλές μεταβλητές, όλες οι δηλώσεις έρχονται πριν την από την πρώτη εκτελέσιμη εντολή του προγράμματος.
- Καθώς κατά την μετάφραση πρέπει είναι γνωστό το μέγεθος της μνήμης πού πρέπει να δεσμευθεί για την αποθήκευση των στοιχείων του πίνακα (και αυτό το μέγεθος να παραμένει αμετάβλητο κατά την εκτέλεση του προγράμματος), η δήλωση πρέπει να γίνεται είτε με απευθείας αριθμητική τιμή (όπως στο παραπάνω παράδειγμα), είτε με σταθερές. Ένας τρόπος είναι

χρησιμοποιώντας την εντολή PARAMETER, οποία μετατρέπει έναν ακέραιο σε σταθερά. Για παράδειγμα, για να δηλώσουμε έναν μονοδιάστατο πίνακα A με 100 στοιχεία διπλής ακρίβειας και έναν μονοδιάστατο πίνακα B με 50 ακέραια στοιχεία:

```
INTEGER N, M
PARAMETER (N=100, M=50)
DOUBLE PRECISION A(N)          πίνακας A με 100 στοιχεία διπλής ακρίβειας
INTEGER B(M)                   πίνακας B με 50 στοιχεία διπλής ακρίβειας
```

- Εναλλακτικά, μπορούμε πρώτα να δηλώσουμε ότι ο A είναι διπλής ακρίβειας, και μετά σε δεύτερη γραμμή να δηλώσουμε ότι είναι πίνακας χρησιμοποιώντας την εντολή DIMENSION:

```
DOUBLE PRECISION A
DIMENSION A(100)               πίνακας A με 100 στοιχεία διπλής ακρίβειας
```

- Η FORTRAN μας δίνει την δυνατότητα να έχουμε και αρνητικούς δείκτες, ή γενικότερα να ορίζουμε έναν ελάχιστο δείκτη και έναν μέγιστο, χρησιμοποιώντας την άνω–κάτω τελεία. Για παράδειγμα εάν θέλουμε ο δείκτης του πίνακα A να παίρνει τιμές από -50 έως 100:

```
DOUBLE PRECISION A(-50:100)    πίνακας A με 151 στοιχεία διπλής ακρίβειας
```

ή ισοδύναμα

```
INTEGER N, M
PARAMETER (N=100, M=-50)       μετατρέπουμε τα N, M σε σταθερές
DOUBLE PRECISION A(M:N)        πίνακας A με 151 στοιχεία διπλής ακρίβειας
```

- Εάν χρησιμοποιείται ένα όνομα σε κάποιον πίνακα, δεν μπορεί να ξαναχρησιμοποιηθεί για κάποια άλλη μεταβλητή (ο πίνακας δεν παύει να είναι μεταβλητή και ο ίδιος, απλώς έχει και δείκτες).

5.2 Εισαγωγή τιμών

Τα στοιχεία του πίνακα τα καλούμε χρησιμοποιώντας ακέραιους δείκτες. Για παράδειγμα το στοιχείο 5 του A το καλούμε ως A(5). Μπορούμε να χρησιμοποιήσουμε ως δείκτη μια απλή ακέραια μεταβλητή. Για παράδειγμα το στοιχείο J του A το καλούμε ως A(J), αρκεί να έχουμε πιο πριν αναθέσει μια τιμή στο J. Στο παραπάνω παράδειγμα με δήλωση A(100), ο πίνακας A έχει N=100 ακέραια στοιχεία, τα A(1), A(2), A(3),...A(99), A(100).

Έστω ένας ακέραιος πίνακας A με πέντε στοιχεία, και θέλουμε για παράδειγμα να τους δώσουμε τιμές 10, 20, 30, 40 και 50 αντίστοιχα. Μπορούμε να δώσουμε τις νέες τιμές στα στοιχεία του πίνακα με τρεις τρόπους:

```
INTEGER A(5)                   πίνακας A με 5 ακέραια στοιχεία
```

- **Απευθείας ανάθεση:**

```
A(1) = 10
A(2) = 20
A(3) = 30
A(4) = 40
A(5) = 50
```

ή με την χρήση κατάλληλου βρόχου DO

```
DO I = 1, 5
   A(I) = 10 * I
END DO
```

- **Εντολή DATA:**
  Χρήσιμη όταν οι διαστάσεις είναι πολλές, και οι αρχικές τιμές των πινάκων είναι οι ίδιες κάθε φορά που εκτελείται το πρόγραμμα (αποφεύγουμε την χρονοβόρα διαδικασία πληκτρολόγησης, και αποφεύγουμε τα λάθη). Μετά την εντολή DATA γράφουμε το όνομα του πίνακα, και μετά ανάμεσα σε καθέτους και χωρισμένες με κόμματα γράφουμε τις τιμές του πίνακα:

  ```
  DATA A / 10, 20, 30, 40, 50/
  ```

  Προσοχή: όταν χρησιμοποιούμε την εντολή DATA πρέπει να δίνουμε όλα τα στοιχεία του πίνακα. Εάν θέλουμε να δώσουμε τιμές σε μερικά από τα στοιχεία ενός πίνακα πρέπει να χρησιμοποιήσουμε κάποια από τις άλλες δύο μεθόδους.

  Η εντολή DATA μπορεί να βρίσκεται σε οποιοδήποτε σημείο του προγράμματος, αλλά καλύτερα να μπαίνει στην αρχή αμέσως μετά τις δηλώσεις.

  Η εντολή DATA μπορεί να χρησιμοποιηθεί συγχρόνως για πολλές μεταβλητές, απλές και πίνακες, ακέραιους και διπλής ακρίβειας. Για παράδειγμα:

  ```
  INTEGER A(5), C
  DOUBLE PRECISION B(2)
  DATA A, B, C / 10, 20, 30, 40, 50, 1.5, 2.5, 100 /
  ```

  Σε αυτό το παράδειγμα εκτός του A, δίνουμε τις τιμές 1.5 και 2.5 στον B και την τιμή 100 στον C.

- **Εντολή εισόδου READ:**

`WRITE(*,*) 'ΕΙΣΑΓΕΤΕ 5 ΤΙΜΕΣ'`	εντολή προτροπής
`READ(*,*) A(1),A(2),A(3),A(4),A(5)`	το πρόγραμμα περιμένει 5 τιμές από το πληκτρολόγιο

  Οι τιμές εισάγονται από το πληκτρολόγιο χωρισμένες με κενά (όχι κόμματα). Εάν εισάγουμε όλα τα στοιχεία του πίνακα, μπορούμε σε συντομογραφία να γράψουμε μόνο το όνομά του:

`READ (*,*) A`	το πρόγραμμα περιμένει 5 τιμές από το πληκτρολόγιο

  Εάν όμως θέλουμε να εισάγουμε κάποια από τα στοιχεία του πίνακα μόνο, η πρώτη μορφή (όπου αναλυτικά τα γράφουμε ένα–ένα) είναι υποχρεωτική. Για παράδειγμα, για να εισάγουμε μόνο τις τρεις πρώτες τιμές:

```
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ 3 ΤΙΜΕΣ'        εντολή προτροπής
READ (*,*) A(1), A(2), A(3)           το πρόγραμμα περιμένει 3 τιμές από το πλη-
                                      κτρολόγιο
```

Εάν έχουμε πάρα πολλά στοιχεία, μπορούμε να χρησιμοποιήσουμε μια έναν βρόγχο DO. Για παράδειγμα:

```
INTEGER A(100)                        πίνακας A με 100 ακέραια στοιχεία
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ 50 ΤΙΜΕΣ'       εντολή προτροπής
DO I = 1, 50
  READ (*,*) A(I)                     το πρόγραμμα διαβάζει μια–μια τις 50 τιμές
                                      από το πληκτρολόγιο (τιμή-enter, τιμή-enter,
                                      τιμή-enter. ...)
END DO
```

ή με έμμεσο DO

```
READ (*,*) (A(I), I = 1, 50)          διαβάζει 50 τιμές (εισάγονται όλες μαζί σε
                                      μια γραμμή στο πληκτρολόγιο)
```

Το έμμεσο DO είναι και η προτιμητέα μέθοδος, εκτός και εάν είναι απαραίτητο να εισάγονται οι τιμές αυστηρώς μία–μία.

5.3 Εξαγωγή τιμών

Με παρόμοιο τρόπο που γίνεται η εισαγωγή, μπορούμε να εξάγουμε/εκτυπώσουμε τις τιμές ενός πίνακα. Εάν είναι λίγα τα στοιχεία μπορούμε να τα εκτυπώσουμε γράφοντάς τα ένα–ένα:

```
WRITE (*,*) A(1),A(2),A(3),A(4),A(5)  το πρόγραμμα εμφανίζει 5 τιμές στην οθόνη
```

Εάν είναι πολλά με έμμεσο DO

```
WRITE (*,*) (A(I), I = 1, 50)         εμφανίζει 50 τιμές στην οθόνη (σε μία γραμμή)
```

ή με άμεσο DO

```
DO I = 1, 50
  WRITE (*,*) A(I)                    το πρόγραμμα εμφανίζει μία–μία τις 50 τιμές
                                      την οθόνη (κάθε τιμή σε νέα γραμμή, δηλ. 50
                                      γραμμές)
END DO
```

5.4 Παραδείγματα

5.4.1 Μέσος όρος, μέγιστη και ελάχιστη τιμή

Πρόγραμμα που ζητάει την εισαγωγή δεδομένων, τα αποθηκεύει σε πίνακα, και υπολογίζει και τυπώνει τον μέσο όρο, το μέγιστο στοιχείο και το ελάχιστο στοιχείο.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM AVER_MIN_MAX`	
`IMPLICIT NONE`	
`INTEGER I, N, NMAX`	
`PARAMETER ( NMAX=1000 )`	
`DOUBLE PRECISION A(NMAX)`	πίνακας A με 1000 στοιχεία διπλής ακρίβειας
`DOUBLE PRECISION AVER, AMIN, AMAX`	
`WRITE(*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν τη μέγιστη τιμή NMAX
`  WRITE (*,*) 'ΛΑΘΟΣ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΑ ΝΟΥΜΕΡΑ'`	προτροπή
`READ (*,*) (A(I),I=1,N)`	εισαγωγή στοιχείων (σε μια γραμμή)
`AVER = 0`	θέτουμε τον αθροιστή στο μηδέν
`DO I = 1, N`	
`  AVER = AVER + A(I)`	αθροίζουμε όλα τα στοιχεία
`END DO`	
`AVER = AVER / N`	διαιρούμε με τον αριθμό των στοιχείων
`AMAX = A(1)`	αρχικά θέτουμε ως μέγιστο το πρώτο στοιχείο
`DO I = 2, N`	διατρέχουμε όλα τα υπόλοιπα στοιχεία
`  IF (A(I).GT.AMAX) AMAX = A(I)`	εάν κάποιο είναι μεγαλύτερο, το κατοχυρώνουμε
`END DO`	
`AMIN = A(1)`	αρχικά θέτουμε ως ελάχιστο το πρώτο στοιχείο
`DO I = 2, N`	διατρέχουμε όλα τα υπόλοιπα στοιχεία
`  IF (A(I).LT.AMIN) AMIN = A(I)`	εάν κάποιο είναι μικρότερο, το κατοχυρώνουμε
`END DO`	
`WRITE(*,*)'Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ:', AVER`	
`WRITE(*,*)'ΤΟ ΜΕΓΙΣΤΟ ΕΙΝΑΙ:  ', AMAX`	

```
WRITE(*,*)'ΤΟ ΕΛΑΧΙΣΤΟ ΕΙΝΑΙ: ', AMIN
END
```

5.4.2 Γραμμική μέθοδος ελαχίστων τετραγώνων

Δοθέντων N σημείων του επιπέδου (x_i, y_i), για $i = 1, 2, \ldots, N$, η βέλτιστη ευθεία $y = a + bx$ που περνάει όσο το δυνατόν κοντύτερα από όλα τα σημεία υπολογίζεται ως εξής:

$$a = \frac{s_{xx}s_y - s_y s_{xy}}{N s_{xx} - s_x s_x} \quad \text{και} \quad b = \frac{N s_{xy} - s_x s_y}{N s_{xx} - s_x s_x}$$

όπου

$$s_x = \sum_{i=1}^{N} x_i \qquad s_y = \sum_{i=1}^{N} y_i \qquad s_{xx} = \sum_{i=1}^{N} x_i^2 \qquad s_{xy} = \sum_{i=1}^{N} x_i y_i$$

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM LLSF`	
`IMPLICIT NONE`	
`INTEGER I, N, NMAX`	
`PARAMETER ( NMAX=1000 )`	
`DOUBLE PRECISION X(NMAX), Y(NMAX)`	πίνακες X, Y με 1000 στοιχεία
`DOUBLE PRECISION SX, SY, SXX, SXY`	
`DOUBLE PRECISION A, B`	
`WRITE(*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν το μέγιστο NMAX
`  WRITE (*,*) 'ΛΑΘΟΣ-ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ X ΣΥΝΙΣΤΩΣΕΣ'`	προτροπή
`READ (*,*) (X(I),I=1,N)`	εισαγωγή στοιχείων (σε μια γραμμή)
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ Y ΣΥΝΙΣΤΩΣΕΣ'`	προτροπή
`READ (*,*) (Y(I),I=1,N)`	εισαγωγή στοιχείων (σε μια γραμμή)
`SX = 0`	θέτουμε αθροιστή στο μηδέν
`SY = 0`	θέτουμε αθροιστή στο μηδέν
`SXX = 0`	θέτουμε αθροιστή στο μηδέν
`SXY = 0`	θέτουμε αθροιστή στο μηδέν
`DO I = 1, N`	
`  SX = SX + X(I)`	αθροίζουμε όλα τα X
`  SY = SY + Y(I)`	αθροίζουμε όλα τα Y
`  SXX = SXX + X(I)**2`	αθροίζουμε όλα τα X**2
`  SXY = SXY + X(I)*Y(I)`	αθροίζουμε όλα τα X*Y

```
END DO
A = (SXX*SY-SX*SXY)/(N * SXX - SX*SX)      ο σταθερός όρος της βέλτιστης ευθείας
B = (N*SXY - SX*SY)/(N * SXX - SX*SX)      η κλίση της βέλτιστης ευθείας
WRITE (*,*) 'Ο ΣΤΑΘΕΡΟΣ ΟΡΟΣ ΕΙΝΑΙ:', A
WRITE (*,*) 'Η ΚΛΙΣΗ ΕΙΝΑΙ:       ', B
END
```

5.4.3 Ταξινόμηση

Αναδιατάξτε έναν μονοδιάστατο πίνακα σε αύξουσα σειρά.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM SORT`	
`IMPLICIT NONE`	
`INTEGER I, J, N, NMAX`	
`PARAMETER ( NMAX=1000 )`	
`DOUBLE PRECISION X(NMAX)`	Πίνακας X με 1000 στοιχεία διπλής ακρίβειας.
`DOUBLE PRECISION XMIN, TEMP`	
`WRITE (*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'`	Προτροπή.
`READ (*,*) N`	Ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	Έλεγχος να μην ξεπερνούν το μέγιστο NMAX
`  WRITE (*,*) 'ΛΑΘΟΣ-ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΑ ΝΟΥΜΕΡΑ'`	προτροπή
`READ (*,*) (X(I),I=1,N)`	εισαγωγή στοιχείων (σε μια γραμμή)
`DO J = 1, N-1`	διατρέχουμε τις θέσεις μια–μια, στην πρώτη θα βάλουμε το μικρότερο, στη δεύτερη το δεύτερο μικρότερο κοκ
`  XMIN = X(J)`	για την θέση J θέτουμε αρχικά ως μικρότερο το X(J)
`  K = J`	το K θα κρατάει την θέση του μικρότερου (από J και πάνω)
`  DO I = J+1, N`	διατρέχουμε όλα τα στοιχεία πάνω από το J
`    IF (X(I).LT.XMIN) THEN`	
`      XMIN = X(I)`	εάν κάποιο είναι μικρότερο, το κατοχυρώνουμε
`      K = I`	κατοχυρώνουμε την θέση του μικρότερου
`    ENDIF`	
`  END DO`	

```
   IF (K.NE.J) THEN                          εάν βρέθηκε στο K≠J μικρότερο στοιχείο
                                             από το J
      TEMP = X(J)
      X(J) = X(K)                            αντάλλαξέ τα μεταξύ τους
      X(K) = TEMP
   ENDIF
END DO
WRITE (*,*) 'ΤΑ ΤΑΞΙΝΟΜΗΜΕΝΑ ΝΟΥΜΕΡΑ ΕΙΝΑΙ:'
WRITE (*,*) (X(I),I=1,N)                     εξαγωγή στοιχείων (σε μια γραμμή)
END
```

5.4.4 Αριθμός σημείων μέσα σε κύκλο

Δοθέντων N σημείων του επιπέδου (x_i, y_i), για $i = 1, 2, \ldots, N$, πόσα είναι μέσα σε κύκλο ακτίνας r και κέντρου x_0, y_0 ;

Κώδικας Fortran | *Σχολιασμός*

```
PROGRAM POINTS_IN_CIRCLE
IMPLICIT NONE
INTEGER I, N, NMAX, NUM
PARAMETER ( NMAX=1000 )
DOUBLE PRECISION X(NMAX), Y(NMAX)            πίνακες X, Y με 1000 στοιχεία
DOUBLE PRECISION R, X0, Y0
WRITE (*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'     προτροπή
READ (*,*) N                                 ο αριθμός των στοιχείων
IF (N.GT.NMAX) THEN                          έλεγχος να μην ξεπερνούν το μέγιστο
   WRITE (*,*) 'ΛΑΘΟΣ-ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'
   STOP                                      εάν το ξεπερνούν τερματίζουμε
ENDIF
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ X ΣΥΝΙΣΤΩΣΕΣ'     προτροπή
READ (*,*) (X(I), I = 1, N)                  εισαγωγή στοιχείων (σε μια γραμμή)
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ Y ΣΥΝΙΣΤΩΣΕΣ'     προτροπή
READ (*,*) (Y(I), I = 1, N)                  εισαγωγή στοιχείων (σε μια γραμμή)
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΚΕΝΤΡΟ ΚΑΙ ΑΚΤΙΝΑ ΚΥΚΛΟΥ'
READ (*,*) X0, Y0, R                         εισαγωγή στοιχείων (σε μια γραμμή)
NUM = 0                                      θέτουμε αθροιστή στο μηδέν
DO I = 1, N
   DIST=SQRT((X(I)-X0)**2+(Y(I)-Y0)**2 )     απόσταση σημείου I από κέντρο του κύκλου
   IF (DIST .LE. R) NUM = NUM + 1            μέσα στον κύκλο αν μικρότερη της ακτίνας
END DO
WRITE(*,*) 'Ο ΑΡΙΘΜΟΣ ΤΩΝ ΣΗΜΕΙΩΝ ΜΕΣΑ ΣΤΟΝ ΚΥΚΛΟ ΕΙΝΑΙ:' , NUM
END
```

Κεφάλαιο 6

Πολυδιάστατοι πίνακες

Μπορούμε να ορίσουμε και πίνακες πολλών διαστάσεων, αναλόγως των αναγκών του προγράμματος.

6.1 Δήλωση

Όπως και πριν, με την διαφορά ότι οι επιπλέον διαστάσεις χωρίζονται με κόμμα. Παραδείγματα:

```
INTEGER A(3, 3)                    ακέραιος πίνακας 3x3, σύνολο 9 στοιχεία
INTEGER B(0:3, 0:3)                ακέραιος πίνακας 4x4, σύνολο 16 στοιχεία
INTEGER C(-3:3, -3:3, -3:3)        ακέραιος πίνακας 7x7x7, σύνολο 343 στοιχεία
```

ή με την χρήση σταθερών

```
INTEGER N, M
PARAMETER(N = 3, M = -3)
INTEGER A(N, N)                    ακέραιος πίνακας 3x3, σύνολο 9 στοιχεία
INTEGER B(0:N, 0:N)                ακέραιος πίνακας 4x4, σύνολο 16 στοιχεία
INTEGER C(M:N, M:N, M:N)           ακέραιος πίνακας 7x7x7, σύνολο 343 στοιχεία
```

ή επίσης χρησιμοποιώντας την εντολή DIMENSION

```
INTEGER A, B, C
DIMENSION A(3, 3), B(0:3, 0:3), C(-3:3, -3:3, -3:3)
```

Στους διδιάστατους πίνακες ο πρώτος δείκτης αντιστοιχεί στη γραμμή και ο δεύτερος στην στήλη. Για παράδειγμα το στοιχείο A(2,3) είναι η δεύτερη γραμμή, τρίτη στήλη.

6.2 Αποθήκευση τιμών

Οι τιμές ενός πίνακα καταλαμβάνουν διαδοχικές θέσεις στην μνήμη. Για έναν μονοδιάστατο πίνακα είναι εύκολο, πχ ο A(-50:50) καταλαμβάνει 101 διαδοχικές θέσεις στην μνήμη, πρώτα ο A(-50), μετά ο A(-49) ... μέχρι τέλος τον A(50).

Για διδιάστατους πίνακες αποθηκεύονται κατά σειρά οι στήλες, ή αλλιώς διαδοχικά διανύουμε τον πρώτο δείκτη και μετά τον δεύτερο. Πχ ο πίνακας A(3,3) αποθηκεύεται κατά σειρά τα στοιχεία A(1,1), A(2,1), A(3,1), A(1,2), A(2,2), A(3,2), A(1,3), A(2,3), A(3,3). Ουσιαστικά βάζουμε τις στήλες του πίνακα την μια κάτω από την άλλη. Για πολυδιάστατους πίνακες η γενίκευση είναι προφανής. Διανύουμε πρώτα τον πρώτο δείκτη, μετά τον δεύτερο κοκ.

6.3 Εισαγωγή τιμών

Έστω ένας διδιάστατος ακέραιος πίνακας A(2, 2) (με 4 στοιχεία), και θέλουμε για παράδειγμα να του δώσουμε τιμές 10, 20, 30 και 40. Όπως και στους μονοδιάστατους πίνακες, μπορούμε να δώσουμε τις νέες τιμές στα στοιχεία του πίνακα με τρεις τρόπους:

```
INTEGER A(2, 2)                          διδιάστατος πίνακας A με 4 ακέραια στοιχεία
```

- **Απευθείας ανάθεση:**

```
A(1, 1) = 10                             διατρέχουμε πρώτα την πρώτη στήλη
A(2, 1) = 20
A(1, 2) = 30                             και κατόπιν την δεύτερη στήλη
A(2, 2) = 40
```

ή με την χρήση κατάλληλου βρόχου DO

```
DO J = 1, 2                              διατρέχουμε τις στήλες
   DO I = 1, 2                           διατρέχουμε τις γραμμές της κάθε στήλης
     A(I, J)=( (J-1)*2 + I ) * 10
   END DO
END DO
```

- **Εντολή DATA:**

Όπως και πριν, μετά την εντολή DATA γράφουμε το όνομα του πίνακα, και μετά ανάμεσα σε καθέτους και χωρισμένες με κόμματα γράφουμε τις τιμές του πίνακα:

```
DATA A / 10, 20, 30, 40/
```

Προσοχή: πρέπει να θυμόμαστε την αυστηρή σειρά πού πρέπει να δίνονται τα στοιχεία, δηλαδή την μία στήλη μετά την άλλη.

Όπως και πριν, όταν χρησιμοποιούμε την εντολή DATA πρέπει να δίνουμε όλα τα στοιχεία του πίνακα. Η εντολή DATA μπορεί να βρίσκεται σε οποιοδήποτε σημείο του προγράμματος, αλλά συνιστάται να μπαίνει στην αρχή αμέσως μετά τις δηλώσεις. Η εντολή DATA μπορεί να χρησιμοποιηθεί συγχρόνως για πολλές μεταβλητές, απλές και πίνακες, ακέραιους και διπλής ακρίβειας. Για παράδειγμα:

```
INTEGER A(2,2), C                        DOUBLE PRECISION B(2)
```

Στο παραπάνω παράδειγμα, εκτός του A, δίνουμε τις τιμές 1.5 και 2.5 στον B και την τιμή 100 στον C.

- **Εντολή εισόδου READ:**

```
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ 4 ΤΙΜΕΣ'          εντολή προτροπής
READ(*,*)A(1,1),A(2,1),A(1,2),A(2,2)    το πρόγραμμα περιμένει 4 τιμές από το
                                         πληκτρολόγιο
```

Οι τιμές εισάγονται από το πληκτρολόγιο χωρισμένες με κενά (όχι κόμματα). Εάν εισάγουμε όλα τα στοιχεία του πίνακα, μπορούμε σε συντομογραφία να γράψουμε μόνο το όνομά του (και πάλι προσοχή στη σειρά εισαγωγής):

```
READ (*,*) A                             το πρόγραμμα περιμένει 5 τιμές από το πλη-
                                         κτρολόγιο
```

Εάν έχουμε πάρα πολλά στοιχεία, μπορούμε να χρησιμοποιήσουμε μια έναν βρόγχο DO. Για παράδειγμα για πίνακα 10x10:

```
INTEGER A(10,10)                         πίνακας A με 100 ακέραια στοιχεία
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ 100 ΤΙΜΕΣ'         εντολή προτροπής
DO J = 1, 10
   DO I = 1, 10
      READ (*,*) A(I, J)                 το πρόγραμμα διαβάζει μια–μια τις 100 τιμές
                                         από το πληκτρολόγιο (τιμή-enter, τιμή-enter,
                                         τιμή-enter)
   END DO
END DO
```

ή με έμμεσο DO

```
DO J = 1, 10
   WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ 10 ΤΙΜΕΣ ΤΗΣ ΣΤΗΛΗΣ',J
   READ (*,*) (A(I, J), I = 1, 10)       διαβάζει 10 τιμές (εισάγονται όλες μαζί
                                         στο πληκτρολόγιο)
END DO
```

ή με διπλό έμμεσο DO

```
READ (*,*) ((A(I,J),I=1,10),J=1,10)      διαβάζει 100 τιμές (εισάγονται όλες
                                         μαζί)
```

Εάν αντίθετα θέλουμε να εισάγουμε μία–μία τις γραμμές αντί για μία–μία στις στήλες, το έμμεσο DO γράφεται:

```
DO I = 1, 10
   WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ 10 ΤΙΜΕΣ ΤΗΣ ΓΡΑΜΜΗΣ', I
   READ (*,*) (A(I, J), J = 1, 10)       διαβάζει 10 τιμές (εισάγονται όλες μαζί
                                         στο πληκτρολόγιο)
END DO
```

Το έμμεσο DO είναι και η προτιμητέα μέθοδος, εκτός και εάν είναι απαραίτητο να εισάγονται οι τιμές αυστηρώς μία–μία.

Παρόμοια με τα παραπάνω εισάγονται οι τιμές σε πίνακες μεγαλύτερων διαστάσεων.

6.4 Εξαγωγή τιμών

Με παρόμοιο τρόπο που γίνεται η εισαγωγή, μπορούμε να εξάγουμε/εκτυπώσουμε τις τιμές ενός πίνακα. Εάν είναι λίγα τα στοιχεία μπορούμε να τα εκτυπώσουμε γράφοντάς τα ένα–ένα

```
WRITE (*,*) A(1,1),A(2,1),A(1,2),A(2,2)
```
το πρόγραμμα εμφανίζει 4 τιμές στην οθόνη

Εάν είναι πολλά με έμμεσο DO, εμφανίζοντας μια–μια τις στήλες

```
INTEGER A(10, 10)
DO J = 1, 10                              εμφανίζονται 10 γραμμές στην οθόνη
   WRITE (*,*) (A(I, J), I = 1, 10)       με 10 τιμές στην κάθε γραμμή
END DO
```

ή μια–μια τις γραμμές

```
DO I = 1, 10
   WRITE (*,*) (A(I, J), J = 1,10)        εμφανίζονται 10 γραμμές στην οθόνη
END DO                                    με 10 τιμές στην κάθε γραμμή
```

Για παράδειγμα, ας δώσουμε τιμές σε έναν πίνακα 3x3 έτσι ώστε το στοιχείο (I,J) να παίρνει την τιμή 10 * I + J:

```
INTEGER A(3, 3)
DO J = 1, 3                               διατρέχουμε τις στήλες
   DO I = 1, 3                            διατρέχουμε τις γραμμές της κάθε στήλης
      A(I, J)= 10 * I + J
   END DO
END DO
```

Ο πίνακας που δημιουργήσαμε είναι:

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}$$

Εάν θέλουμε να τον εκτυπώσουμε και να εμφανιστεί με την ίδια μορφή, πρέπει να εκτυπώσουμε μια–μια τις γραμμές, δηλαδή:

```
DO I = 1, 3
   WRITE(*,*) (A(I, J), J = 1,3)          εμφανίζονται 3 γραμμές στην οθόνη
END DO
```

Αντίθετα, εάν θέλουμε να εμφανίσουμε μια–μια τις στήλες, γράφουμε

```
DO J = 1, 3
   WRITE(*,*) (A(I, J), I = 1,3)          εμφανίζονται 3 γραμμές στην οθόνη
END DO
```

και στην οθόνη θα εμφανιστεί ο αντι-μετατεθιμένος πίνακας:

```
11  21  31
12  22  32
13  23  33
```

6.5 Παραδείγματα

6.5.1 Δημιουργία και ίχνος πίνακα NxN

Δημιουργία πίνακα με στοιχεία (I, J)=sin (2π/(I+J)). Το ίχνος είναι το άθροισμα των στοιχείων της διαγωνίου.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM MATRIX_TRACE`	
`IMPLICIT NONE`	
`INTEGER I, J, N, NMAX`	
`PARAMETER ( NMAX=100 )`	
`DOUBLE PRECISION A(NMAX, NMAX)`	πίνακας A με 10000 στοιχεία
`DOUBLE PRECISION PI, TRACE`	
`WRITE(*,*)'ΠΟΙΟ ΤΟ ΜΕΓΕΘΟΣ ΤΟΥ ΠΙΝΑΚΑ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν το μέγιστο NMAX
`  WRITE (*,*) 'ΛΑΘΟΣ-ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`PI = ACOS(-1.0)`	
`DO J = 1, N`	
`  DO I = 1, N`	
`    A(I, J) = SIN( (2*PI) / (I+J) )`	
`  END DO`	
`END DO`	
`TRACE = 0`	θέτουμε αθροιστή στο μηδέν
`DO I = 1, N`	
`  TRACE = TRACE + A(I, I)`	προσθέτουμε όλα τα στοιχεία της διαγωνίου
`END DO`	
`WRITE (*,*) 'Ο ΠΙΝΑΚΑΣ ΕΙΝΑΙ:'`	
`DO I = 1, N`	
`  WRITE(*,*) (A(I, J), J = 1, N)`	εκτυπώνουμε τον πίνακα, μια-μια τις γραμμές
`END DO`	
`WRITE(*,*)'ΤΟ ΙΧΝΟΣ ΤΟΥ ΠΙΝΑΚΑ ΕΙΝΑΙ:',TRACE`	
`END`	

6.5.2 Πολλαπλασιασμός δύο πινάκων

Το στοιχείο (I, J) το γινομένου, είναι το εσωτερικό γινόμενο της I γραμμής του πρώτου πίνακα επί τη J στήλη του δεύτερου πίνακα.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM MATRIX_MULTIPLY`	
`IMPLICIT NONE`	
`INTEGER I, J, K, N, NMAX`	
`PARAMETER ( NMAX=100 )`	
`DOUBLE PRECISION A(NMAX,NMAX)`	πίνακας A με 10000 στοιχεία
`DOUBLE PRECISION B(NMAX,NMAX)`	πίνακας B με 10000 στοιχεία
`DOUBLE PRECISION C(NMAX,NMAX)`	πίνακας C με 10000 στοιχεία, για το γινόμενο A*B
`WRITE(*,*)'ΠΟΙΟ ΤΟ ΜΕΓΕΘΟΣ ΤΩΝ ΠΙΝΑΚΩΝ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν το μέγιστο
`  WRITE(*,*) 'ΛΑΘΟΣ - ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`DO J = 1, N`	
`  WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ ΤΙΜΕΣ ΤΗΣ ΣΤΗΛΗΣ', J, 'ΤΟΥ ΠΙΝΑΚΑ A'`	
`  READ (*,*) (A(I, J),I=1,N)`	διαβάζει N τιμές
`END DO`	
`DO J = 1, N`	
`  WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ ΤΙΜΕΣ ΤΗΣ ΣΤΗΛΗΣ', J, 'ΤΟΥ ΠΙΝΑΚΑ B'`	
`  READ (*,*) (B(I, J),I=1,N)`	διαβάζει N τιμές
`END DO`	
`DO J = 1, N`	για κάθε στήλη J του γινομένου C
`  DO I = 1, N`	για κάθε γραμμή I του γινομένου C
`    C(I, J) = 0`	
`    DO K = 1, N`	διατρέχουμε την γραμμή I του A και την στήλη J του B
`      C(I,J) = C(I,J) + A(I,K)*B(K,J)`	προσθέτουμε τα αντίστοιχα γινόμενα
`    END DO`	δημιουργώντας το στοιχείο (I, J) του C
`  END DO`	
`END DO`	
`WRITE(*,*)'ΤΟ ΓΙΝΟΜΕΝΟ ΤΩΝ ΠΙΝΑΚΩΝ A ΚΑΙ B ΕΙΝΑΙ:'`	
`DO I = 1, N`	
`WRITE(*,*) (C(I, J), J = 1, N)`	εκτυπώνουμε μια–μια τις γραμμές
`END DO`	
`END`	

Κεφάλαιο 7

Υποπρογράμματα: Συναρτήσεις

Πολλές φορές χρειάζεται να ξαναγράψουμε το ίδιο κομμάτι κώδικα πολλές φορές. Για να το αποφύγουμε χρησιμοποιούμε υπορουτίνες και συναρτήσεις. Μια συνάρτηση είναι μια ομαδοποίηση εντολών. Έχει μια λίστα μεταβλητών στην οποία μπαίνουν οι εισερχόμενες μεταβλητές (τυπικές παράμετροι), δηλαδή οι μεταβλητές οι οποίες είναι απαραίτητες για την εκτέλεση των εντολών. Το αποτέλεσμα των εντολών πρέπει να είναι μια μεταβλητή μόνο, η οποία γυρνάει κατά την έξοδο. Το όνομα της μεταβλητής εξόδου είναι το όνομα της συνάρτησης.

7.1 Δήλωση

Η δήλωση μιας συνάρτησης γίνεται μετά το τέλος του προγράμματος (μετά την εντολή END) με την χρήση της εντολής FUNCTION. Καθώς στο όνομα της συνάρτησης συνυπάρχει και η επιστρεφόμενη μεταβλητή, θα πρέπει κατά την δήλωση να δηλώσουμε και τον τύπο της μεταβλητής που γυρνάει ως αποτέλεσμα (δηλαδή τον τύπο της συνάρτησης). Αποφεύγουμε να χρησιμοποιούμε κοινά ονόματα με συναρτήσεις βιβλιοθήκης ή άλλα υποπρογράμματα, ή άλλες απλές μεταβλητές. Το όνομα της κάθε συνάρτησης πρέπει να είναι μοναδικό.

«ΤΥΠΟΣ» `FUNCTION` *«ΟΝΟΜΑ»* (*«ΤΥΠΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ»*)	
`IMPLICIT NONE`	
ΔΗΛΩΣΕΙΣ ΜΕΤΑΒΛΗΤΩΝ	τυπικές μεταβλητές εισόδου, καθώς και άλλες τοπικές
⋮	
ΚΩΔΙΚΑΣ	όλες οι εντολές που θέλουμε να εκτελεστούν
⋮	
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

Μέσα στις παρενθέσεις είναι η λίστα των μεταβλητών (τυπικές παράμετροι). Εδώ μπαίνουν οι μεταβλητές (χωρισμένες από κόμματα) που εισέρχονται από το πρόγραμμα. Δεν υπάρχει περιορισμός

πόσα και τι. Μπορούμε να έχουμε σύνολο μέχρι και 256 μεταβλητές. Οι μεταβλητές αυτές ονομάζονται «τυπικές» και ήδη υπάρχουν στο κυρίως πρόγραμμα. Παρ' όλα αυτά, μέσα στην συνάρτηση πρέπει να ξαναδηλώσουμε τι είδους μεταβλητή είναι η κάθε μία. Όπως και στο κυρίως, ξεκινάμε με το IMPLICIT NONE, και μετά δηλώνουμε. Μέσα στην συνάρτηση είναι σαν και ξεκινάμε νέο πρόγραμμα.

Όταν τελειώσουν οι εντολές γυρνάει πίσω στο κυρίως πρόγραμμα με την εντολή RETURN. Το σώμα τον εντολών της συνάρτησης τελειώνει με το END. Εάν χρειαστεί να γυρίσει στο κυρίως πρόγραμμα σε διαφορετικά σημεία (πριν το τέλος των εντολών), αναλόγως για παράδειγμα με το αποτέλεσμα κάποιας εντολής IF, μπορούμε να έχουμε πολλές εντολές RETURN. Πάντα όμως πρέπει να υπάρχει ένα END στο τέλος όλων των εντολών. Εάν δεν βάλουμε RETURN, τότε η συνάρτηση γυρνάει στο κυρίως πρόγραμμα με την εκτέλεση του END.

Μέσα στην συνάρτηση οι μεταβλητές μπορούν να έχουν διαφορετικά ονόματα, αλλά προφανώς πρέπει να είναι του ίδιου τύπου, δηλαδή να υπάρχει μια προς μια αντιστοιχία για κάθε τυπική μεταβλητή όσο αναφορά τον τύπο της.

Μέσα στην συνάρτηση μπορεί να χρειαστεί να ορίσουμε επιπλέον μεταβλητές, τοπικές, οι οποίες θα χρησιμοποιηθούν προσωρινά για την εκτέλεση των εντολών. Μετά την εκτέλεση της συνάρτησης αυτές οι τοπικές μεταβλητές καταστρέφονται. Οι τυπικές μεταβλητές (μεταβλητές εισόδου) από την άλλη δεν καταστρέφονται. Οπότε χρειάζεται προσοχή: εάν η τιμή τους αλλάξει κατά την εκτέλεση των εντολών της συνάρτησης, η νέα τους τιμή θα διατηρηθεί ακόμα και μετά την επιστροφή στο κυρίως πρόγραμμα. Επίσης, θα πρέπει πριν το τέλος της συνάρτησης να δοθεί τιμή στη μεταβλητή εξόδου (δηλαδή το όνομα της συνάρτησης), αλλιώς δεν θα γυρίζει τίποτα όταν καλείται από το κυρίως πρόγραμμα.

Για παράδειγμα, έστω ότι επιθυμούμε μια συνάρτηση για τον υπολογισμό του μέτρου ενός διδιάστατου διανύσματος. Στην είσοδο θα πρέπει να βάλουμε τις X και Y συνιστώσες του διανύσματος, και στην έξοδο να πάρουμε το μέτρο. Και οι τρεις μεταβλητές θα πρέπει να είναι διπλής ακρίβειας. Θα ονομάσουμε την συνάρτησή μας MAGNITUDE:

```
DOUBLE PRECISION FUNCTION MAGNITUDE ( X, Y )
IMPLICIT NONE
DOUBLE PRECISION X, Y                    δήλωση τυπικών μεταβλητών
MAGNITUDE = SQRT(X**2 + Y**2)            εντολές συνάρτησης
RETURN                                   επιστροφή στο κυρίως πρόγραμμα
END
```

7.1.1 Κλήση

Στο παραπάνω παράδειγμα καλούμε μια άλλη συνάρτηση, την εσωτερική συνάρτηση SQRT. Στην πράξη μια συνάρτηση που κατασκευάζουμε χρησιμοποιείται όπως και οι εσωτερικές συναρτήσεις. Την καλούμε στο κυρίως πρόγραμμα με τον ίδιο τρόπο, ή την καλούμε μέσα σε άλλη συνάρτηση που έχουμε κατασκευάσει, κοκ.

Καθώς το όνομα της συνάρτησης χρησιμοποιείται και ως μεταβλητή, θα πρέπει να δηλώνεται στο κυρίως πρόγραμμα όπως και όλες οι απλές μεταβλητές.

Ως παράδειγμα, ας δούμε το πλήρες πρόγραμμα που θα χρησιμοποιούσε την παραπάνω συνάρτηση:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM VECTOR2D`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, Y, M, MAGNITUDE`	δήλωση μεταβλητών (απλών και συναρτήσεων)
`WRITE(*,*)'ΕΙΣΑΓΕΤΕ X ΚΑΙ Y ΣΥΝΙΣΤΩΣΕΣ;'`	προτροπή
`READ (*,*) X, Y`	διάβασμα συνιστωσών
`M = MAGNITUDE(X, Y)`	καλούμε την συνάρτηση, και αποθηκεύουμε την τιμή της στην M
`WRITE(*,*) 'ΤΟ ΜΕΤΡΟ ΤΟΥ ΔΙΑΝΥΣΜΑΤΟΣ ΕΙΝΑΙ',M`	
`END`	
`DOUBLE PRECISION FUNCTION MAGNITUDE(X, Y)`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, Y`	δήλωση τυπικών μεταβλητών
`MAGNITUDE = SQRT(X**2 + Y**2)`	εντολές συνάρτησης
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

Στο παραπάνω παράδειγμα θα μπορούσαμε ισοδύναμα να καλούσαμε την συνάρτηση κατά την εκτύπωση:

```
WRITE (*,*) 'ΤΟ ΜΕΤΡΟ ΤΟΥ ΔΙΑΝΥΣΜΑΤΟΣ ΕΙΝΑΙ', MAGNITUDE(X, Y)
```

Έτσι όμως δεν σώζεται η τιμή του μέτρου. Αυτό λοιπόν συνίσταται μόνο όταν δεν θα χρειαστούμε το μέτρο για κανένα μελλοντικό υπολογισμό.

Στην είσοδο μιας συνάρτησης μπορούμε να έχουμε και πίνακες. Στη λίστα εισόδου γράφουμε μόνο το όνομα του πίνακα, και μέσα στην συνάρτηση δηλώνουμε το μέγεθος και τις διαστάσεις του (ουσιαστικά αυτό που κάνουμε είναι να δίνουμε την διεύθυνση στην μνήμη του πρώτου στοιχείου του πίνακα, και καθώς όλα τα στοιχεία ενός πίνακα είναι αποθηκευμένα διαδοχικά στην μνήμη, το πρόγραμμα κατόπιν ξέρει πως να τα βρει). Το παραπάνω παράδειγμα με χρήση πινάκων, μπορεί εύκολα να γενικευθεί για διανύσματα N διαστάσεων:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM VECTOR_ND`	
`IMPLICIT NONE`	
`INTEGER NMAX`	
`PARAMETER (NMAX = 100)`	
`DOUBLE PRECISION R(NMAX), M, MAGNITUDE_ND`	δήλωση μεταβλητών
`WRITE (*,*) 'ΠΟΙΑ Η ΔΙΑΣΤΑΣΗ ΤΟΥ ΔΙΑΝΥΣΜΑΤΟΣ;'`	
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν το μέγιστο NMAX

```
   WRITE (*,*) 'ΛΑΘΟΣ-ΠΙΟ ΜΙΚΡΗ ΔΙΑΣΤΑΣΗ'
   STOP                                          εάν το ξεπερνούν τερματίζουμε
ENDIF
WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ ΣΥΝΙΣΤΩΣΕΣ'          προτροπή
READ (*,*) (R(I), I = 1, N)                      εισαγωγή στοιχείων (σε μια γραμμή)
M = MAGNITUDE_ND(R, NMAX, N)                     καλούμε την συνάρτηση, αποθηκεύουμε την
                                                 τιμή στην M
WRITE (*,*) 'ΤΟ ΜΕΤΡΟ ΤΟΥ ΔΙΑΝΥΣΜΑΤΟΣ ΕΙΝΑΙ',M
END
```

```
DOUBLE PRECISION FUNCTION MAGNITUDE_ND(R, NMAX, N)
IMPLICIT NONE
INTEGER NMAX, N, I
DOUBLE PRECISION R(NMAX), MAG                    δήλωση τυπικών και τοπικών μεταβλητών
MAG = 0                                          θέτουμε τον αθροιστή στο μηδέν
DO I = 1, N
   MAGN = MAGN + R(I)**2                         παίρνουμε το άθροισμα των τετραγώνων
END DO
MAGNITUDE = SQRT( MAGN )                         το μέτρο του διανύσματος
RETURN                                           επιστροφή στο κυρίως πρόγραμμα
END
```

Στο παραπάνω παράδειγμα έχουμε την επιπλέον πολυπλοκότητα ότι πρέπει να εισάγουμε την μέγιστη διάσταση του πίνακα, καθώς και την διάσταση που θα κάνουμε τον υπολογισμό. Έχουμε όμως και το κέρδος ότι μια και μοναδική συνάρτηση μπορεί να χρησιμοποιηθεί για οποιαδήποτε διάσταση.

7.1.2 Διατήρηση τοπικών μεταβλητών

Σε πολλές περιπτώσεις μπορεί να χρειαστεί μέρος της εισόδου μιας συνάρτησης να πάρει την πρώτη της αρχική τιμή μέσα στην συνάρτηση, ή κάποιες από τις τοπικές μεταβλητές να διατηρούν την τιμή τους και μετά την έξοδο από την συνάρτηση, αλλά και όταν την ξανακαλούμε. Μπορούμε να κάνουμε όλα αυτά με την χρήση των εντολών DATA, SAVE, και COMMON. Θα εξετάσουμε εδώ τις δύο πρώτες, και θα αφήσουμε την εντολή COMMON για τις υπορουτίνες.

- Η εντολή DATA χρησιμοποιείται για να δώσει αρχική τιμή σε μια μεταβλητή.
- Η εντολή SAVE εξασφαλίζει ότι μια μεταβλητή διατηρεί την τιμή της (μένει στη μνήμη) κατά την διάρκεια διαδοχικών καλεσμάτων της συνάρτησης.
- Η εντολή DATA μόνη της δίνει την ίδια αρχική τιμή στην μεταβλητή κάθε φορά που καλείται η συνάρτηση.
- Η εντολή SAVE μόνη της σώζει την τιμή της μεταβλητής σε διαδοχικά καλέσματα.
- Οι δύο εντολές μαζί λειτουργούν ώστε η DATA να δίνει αρχική τιμή μόνο την πρώτη φορά που καλείται η συνάρτηση, και κατά τα διαδοχικά καλέσματα να σώζεται η νέα τιμή την κάθε φορά.

Σαν παράδειγμα, έστω μια συνάρτηση ονομαζόμενη AVERAGE για τον υπολογισμό του μέσου όρου, η οποία σώζει τον παλαιότερο μέσο όρο και με κάθε κλήση υπολογίζει τον νέο μέσο όρο. Για την αποθήκευση χρησιμοποιεί τις εντολές DATA, SAVE. Γράφουμε παρακάτω το πλήρες πρόγραμμα:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM AVERAGE_NUMBERS`	
`IMPLICIT NONE`	
`INTEGER I, N`	δήλωση μεταβλητών
`DOUBLE PRECISION X, AVERAGE`	δήλωση μεταβλητών και συνάρτησης
`WRITE(*,*)'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`DO I = 1, N`	
`  WRITE (*,*) 'ΔΩΣΤΕ ΤΟ ΕΠΟΜΕΝΟ ΝΟΥΜΕΡΟ;'`	
`  READ (*,*) X`	εισαγωγή του επόμενου στοιχείου.
`  WRITE(*,*)'ΝΕΟΣ ΜΕΣΟΣ ΟΡΟΣ:',AVERAGE(X)`	υπολογισμός και εκτύπωση
`END DO`	
`END`	
`DOUBLE PRECISION FUNCTION AVERAGE(X)`	συνάρτηση με μια είσοδο
`IMPLICIT NONE`	
`DOUBLE PRECISION X, A`	δήλωση τυπικής (X) και τοπικής (A) μεταβλητής
`INTEGER M`	δήλωση τοπικής μεταβλητής
`DATA M, A / 0, 0 /`	οι τοπικές μεταβλητές παίρνουν αρχική τιμή κατά την πρώτη κλήση
`SAVE M, A`	η τιμή των τοπικών μεταβλητών να σώζεται στην κάθε κλήση
`A = A + X`	η τοπική μεταβλητή A σώζει το άθροισμα των στοιχείων
`M = M + 1`	η τοπική μεταβλητή N σώσει τον αριθμό των στοιχείων
`AVERAGE = A / M`	ο προσωρινός μέσος όρος
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

7.2 Παραδείγματα

7.2.1 Τιμή συνάρτησης και παραγώγου της

μια συνάρτηση για τον υπολογισμό της τιμής της

$$f(x) = \frac{2x^2 + 3x + 4}{5x^2 + 6x + 7}$$

και μια για την παράγωγό της.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION FUNCTION FUNCT1( X )`	συνάρτηση με μια είσοδο
`IMPLICIT NONE`	
`DOUBLE PRECISION X`	δήλωση τυπικής μεταβλητής
`FUNCT1 = (2*X**2+3*X+4)/(5*X**2+6*X+7)`	υπολογισμός
`RETURN`	
`END`	

Ο αριθμητικός υπολογισμός της παραγώγου μιας συνάρτησης γίνεται βάσει του ορισμού της παραγώγου

$$\lim_{e \to 0} \frac{f(x+e) - f(x-e)}{2e}$$

με την κεντρική διαφορά 2e να είναι ένα πολύ μικρό ποσοστό (ένα εκατομμυριοστό στο παράδειγμα) του X.

Κώδικας Fortran	*Σχολιασμός*
`DOUBLE PRECISION FUNCTION DERIV1( X )`	συνάρτηση με μια είσοδο
`IMPLICIT NONE`	
`DOUBLE PRECISION X, E, FUNCT1`	δήλωση τυπικής, τοπικής μεταβλητής και συνάρτησης
`E = 1.0D-6 * X`	τοπική μεταβλητή για την διαφορά
`DERIV1 = (FUNCT1(X+E)-FUNCT1(X-E))/(2*E)`	υπολογισμός καλώντας τη συνάρτηση
`RETURN`	
`END`	

Κεφάλαιο 8

Υποπρογράμματα: Υπορουτίνες

Οι συναρτήσεις είναι ιδανικές για ομαδοποίηση μικρών κομματιών κώδικα και όταν το αποτέλεσμα είναι ένα νούμερο. Στις περιπτώσεις που χρειαζόμαστε περισσότερα αποτελέσματα να γυρίσουν πίσω, ή και γενικότερα όταν οι πράξεις που θέλουμε να ομαδοποιήσουμε είναι πολλές, χρησιμοποιούμε μια πιο γενικευμένη μορφή υποπρογράμματος, την υπορουτίνα. Η βασική διαφορά είναι ότι η υπορουτίνα δεν γυρνάει το αποτέλεσμα μέσω του ονόματός της κατά την κλήση, αλλά μέσω της λίστας των μεταβλητών, στην οποία δηλαδή εισέρχονται και οι μεταβλητές εισόδου αλλά και οι μεταβλητές εξόδου. Κατά συνέπεια, μπορούμε να έχουμε πάνω από μια εξόδους. Βέβαια, ότι μπορεί να γίνει με μια υπορουτίνα μπορεί να γίνει και με μια συνάρτηση (πχ αλλάζοντας τις τιμές των μεταβλητών εισόδου μέσα στη συνάρτηση).

Στην πράξη αυτά τα δύο είδη υποπρογραμμάτων δεν έχουν και πολλές διαφορές, απλά βοηθούν μέσω των μικρών διαφορών τους να κάνουν ένα πρόγραμμα πιο ευανάγνωστο και πιο σπονδυλωτό. Στο τέλος, εξαρτάται στον προγραμματιστή να διαλέξει το προσωπικό του στυλ. Ένας γενικός κανόνας θα μπορούσε να είναι ότι για μια πράξη με ένα αποτέλεσμα (και χωρίς μεταβολή των μεταβλητών εισόδου) να χρησιμοποιούμε συναρτήσεις, ενώ για πολλές πράξεις με πολλά αποτελέσματα (και πιθανές μεταβολές των μεταβλητών εισόδου) να χρησιμοποιούμε υπορουτίνες.

8.1 Δήλωση

Η δήλωση μιας υπορουτίνας γίνεται μετά το τέλος του προγράμματος (μετά την εντολή END) με την χρήση της εντολής SUBROUTINE. Καθώς με το όνομα της υπορουτίνας δεν γυρνάει κάποια μεταβλητή στο κυρίως πρόγραμμα, η υπορουτίνα δεν έχει τύπο. Αποφεύγουμε να χρησιμοποιούμε κοινά ονόματα με συναρτήσεις βιβλιοθήκης ή άλλα υποπρογράμματα, ή άλλες απλές μεταβλητές. Το όνομα της κάθε υπορουτίνας πρέπει να είναι μοναδικό.

```
SUBROUTINE  «ΟΝΟΜΑ»  ( «ΤΥΠΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ» )
 IMPLICIT NONE
 ΔΗΛΩΣΕΙΣ ΜΕΤΑΒΛΗΤΩΝ     τυπικές μεταβλητές εισόδου και εξόδου, καθώς και άλλες τοπικές
    ⋮
 ΚΩΔΙΚΑΣ                  όλες οι εντολές που θέλουμε να εκτελεστούν
    ⋮
 RETURN                   επιστροφή στο κυρίως πρόγραμμα
 END
```

Μέσα στις παρενθέσεις είναι η λίστα των μεταβλητών εισόδου και εξόδου (τυπικές παράμετροι). Εδώ μπαίνουν οι μεταβλητές (χωρισμένες από κόμματα) που εισέρχονται από το πρόγραμμα. Δεν υπάρχει περιορισμός πόσα και τι. Μπορούμε να έχουμε σύνολο μέχρι και 256 μεταβλητές. Οι μεταβλητές αυτές ονομάζονται «τυπικές» και ήδη υπάρχουν στο κυρίως πρόγραμμα. Όπως και στην συνάρτηση, μέσα στην υπορουτίνα πρέπει να ξαναδηλώσουμε τι είδους μεταβλητή είναι η κάθε μία. Όπως και στο κυρίως, ξεκινάμε με το IMPLICIT NONE, και μετά δηλώνουμε. Μια υπορουτίνα είναι σαν ένα νέο μικρό πρόγραμμα.

Όλα τα υπόλοιπα είναι τα ίδια με την συνάρτηση: ονοματολογία και δήλωση τυπικών μεταβλητών, ορισμός τοπικών μεταβλητών, χρησιμοποίηση της εντολής επιστροφής RETURN. Το σώμα τον εντολών της υπορουτίνας τελειώνει όπως και της συνάρτησης και του κυρίως προγράμματος με το END.

Για παράδειγμα, έστω ότι επιθυμούμε μια υπορουτίνα για τον υπολογισμό του μέτρου ενός διδιάστατου διανύσματος καθώς και της γωνίας του με τον άξονα των X, δηλαδή την μετατροπή από καρτεσιανές σε πολικές συντεταγμένες. Στην είσοδο θα πρέπει να βάλουμε τις X και Y συνιστώσες του διανύσματος, και στην έξοδο να πάρουμε το μέτρο και γωνία. Και οι τέσσερις μεταβλητές θα πρέπει να είναι διπλής ακρίβειας. Θα ονομάσουμε την υπορουτίνα μας CARTESIAN2POLAR:

Κώδικας Fortran	*Σχολιασμός*
`SUBROUTINE CARTESIAN2POLAR (X,Y,R,THETA)`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, Y, R, THETA`	δήλωση τυπικών μεταβλητών
`R = SQRT(X**2 + Y**2)`	εντολές υπορουτίνας
`THETA = ATAN (Y / X)`	
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

8.2 Κλήση

Μια υπορουτίνα καλείται με την εντολή CALL, και μπορεί να καλεστεί στο κυρίως πρόγραμμα αλλά και μέσα σε κάποια άλλη υπορουτίνα ή συνάρτηση (ή αντίστοιχα φυσικά, μια συνάρτηση να καλεστεί μέσα σε μια υπορουτίνα). Καθώς το όνομα της υπορουτίνας δεν χρησιμοποιείται ως μεταβλητή, δεν δηλώνεται στο κυρίως πρόγραμμα.

Ως παράδειγμα, ας δούμε το πλήρες πρόγραμμα που θα χρησιμοποιούσε την παραπάνω υπορουτίνα:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM VECTOR2D`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, Y, R, THETA`	δήλωση μεταβλητών
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ  X,Y ΣΥΝΙΣΤΩΣΕΣ;'`	προτροπή
`READ (*,*) X, Y`	διάβασμα συνιστωσών
`CALL CARTESIAN2POLAR(X, Y, R, THETA)`	καλούμε την υπορουτίνα
`WRITE( *,*)'ΤΟ ΜΕΤΡΟ ΚΑΙ Η ΓΩΝΙΑ ΤΟΥ ΔΙΑΝΥΣΜΑΤΟΣ ΕΙΝΑΙ',R,THETA`	
`END`	
`SUBROUTINE CARTESIAN2POLAR(X, Y, R, THETA)`	
`IMPLICIT NONE`	
`DOUBLE PRECISION X, Y, R, THETA`	δήλωση τυπικών μεταβλητών
`R = SQRT(X**2 + Y**2)`	εντολές υπορουτίνας
`THETA = ATAN (Y / X)`	
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

Προσοχή: αντίθετα με την περίπτωση της συνάρτησης (που το όνομά της είναι μεταβλητή), την υπορουτίνα δεν μπορούμε να την καλέσουμε κατά την εκτύπωση. Το παρακάτω θα προκαλούσε λάθος κατά την μετάφραση:

`WRITE (*,*) 'ΤΟ ΜΕΤΡΟ ΚΑΙ Η ΓΩΝΙΑ',CARTESIAN2POLAR(X,Y,R,THETA)` **ΛΑΘΟΣ!!!**

Όπως κι στην συνάρτηση, στην είσοδο μιας υπορουτίνας μπορούμε να έχουμε και πίνακες. Στη λίστα εισόδου/εξόδου γράφουμε μόνο τα ονόματα των πινάκων, και μέσα στην υπορουτίνα δηλώνουμε το μέγεθος και τις διαστάσεις τους. Το παραπάνω παράδειγμα με χρήση πινάκων:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM VECTOR2D`	
`IMPLICIT NONE`	
`DOUBLE PRECISION CART(2), POL(2)`	δήλωση μεταβλητών
`WRITE(*,*) 'ΕΙΣΑΓΕΤΕ ΤΙΣ X,Y ΣΥΝΙΣΤΩΣΕΣ;'`	προτροπή
`READ (*,*) CART(1), CART(2)`	διάβασμα συνιστωσών
`CALL CARTESIAN2POLAR(CART, POL)`	καλούμε την υπορουτίνα
`WRITE (*,*) 'ΤΟ ΜΕΤΡΟ ΕΙΝΑΙ', POL(1)`	
`WRITE (*,*)  Ή ΓΩΝΙΑ ΕΙΝΑΙ', POL(2)`	
`END`	
`SUBROUTINE CARTESIAN2POLAR(CART, POL)`	
`IMPLICIT NONE`	
`DOUBLE PRECISION CART(2), POL(2)`	δήλωση τυπικών μεταβλητών
`POL(1) = SQRT(CART(1)**2 + CART(2)**2)`	εντολές υπορουτίνας

```
POL(2) = ATAN (CART(2) / CART(1))
RETURN                                          επιστροφή στο κυρίως πρόγραμμα
END
```

Όπως έχουμε επισημάνει και πριν, τα ονόματα των τυπικών μεταβλητών μπορούν να αλλάζουν μέσα στην υπορουτίνα, το κυρίως πρόγραμμα γνωρίζει μόνο τι εισάγεται/εξάγεται μέσω της λίστας μεταβλητών και με ποια σειρά. Τα ονόματα μέσα στην υπορουτίνα δεν έχουν σημασία. Η παραπάνω υπορουτίνα θα μπορούσε εντελώς ισοδύναμα να γραφτεί ως:

Κώδικας Fortran	*Σχολιασμός*
`SUBROUTINE CARTESIAN2POLAR(C, P)`	
`IMPLICIT NONE`	
`DOUBLE PRECISION C(2), P(2)`	δήλωση τυπικών μεταβλητών
`P(1) = SQRT(C(1)**2 + C(2)**2)`	εντολές υπορουτίνας
`P(2) = ATAN (C(2) / C(1))`	
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

Έτσι, μια υπορουτίνα (ή και συνάρτηση) να μπορεί να χρησιμοποιηθεί από πολλά προγράμματα, ανεξάρτητα από την ακριβή ονοματολογία των μεταβλητών του κάθε προγράμματος, δηλαδή δεν χρειάζεται να αλλάζουμε όλα τα ονόματα των μεταβλητών της υπορουτίνας για κάθε νέο πρόγραμμα στο οποίο θέλουμε να την καλέσουμε.

Επίσης, όπως και στις συναρτήσεις, για πίνακες μεγάλου μεγέθους μπορούμε να έχουμε το μέγεθος να εισέρχεται σαν ακέραια μεταβλητή από την λίστα εισόδου.

8.3 Δηλώσεις μεταβλητών με COMMON BLOCKS

Οι εντολές DATA και SAVE χρησιμοποιούνται εδώ όπως και στις συναρτήσεις. Θα εξετάσουμε εδώ την εντολή COMMON. Αυτή συνίσταται μεν όταν θέλουμε την διατήρηση της τιμής πολλών μεταβλητών, αλλά κυρίως όταν χρειάζεται η εισαγωγή και εξαγωγή πολλών μεταβλητών κατά την κλήση μιας υπορουτίνας. Ένα COMMON BLOCK έχει όνομα και την λίστα των μεταβλητών που εμπεριέχει (ο τύπος των μεταβλητών πρέπει υποχρεωτικά να έχει προηγηθεί της δήλωσης του COMMON), χωρισμένα με καθέτους (οι μεταβλητές μεταξύ τους χωρίζονται με κόμματα). Στην περίπτωση που είναι πίνακες, γράφουμε μόνο το όνομά τους εάν ήδη έχει προηγηθεί δήλωση των διαστάσεών τους. Εάν όχι, μπορούμε να συμπεριλάβουμε την δήλωση των διαστάσεων μέσα στον ορισμό του COMMON. Για παράδειγμα:

```
DOUBLE PRECISION A(3, 3), B(10), Z
INTEGER L(10), N
COMMON / MYDATA / A, B, Z, L, N
```

είναι τελείως ισοδύναμο με

```
DOUBLE PRECISION A, B, Z
INTEGER L, N
COMMON / MYDATA / A(3, 3), B(10), Z, L(10), N
```

Το MYDATA είναι το όνομα του παραπάνω COMMON BLOCK. Μπορούμε να ορίσουμε όσα COMMON BLOCKS θέλουμε, αρκεί να έχουν διαφορετικά ονόματα, το οποία δεν πρέπει να συμπίπτουν με ονόματα άλλων μεταβλητών, συναρτήσεων η υπορουτίνων. Τα COMMON BLOCK δηλώνεται σε κάθε κομμάτι κώδικα (κυρίως πρόγραμμα και υπορουτίνες) στο οποίο θα χρειαστούμε πρόσβαση στις μεταβλητές αυτές. Για παράδειγμα ας ξαναγράψουμε το παράδειγμα της συνάρτησης AVERAGE για τον υπολογισμό του μέσου όρου, χρησιμοποιώντας υπορουτίνες και COMMON BLOCK:

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM AVERAGE_NUMBERS`	
`IMPLICIT NONE`	
`INTEGER I, N, NUM`	δήλωση μεταβλητών
`DOUBLE PRECISION X, SUM, AV`	δήλωση μεταβλητών και συνάρτησης
`COMMON / AVER / NUM, SUM, AV`	δημιουργία COMMON BLOCK ονόματι AVER
`WRITE (*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`SUM = 0`	ο αθροιστής στο μηδέν
`NUM = 0`	ο αριθμός στοιχείων στο μηδέν
`DO I = 1, N`	
`  WRITE(*,*)'ΔΩΣΤΕ ΤΟ ΕΠΟΜΕΝΟ ΝΟΥΜΕΡΟ;'`	προτροπή
`  READ (*,*) X`	εισαγωγή του επόμενου στοιχείου
`  CALL AVERAGE( X )`	κάλεσμα υπορουτίνας
`  WRITE(*,*)'Ο ΝΕΟΣ ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ:',AV`	εκτύπωση
`END DO`	
`END`	
`SUBROUTINE AVERAGE( X )`	υπορουτίνα με μια είσοδο
`IMPLICIT NONE`	
`DOUBLE PRECISION X, SUM, AV`	δήλωση τυπικής (X) και τοπικής (A) μεταβλητής
`INTEGER NUM`	δήλωση τοπικής μεταβλητής
`COMMON / AVE / NUM, SUM, AV`	
`SUM = SUM + X`	η τοπική μεταβλητή A σώζει το άθροισμα των στοιχείων
`NUM = NUM + 1`	η τοπική μεταβλητή N σώσει τον αριθμό των στοιχείων
`AV = SUM / NUM`	
`RETURN`	επιστροφή στο κυρίως πρόγραμμα
`END`	

8.4 Παραδείγματα

8.4.1 Ταξινόμηση

Αναδιατάξτε έναν μονοδιάστατο πίνακα σε αύξουσα σειρά. Είναι ακριβώς το ίδιο πρόγραμμα με το παράδειγμα ταξινόμησης 5.4.3, με τη διαφορά ότι εδώ θα ορίσουμε και θα χρησιμοποιήσουμε δύο υπορουτίνες. Παρατηρήστε πως αλλάζουν τα διάφορα μέρη του συνολικού προγράμματος, ενώ στο τέλος οι εντολές που εκτελούνται παραμένουν οι ίδιες.

Κώδικας Fortran	*Σχολιασμός*
`PROGRAM SORT`	
`IMPLICIT NONE`	
`INTEGER N, NMAX`	
`PARAMETER ( NMAX=1000 )`	
`DOUBLE PRECISION A(NMAX)`	πίνακας A με 1000 στοιχεία διπλής ακρίβειας
`WRITE (*,*) 'ΠΟΣΑ ΝΟΥΜΕΡΑ ΘΑ ΕΙΣΑΓΕΤΕ;'`	προτροπή
`READ (*,*) N`	ο αριθμός των στοιχείων
`IF (N.GT.NMAX) THEN`	έλεγχος να μην ξεπερνούν το μέγιστο NMAX
`  WRITE (*,*) 'ΛΑΘΟΣ - ΠΙΟ ΛΙΓΑ ΝΟΥΜΕΡΑ'`	
`  STOP`	εάν το ξεπερνούν τερματίζουμε
`ENDIF`	
`WRITE (*,*) 'ΕΙΣΑΓΕΤΕ ΤΑ ΝΟΥΜΕΡΑ'`	προτροπή
`READ (*,*) (A(I), I = 1, N)`	εισαγωγή στοιχείων (σε μια γραμμή)
`DO J = 1, N`	διατρέχουμε τις θέσεις, στην πρώτη θα βάλουμε το μικρότερο,κοκ
`  CALL ARRAYMIN(A, NMAX, J, N, K)`	κάλεσμα υπορουτίνας ελαχίστου
`  IF (K.NE.J) CALL EXCHANGE(A(J),A(K))`	ανταλλαγή δύο στοιχείων πίνακα
`END DO`	
`WRITE (*,*) 'ΤΑΞΙΝΟΜΗΜΕΝΑ ΝΟΥΜΕΡΑ:'`	
`WRITE (*,*) (A(I), I = 1, N)`	εξαγωγή στοιχείων (σε μια γραμμή)
`END`	
`SUBROUTINE ARRAYMIN(A, NMAX, J, N, K)`	Η υπορουτίνα ARRAYMIN βρίσκει το ελάχιστο μεταξύ J και N
`IMPLICIT NONE`	
`INTEGER NMAX, I, J, N, K`	
`DOUBLE PRECISION A(NMAX), AMIN`	
`AMIN = A(J)`	για την θέση J θέτουμε αρχικά ως μικρότερο το A(J)
`K = J`	το K θα κρατάει την θέση του μικρότερου (από J και πάνω)
`DO I = J+1, N`	διατρέχουμε όλα τα στοιχεία πάνω από το J

```
   IF(A(I) .LT. AMIN) THEN
      AMIN = A(I)                    εάν κάποιο είναι μικρότερο, το κατοχυρώ-
                                     νουμε
      K = I                          κατοχυρώνουμε την θέση του μικρότερου
   ENDIF
END DO
RETURN
END
```

```
SUBROUTINE EXCHANGE(X, Y)            Η υπορουτίνα EXCHANGE, ανταλλάσσει
                                     τις τιμές των X και Y
IMPLICIT NONE
DOUBLE PRECISION X, Y, TEMP
TEMP = X
X = Y                                ανταλλάσσουμε τις τιμές των X και Y
Y = TEMP
RETURN
END
```

Προσέξτε στο παραπάνω ότι η είσοδος της υπορουτίνας EXCHANGE είναι δύο ρητοί διπλής ακρίβειας. Στο κάλεσμά της από το κυρίως πρόγραμμα, εισάγουμε δύο συγκεκριμένα στοιχεία του πίνακα, και όχι τους ίδιους τους πίνακες. Αυτή η υπορουτίνα δεν έχει κάποια διαφορετική έξοδο, η είσοδος είναι και η έξοδος. Επίσης προσέξτε ότι η έξοδος της υπορουτίνας ARRAYMIN είναι ο δείκτης K στον οποίο βρίσκεται η μικρότερη τιμή μεταξύ των δεικτών J και N. Μπορεί για ένα τόσο απλό παράδειγμα η χρησιμοποίηση υπορουτίνων να φαίνεται υπερβολική, αλλά όσο αυξάνει η πολυπλοκότητα του προγράμματος, πολύ γρήγορα γίνεται απαραίτητη.