ΘΩΜΑΣ Β. ΜΠΑΚΑΣ
ΕΠΙΚ. ΚΑΘΗΓΗΤΗΣ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ

ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ FORTRAN

ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΠΑΡΑΔΟΣΕΙΣ

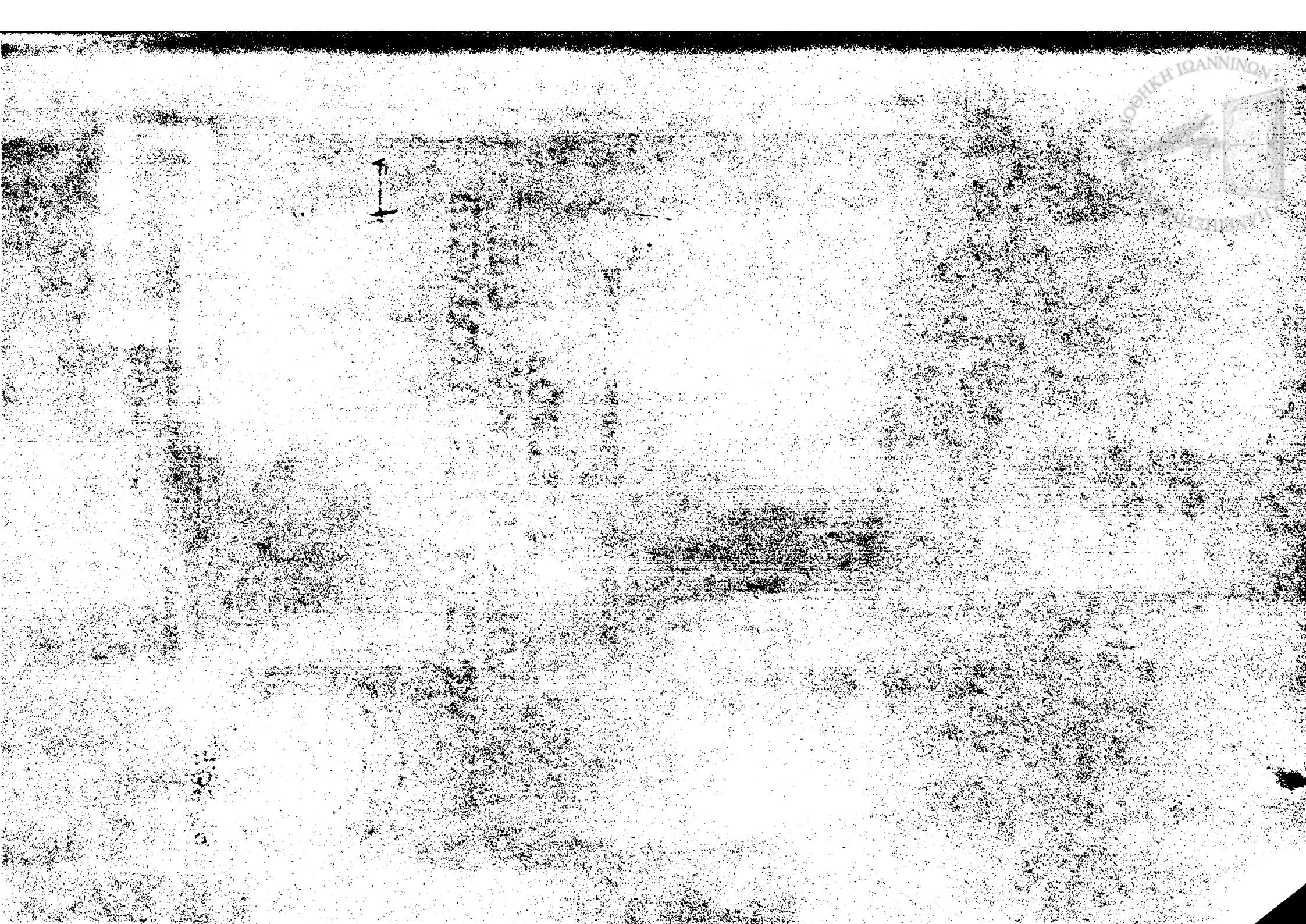
ΙΩΑΝΝΙΝΑ 1994

ΘΩΜΑΣ Β. ΜΠΑΚΑΣ
ΕΠΙΚ. ΚΑΘΗΓΗΤΗΣ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ

ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ FORTRAN

ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΠΑΡΑΔΟΣΕΙΣ

ΙΩΑΝΝΙΝΑ 1994



ΠΡΟΛΟΓΟΣ

Οι ηλεκτρονικοί υπολογιστές αποτελούν σήμερα ένα συνηθισμένο εργαλείο στην καθημερινή δραστηριότητα της κοινωνίας. Στις περισσότερες περιπτώσεις βέβαια, η χρήστης δεν ξέρει- και στις περισσότερες περιπτώσεις δεν θέλει να ξέρει- πως λειτουργεί ο υπολογιστής και πως είναι κατασκευασμένο το πρόγραμμα το οποίο χρησιμοποιεί, είτε αυτό είναι παιχνίδι είτε κάποια εμπορική εφαρμογή.

Το βιβλίο αυτό δίνει κάποια στοιχειώδη περιγραφή της δομής ενός ηλεκτρονικού υπολογιστή και σαν κύριο σκοπό έχει να δώσει τα απαραίτητα στοιχεία για την διδασκαλία της γλώσσας προγραμματισμού FORTRAN. Ο αναγνώστης θα πρέπει να κατανοήσει την ιδιαιτερότητα που υπάρχει στην εκμάθηση γλωσσών προγραμματισμού. Με κανένα τρόπο δεν πρέπει να θεωρηθεί οτι αρκεί η μελέτη ενός ή περισσοτέρων βιβλίων σχετικών με το θέμα, αλλα πρωταρχική σημασία παίζει η εξάσκηση. Συνιστάται λοιπόν, ιδιαίτερα στους μαθητευόμενους στον προγραμματισμό, η συγγραφή όσο το δυνατόν περισσοτέρων, απλών στην αρχή, προγραμμάτων.

Το βιβλίο αυτό δεν πρέπει να θεωρηθεί ως βιβλίο αναφοράς για όλες τις εντολές της γλώσσας FORTRAN 77. Σκοπός του είναι να δώσει τις περισσότερο χρησιμοποιούμενες εκφράσεις και εντολές, οι οποίες καλύπτουν σχεδόν το σύνολο των αναγκών ενός προγραμματιστή. Παρ'όλο που οι εντολές είναι αυτές της έκδοσης FORTRAN 77, καλό θα είναι, σε περιπτώσεις αμφιβολίας, η ενημέρωση απο τα σχετικά εγχειρίδια της έκδοσης η οποία πρόκειται να χρησιμοποιηθεί.

Η παρούσα έκδοση στηρίχθηκε, εν μέρει, στο βιβλίο "Προγραμματισμός σε FORTRAN" των συναδέλφων κ.κ. Ι.Η. Λαγαρή και Γ. Παντή απο όπου έχουν ληφθεί πολλές απο τις ασκήσεις στο τέλος του βιβλίου. Το μεγαλύτερο μέρος όμως έχει ξαναγραφεί και εμπλουτιστεί με παραδείγματα, για την καλλίτερη κατανόηση των εντολών. Η επισήμανση ασαφειών ή/και παραλείψεων απο τους αναγνώστες θα αποτελέσει σημαντική βοήθεια για την βελτίωση του βιβλίου αυτού σε μελλοντική έκδοση.

Απο την θέση αυτή θεωρώ υποχρέωση να ευχαριστήσω τον αναπλ. καθηγητή κ. Β. Παπαευθυμίου και την επικ. καθηγήτρια κ. Α. Μουκαρίκα του Τμήματος Φυσικής καθώς επίσης και τον αναπλ. καθηγητή του Τμήματος Πληροφορικής κ. Ι. Η. Λαγαρή, για τις συζητήσεις που είχαμε πρίν και κατα την διάρκεια της συγγραφής του παρόντος και την κ. Φ. Φουντουλάκη-Βέργου για την επιμελημένη δακτυλογράφηση του κειμένου.

Ιωάννινα 1994

Θ. Μπάκας

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Από τη στιγμή που ο άνθρωπος ένοιωσε την ανάγκη να μετρήσει κάποια μεγέθη και να κάνει κάποιους υπολογισμούς, χρειάστηκε να χρησιμοποιήσει κάποια βοηθητικά εργαλεία. Αρχικά τα εργαλεία αυτά ήταν μέρη του σώματός του (δάκτυλα, πόδια), ενώ αργότερα προχώρησε στην κατασκευή εργαλείων που τον βοηθούσαν στις μετρήσεις και τους υπολογισμούς. Ο άβακας μπορεί να θεωρηθεί ως ένας πρώτος υπολογιστής, ενώ πολυ αργότερα κατασκευάστηκε ο λογαριθμικός κανόνας από τον Napier. Μετά το 1640 κατασκευάστηκαν μηχανικά συστήματα που ήσαν ικανά να εκτελούν αριθμητικές πράξεις, αρχικά πρόσθεση και αφαίρεση και αργότερα (περί το 1810) πολλαπλασιαμό και διαίρεση.

Ο πρώτος Η/Υ κατασκευάστηκε το 1946 στο Παν/μιο της Πενσυλβάνιας και λειτουργούσε με ηλεκτρικό ρεύμα, έχοντας ως βάση το δεκαδικό σύστημα. Το 1949 κατασκευάστηκε ο πρώτος Η/Υ, στο Παν/μιο Cambridge, που χρησιμοποιούσε ως βάση αρίθμησης το δυαδικό σύστημα. Οι πρώτοι υπολογιστές που κατασκευάστηκαν στη δεκαετία του 50 είχαν περιορισμένες υπολογιστικές δυνατότητες και επειδή χρησιμοποιούσαν λυχνίες κατανάλωναν τεράστιες ποσότητες ηλεκτρικής ισχύος.

Με την εξέλιξη της τεχνολογίας, και ουσιαστικά με την κατασκευή των τρανζίστορς και των ολοκληρωμένων κυκλωμάτων, η εξέλιξη του Η/Υ υπήρξε ραγδαία και μειώθηκε εντυπωσιακά το κόστος αγοράς και λειτουργίας των Η/Υ. Σημαντικό ρόλο στην εξάπλωση της χρήσης των Η/Υ έπαιξε η τεχνολογία της καταγραφής και ανάγνωσης στοιχείων. Σήμερα το πλέον διαδεδομένο μέσο καταγραφής και ανάγνωσης στοιχείων Η/Υ είναι οι μαγνητικοί δίσκοι και ταινίες, ενώ εξελίσσεται η τεχνολογία των οπτικών μέσων αποθήκευσης στοιχείων.

Οι σύγχρονοι υπολογιστές χρησιμοποιούν ως βάση αρίθμησης το δυαδικό σύστημα, που σημαίνει ότι "καταλαβαίνουν" μόνο τα ψηφία 0 και 1. Κάθε αριθμός του δεκαδικού συστήματος , όπως επίσης και κάθε γράμμα ή σύμβολο, μπορεί να παρασταθεί στο δυαδικό σύστημα και αντίστροφα. Στον παρακάτω πίνακα φαίνεται μια τέτοια αντιστοιχεία δεκαδικών αριθμών και η έκφρασή τους στο δυαδικό σύστημα.

Δεκαδική μορφή	Δυαδική μορφή	Δεκαδική μορφή	Δυαδική μορφή
0	00110000	5	00110101
1	00110001	6	00110110
2	00110010	7	00110111
3	00110011	8	00111000
4	00110100	9	00111001

Οι πράξεις μεταξύ δυαδικών αριθμών ακολουθούν τους βασικούς κανόνες των πράξεων στο δεκαδικό σύστημα, με τη διαφορά ότι γίνονται στο δυαδικό σύστημα. Έτσι, στο δυαδικό σύστημα, η πράξη ένα και ένα δίνει αποτέλεσμα δέκα.
π.χ

1 0	0	1	101	11101	11111	
+ 0	+ 1	+ 0	+ 1	+ 1	+ 1001	00001
1 1	0	10	110	100110	100000	

και δέκα πλην ένα ίσον ένα

0 1	1	10	1011	101101	
- 0	- 0	-1	-1	- 0101	- 010001
0 1	0	1	0110	011100	

Με ανάλογο τρόπο γίνονται οι πράξεις του πολλαπλασιασμού και διαιρέσεως δυαδικό αριθμών. Παρατηρούμε λοιπόν ότι το αποτέλεσμα των πράξεων μεταξύ δυαδικών αριθμών είναι ένα δυαδικός αριθμός.

Ένα ερώτημα που δημιουργείται σε πολλούς μή ειδικούς είναι γιατί οι ηλεκτρονικοί υπολογιστές να λειτουργούν στο δυαδικό σύστημα και όχι στο δεκαδικό, το οποίο ο άνθρωπος έχει μάθει και καταννοεί. Όπως φαίνεται από τα προηγούμενα, οι πράξεις μεταξύ δυαδικών αριθμών δίνουν ως αποτέλεσμα ένα δυαδικό αριθμό. Η κατασκευή του ηλεκτρονικού υπολογιστή, που λειτουργεί στο δυαδικό σύστημα, είναι φθηνή και αξιόπιστη καθόσον τα ηλεκτρονικά κυκλώματα οφείλουν να αισθάνονται μόνον δύο καταστάσεις. Την κατάσταση "0" (μηδέν) και την κατάσταση "1" (ένα) που ισοδυναμούν με τη μή διέλευση ή διέλευση ρεύματος από το κύκλωμα. Η κατάσταση μηδέν συνήθως είναι η εφαρμογή τάσης μεταξύ 0 και +0.4 V ενώ η κατάσταση ένα είναι η εφαρμογή τάσης μεταξύ +4 και +5V. Έτσι λοιπόν τα ηλεκτρονικά κυκλώματα του υπολογιστή καλούνται να διαχωρίσουν (αναγνωρίσουν) δύο τελείως διακριτές καταστάσεις 0-0.4V και 4-5V, η δε διαδικασία αυτή είναι πολύ εύκολη και έχει μεγάλη αξιοπιστία.

Αν ο υπολογιστής ήταν κατασκευασμένος να λειτουργεί στο δεκαδικό σύστημα, τότε τα ηλεκτρονικά του κυκλώματα θα έπρεπε να είναι ικανά να αναγνωρίζουν δέκα διαφορετικές καταστάσεις (τιμές τάσεως) μεταξύ 0V και 9V (Δηλ. 0V="0", 1V="1", 2V="2" κλπ.) Με δοδομένο ότι τα διάφορα στοιχεία του κυκλώματος (πυκνωτές, αντιστάσεις, επαγωγές κλπ) δημιουργούν κάποια πτώση τάσεως, μια δεδομένη στάθμη π.χ. 5V θα ήταν δυαντόν στο σημείο αναγνώρισης να καταλήξει

ως 4.7V ή και λιγώτερο. Η πιθανότητα λοιπόν εσφαλμένης αναγνώρισης είναι αυξημένη και κατά συνέπεια το λανθασμένο αποτέλεσμα είναι πιο πιθανό.

Ένας σύγχρονος ηλεκτρονικός υπολογιστής, μπορεί να θεωρηθεί οτι αποτελείται απο τέσσερα κύρια τμήματα, καθένα απο τα οποία εκτελεί μια ορισμένη εργασία. Τα τμήματα αυτά είναι: Η μνήμη, η αριθμητική μονάδα, η μονάδα ελέγχου και η μονάδα εισόδου- εξόδου. Μπορεί δε να θεωρηθεί οτι ο διαχωρισμός αυτός είναι ανεξάρτητος απο το μέγεθος και τις δυνατότητες του υπολογιστή. Σε μερικές περιπτώσεις η αριθμητική μονάδα και η μονάδα ελέγχου αποτελούν μια ενότητα. Τα τέσσερα αυτά τμήματα μαζί αποτελούν αυτό που ονομάζεται Central Processing Unit (CPU) και το τμήμα εισόδου-εξόδου είναι το κανάλι επικοινωνίας του υπολογιστή με τον χρήστη.

Ανάλογα με την υπολογιστική ισχύ (δηλ. πόσες πράξεις ανα δευτερόλεπτο μπορούν να εκτελεστούν), οι υπολογιστές χωρίζονται σε κατηγορίες όπως: micro, mini, maxi, work station, main frame, super computer.

Ο κάθε υπολογιστής μπορεί να θεωρηθεί χωρίς υπερβολή, οτι είναι ένα "χαζοκούτι". Ο χαρακτηρισμός αυτός αποδίδεται λόγω του γεγονότος οτι "δεν ξέρει τι να κάνει" και δεν αναγνωρίζει τίποτε απο το περιβάλλον του. Απο τα ηλεκτρονικά κυκλώματα οδηγείται και "διαβάζει" οδηγίες απο κάποια μνήμη, της οποίας το περιεχόμενο δεν σβήνει ακόμη και όταν ο υπολογιστής δεν τροφοδοτείται απο ηλεκτρικό ρεύμα. Οι οδηγίες βρίσκονται σε έναν τύπο μνήμης που ονομάζεται Random Access Memory (RAM), πληροφορούν τον υπολογιστή σχετικά με το περιβάλλον του και τον καθοδηγούν ώστε να διαβάσει άλλες πληροφορίες που βρίσκονται καταχωρημένες σε κάποια περιφερειακή μνήμη. Απο το σημείο αυτό και μετά ο υπολογιστής είναι σε θέση να δεχθεί και να εκτελέσει εντολές του χρήστη. Με την έννοια περιφερειακή μνήμη νοείται κάθε μέσο αποθήκευσης πληροφοριών, όπως σκληρός δίσκος, μαγνητική ταινία κλπ..

Στην περιφερειακή μνήμη μπορούν να βρίσκονται αποθηκευμένα προγράμματα και διάφορα άλλα στοιχεία. Ο υπολογιστής όμως δεν ξέρει τι να κάνει σαν επόμενο βήμα και πρέπει να τον καθοδηγήσει ο χρήστης, δίνοντας τις κατάλληλες εντολές. Για παράδειγμα, για να εκτελεστεί κάποιο πρόγραμμα, πρέπει πρώτα να μεταφερθεί απο την περιφερειακή μνήμη όπου βρίσκεται αποθηκευμένο, στην κεντρική μνήμη του υπολογιστή. Την απολουθία των εντολών που είναι απαραίτητες για να γίνει η διαδικασία αυτή, πρέπει να την δώσει ο χρήστης.

Η εισαγωγή εντολών και δεδομένων γίνεται απο το πληκτρολόγιο και η έξοδος των αποτελεσμάτων απο τον υπολογιστή γίνεται συνήθως στην οθόνη. Είναι δυνατόν η είσοδος και η έξοδος να γίνεται και σε κάποιες άλλες μονάδες, όπως σκληρός δίσκος, μαγνητική ταινία, εκτυπωτής κλπ., ανάλογα με τις οδηγίες του χρήστη. Η διαδικασία αυτή ακολουθείται στις περιπτώσεις όπου το πλήθος των

εξερχόμενων στοιχείων είναι μεγάλο και χρειάζεται χρόνος για την μελέτη των αποτελεσμάτων.

Το κυριότερο πρόγραμμα που βρίσκεται καταχωρημένο σε κάποια περιφερειακή μνήμη είναι το λειτουργικό σύστημα του υπολογιστή. Πρόκειται για ένα πρόγραμμα το οποίο ελέγχει και ρυθμίζει όλη την λειτουργία του υπολογιστή. Εκτός απο το λειτουργικό σύστημα, στις περιφερειακές μνήμες μπορούν να περιέχονται και διάφορα άλλα προγράμματα τα οποία επιτελούν διαφορετική εργασία το καθένα. Τέτοια προγράμματα είναι προγράμματα επεξεργασίας κειμένου, σχεδιασμού, παιχνίδια κλπ..Οι γλώσσες προγραμματισμού μπορούν να θεωρηθούν ως διαφορετικού είδους προγράμματα, τα οποία επιτρέπουν την δημιουργία άλλων προγραμμάτων.

1. ΓΕΝΙΚΑ ΓΙΑ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Μια γλώσσα προγραμματισμού συνίσταται απο ένα περιορισμένο υποσύνολο της αγγλικής γλώσσας και το οποίο αποτελείται απο εκφράσεις σαφώς ορισμένες, χωρίς παρερμηνεία. Οι υπολογιστές μπορούν να εκτελούν ένα περιορισμένο σύνολο εντολών που εκφράζονται σε δυαδικό κώδικα, δηλ. ακολουθία μονάδων και μηδενικών. Η συγγραφή και ανάγνωση, απο τον χρήστη, προγραμμάτων σε δυαδικό κώδικα είναι πολύ δύσκολη και χρονοβόρα διαδικασία για τους περισσότερους χρήστες. Για να ξεπεραστούν αυτές οι δυσκολίες, έχουν δημιουργηθεί κατα καιρούς πολλές γλώσσες προγραμματισμού, οι οποίες σκοπό έχουν να διευκολύνουν την συγγραφή προγραμμάτων σε μορφή η οποία να είναι κατανοητή απο τους χρήστες. Η μορφή αυτή των προγραμμάτων είναι βέβαια κατανοητή απο τον προγραμματιστή, αλλα όχι απο τον υπολογιστή. Χρειάζεται λοιπόν να παρεμβληθεί ένας μεταφραστής (Compiler), ο οποίος μεταφράζει το πρόγραμμα που είναι κατανοητό στον χρήστη, σε μορφή η οποία είναι κατανοητή απο τον υπολογιστή. Αυτή την μορφή προγράμματος μπορεί να εκτελέσει ο υπολογιστής, μετά απο προτροπή του χρήστη και όχι την αρχική μορφή την οποία καταλαβαίνει ο χρήστης και μόνον.

Γενικά υπάρχουν δύο ειδών γλώσσες προγραμματισμού: οι συμβολικές γλώσσες και οι γλώσσες ανώτερου επιπέδου. Η συμβολική γλώσσα είναι μία συμβολική αναπαράσταση δυαδικών εντολών, την οποία αντιλαμβάνεται άμεσα ο υπολογιστής και μπορεί να εκτελέσει τις εντολές χωρίς την παρεμβολή μεταφραστού (Compiler). Οι γλώσσες του τύπου αυτού είναι πολύ δύσκολες στην κατανόηση και την συγγραφή των εντολών και χρησιμοποιούνται απο πολύ λίγους ειδικούς του θέματος.

Οι γλώσσες προγραμματισμού ανώτερου επιπέδου αναπτύχθησαν για να διευκολύνουν τους χρήστες στην συγγραφή προγραμμάτων. Μια απο τις πρώτες γλώσσες που ορίστηκαν ήταν η FORTRAN (FORmula TRANslator) και εξακολουθεί να είναι μια απο τις περισσότερο χρησιμοποιούμενες γλώσσες για επιστημονικούς υπολογισμούς. Λόγω του μεγάλου πλήθους προγραμμάτων που υπάρχουν, η FORTRAN εξελίχθηκε με την πάροδο των ετών και η σημερινή της μορφή είναι διαφορετική απο αυτήν που ορίστηκε αρχικά, διατηρώντας όμως πολλά στοιχεία απο τον αρχικό σχεδιασμό της.

Ως εξέλιξη της FORTRAN μπορεί να θεωρηθεί η γλώσσα BASIC (Begginer's All-purpose Symbolic Instruction Code) με όλες τις μορφές που εξελίχθηκε (GW BASIC, Quick BASIC, Array BASIC κλπ.). Η γλώσσα BASIC προσαρμόζεται εύκολα σε εναν υπολογιστή και χρειάζεται μικρή μνήμη. Λόγω αυτών των πλεονεκτημάτων και του γεγονότος οτι οι κανόνες της είναι απλοί και λίγοι, έγινε η ευρύτερα

χρησιμοποιούμενη γλώσσα σε μικροϋπολογιστές. Οι ίδιοι λόγοι όμως την καθιστούν δύσχρητη έως ακατάλληλη για σύνθετα προγράμματα.

Η γλώσσα ALGOL (ALGOrithmic Language) προήλθε απο εξέλιξη της FORTRAN για την αντιμετώπιση προβλημάτων προγραμματισμού σύνθετων αλγορίθμων. Η ALGOL δίνει ένα πολύ καλό εργαλείο για την χρησιμοποίηση πολύπλοκων αλγορίθμων, όμως είναι αρκετά πολύπλοκη και δύσκολη στην εκμάθησή της.

Η Pascal είναι μία γλώσσα που προέρχεται απο τις γλώσσες ALGOL και PL/1 και είναι σχετικά εύκολη στην εκμάθηση όπως επίσης και κατάλληλη για προγραμματισμό πολύπλοκων αλγορίθμων.

Η γλώσσα C είναι μια ευρύτατα χρησιμοποιούμενη, απο πολλούς προγραμματιστές, γλώσσα προγραμματισμού. Η γλώσσα αυτή αναπτύχθηκε με σκοπό να διευκολύνει κύρια την ανάπτυξη λειτουργικών συστημάτων. Το μεγαλύτερο μέρος απο το λειτουργικό σύστημα UNIX είναι γραμμένο σε γλώσσα C. Επειδή έχει πολλές δυνατότητες και ευελιξία, είναι μια πολύ χρήσιμη γλώσσα προγραμματισμού γενικών εφαρμογών.

Υπάρχουν πάρα πολλές ακόμη γλώσσες, άλλες περισσότερο και άλλες λιγώτερο επιτυχημένες και δημοφιλείς, αλλά η έστω και συνοπτική περιγραφή τους ξεφεύγει απο τους σκοπούς του παρόντος. Διάφορες γλώσσες αναπτύχθησαν για να αντιμετωπίσουν συγκεκριμένα προβλήματα και εφαρμογές, με συνέπεια η περιοχή των εφαρμογών της κάθε μιας να είναι περιορισμένη. Αυτό που πρέπει να γίνει κατανοητό είναι οτι δέν υπάρχουν καλές και κακές γλώσσες προγραμματισμού. Όλες οι γλώσσες είναι καλές για τους σκοπούς που έχουν αναπτυχθεί. Είναι ευθύνη του προγραμματιστή να επιλέξει την κατάλληλη γλώσσα για να αναπτύξει την εφαρμογή που τον ενδιαφέρει. Επειδή είναι δύσκολο για κάποιον να μάθει όλες τις γλώσσες, ενδύκνειται η σωστή εκμάθηση μιάς γλώσσας η οποία να καλύπτει το μεγαλύτερο μέρος των αναγκών του.

Για τον κλάδο των Θετικών Επιστημών οι περισσότερο χρησιμοποιούμενες γλώσσες είναι οι : FORTRAN, Pascal και C. Οι βασικοί κανόνες των γλωσσών αυτών δεν διαφέρουν σημαντικά, με συνέπεια η σωστή εκμάθηση μιάς εξ αυτών να διευκολύνει την εκμάθηση των άλλων δύο. Το παρόν σύγγραμμα καλύπτει τις βασικές ανάγκες προγραμματισμού σε γλώσσα FORTRAN.

1.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΓΛΩΣΣΑΣ FORTRAN

Η γλώσσα προγραμματισμού FORTRAN που το όνομά της προέρχεται από τον όρο FORmula TRANslator, είναι η περισσότερο διαδεδομένη γλώσσα για την

επίλυση προβλημάτων των θετικών επιστημών και ιδιαίτερα της Φυσικής. Αν και σε διάφορες εποχές εκφράστηκε η άποψη ότι δεν είναι ευέλικτη και εξελιγμένη σαν γλώσσα, εν τούτοις ο υπάρχων πλούτος προγραμμάτων την καθιστά μια από τις πιό χρήσιμες και διαδεδομένες γλώσσες προγραμματισμού, ιδιαίτερα σε προβλήματα Μαθηματικών και Φυσικής.

Κάθε γλώσσα προγραμματισμού αποτελεί ένα υποσύνολο μιας γλώσσας καθομιλουμένης και συνήθως αυτή είναι η αγγλική γλώσσα. Το περιορισμένο υποσύνολο εννοιών της αγγλικής γλώσσας οι οποίες δεν επιδέχονται παρερμηνεία και υπόκεινται σε σαφείς περιορισμούς αποτελεί το αλφάβητό τους.

Η γλώσσα FORTRAN είναι μια απο τις λεγόμενες γλώσσες ανώτερου επιπέδου και όπως κάθε τέτοια γλώσσα, για να εκτελεστεί από υπολογιστή χρειάζεται τη μεσολάβηση ενός ειδικού προγράμματος που ονομάζεται διερμηνέας ή μεταγλωττιστής (Compiler) με τον οποίο μεταφράζεται το πρόγραμμα σε μια διαδοχή δυαδικών αριθμών κατανοητών στον υπολογιστή.

Με τον όρο πρόγραμμα εννοείται ένα σύνολον εντολών που έχουν σχεδιαστεί και καταγραφεί έτσι ώστε όταν εκτελούνται από έναν υπολογιστή, να επιλύουν ένα συγκεκριμένο πρόβλημα. Έτσι έχουν γραφεί προγράμματα για παιχνίδια, εργασίες διαχείρισης επιχειρήσεων, επιστημονικών υπολογισμών κλπ.

Υπάρχουν πολλές εκδόσεις της γλώσσας προγραμματισμού FORTRAN, από διάφορους οίκους, με μικρές διαφορές μεταξύ τους και για διάφορους τύπους υπολογιστών. Ανεξάρτητα από τις διαφορές που έχουν μεταξύ τους, όλες υπακούουν σε ορισμένους κανόνες που ορίζονται στο American National Standards Institute (ANSI). Στο βιβλίο αυτό περιγράφονται οι εντολές εκείνες οι οποίες είναι κοινές για όλες τις εκδόσεις της FORTRAN (Standard Fortran).

Αν και τα περισσότερα παραδείγματα που θα αναφερθούν στην ανάπτυξη των διαφόρων θεμάτων είναι μαθηματικές εφαρμογές, αυτό δεν πρέπει να θεωρηθεί ως μειονέκτημα. Τα παραδείγματα πολλές φορές είναι επιλεγμένα έτσι ώστε να τονίσουν μια συγκεκριμένη εφαρμογή ή εντολή και όχι να αναπτύξουν την καλύτερη διαδικασία επίλυσης ενός γενικώτερου προβλήματος. Είναι σίγουρο οτι μπορούν να βρεθούν πολλοί και κομψότεροι τρόποι για την επίλυση των προβλημάτων στα παραδείγματα που αναφέρονται.

Ο αναγνώστης θα πρέπει να κατανοήσει ότι, όπως όλες οι γλώσσες προγραμματισμού, έτσι και η FORTRAN δεν εμπεδώνεται διαβάζοντας ένα ή περισσότερα βιβλία άλλά μόνον με την πρακτική εξάσκηση. Συνιστάται λοιπόν προς τον αναγνώστη η εντατική εξάσκηση. Ως συνεισφορά στην καλλίτερη κατανόηση της θεωρίας θα αναγράφονται παραδείγματα και κατόπιν θα αναλύονται τα προγράμματα βήμα προς βήμα για να τονισθεί η λειτουργία κάθε εντολής.

2. ΓΕΝΙΚΟΙ ΚΑΝΟΝΕΣ

Η FORTRAN, όπως και άλλες γλώσσες προγραμματισμού Η/Υ, ακολουθεί κάποιους κανόνες. Για τη σωστή συγγραφή ενός προγράμματος, το οποίο θα επιλύει ενα συγκεκριμένο πρόβλημα, θα πρέπει κατ' αρχήν να ακολουθούνται αυτοί οι απλοί αλλα ουσιαστικοί κανόνες. Βασικό σημείο είναι η κατανόηση του προβλήματος και η σχεδίαση των βημάτων που απαιτούνται για την ορθή επίλυσή του. Η καταγραφή του προγράμματος στο χαρτί πρέπει να είναι το επόμενο βήμα. Θα πρέπει να τονιστεί ότι οι Η/Υ δεν κάνουν λάθη, αλλά εκτελούν ακριβώς τις εντολές του προγραμματιστή.

Το πρόγραμμα που γράφεται σε γλώσσα FORTRAN δεν είναι ακριβώς η μορφή με την οποία εκτελείται από τον υπολογιστή, αλλά ένας ενδιάμεσος κώδικας επικοινωνίας χρήστη και υπολογιστή. Η γλώσσα αυτή επικοινωνίας, που είναι άμεσα κατανοητή από τον χρήστη, έχει το δικό της αλφάβητο, που αποτελείται από τα 26 γράμματα του λατινικού αλφάβητου (A-Z), τους αριθμούς από το 0-9, καθώς επίσης και από ειδικούς χαρακτήρες όπως =, +, -, *, /, (,), ..., $, ', : και το κενό διάστημα. Το δεκαδικό μέρος ενός αριθμού ορίζεται από την τελεία (.) και όχι από το κόμμα (,), ενώ το πρόσημο (+ ή -) πρέπει υποχρεωτικά να αναγράφεται για τους αρνητικούς αριθμούς και προαιρετικά για τους θετικούς.

Ορισμένα σημεία που αξίζει να σημειωθούν είναι ο ορισμός της τιμής μιας μεταβλητής. Π.χ. η εντολή x=x+2. δεν έχει κανένα νόημα στην άλγεβρα, αλλά είναι απόλυτα σωστή στην FORTRAN. Επίσης για λόγους οικονομίας χώρου αποθήκευσης, χρόνου και τρόπου επεξεργασίας των πράξεων, γίνεται σαφής διαχωρισμός μεταξύ ακεραίων και πραγματικών αριθμών, παρ' όλο που οι πρώτοι αποτελούν υποσύνολο των δεύτερων. Αυτό θα γίνει περισσότερο κατανοητό στα παρακάτω κεφάλαια.

Για να γραφεί ένα πρόγραμμα FORTRAN χρειάζεται να χρησιμοποιηθεί ένα άλλο πρόγραμμα το οποίο ονομάζεται επεξεργαστής κειμένου (editor). Το ακριβές όνομα του editor εξαρτάται από το τί είναι διαθέσιμο στον υπολογιστή στον οποίο πρόκειται να γραφεί το πρόγραμμα και θα πρέπει να είναι γνωστός ο τρόπος με τον οποίο καλείται για εκτέλεση.

Στο σημείο αυτό θα πρέπει να αναφερθούν ορισμένοι κανόνες σχετικά με το πώς θέλει η FORTRAN να είναι γραμμένο ένα πρόγραμμα.

1. Όλες οι εκτελέσιμες εντολές γράφονται ανά μία σε κάθε γραμμή και από τη στήλη 7 έως και 72. Αν μία εντολή είναι αρκετά μεγάλη και δεν χωρά μέσα σε αυτό το διάστημα, τότε η εντολή μπορεί να συνεχιστεί στην επόμενη γραμμή, βάζοντας έναν οποιοδήποτε χαρακτήρα στην 6η στήλη.

2. Ως πρώτη εντολή θα πρέπει απαραίτητα να αναφέρεται το όνομα του προγράμματος π.χ. PROGRAM PNAME, όπου PNAME το όνομα του προγράμματος το οποίο θα πρέπει να περιέχει μέχρι 7 αλφαριθμητικούς χαρακτήρες και καλό θα είναι ο πρώτος χαρακτήρας να μην είναι αριθμός. Με την έννοια αλφαριθμητικοί χαρακτήρες εννοούνται τα 26 γράμματα του λατινικού αλφαβήτου και οι 10 αριθμοί από 0-9. Δεν επιτρέπονται οι άλλοι ειδικοί χαρακτήρες.
3. Είναι χρήσιμο, αλλά όχι υποχρεωτικό, να υπάρχουν σχόλια σε διάφορα σημεία του προγράμματος, τα οποία διευκολύνουν τον προγραμματιστή πληροφορώντας τον για την ενέργεια που πρόκειται να ακολουθήσει ή για τις ενέργειες που προηγήθηκαν. Τα σχόλια αυτά δηλώνονται με το γράμμα C στην πρώτη στήλη της γραμμής, δεν εκτελούνται από τον υπολογιστή, μπορούν δε να υπάρχουν πολλές γραμμές με σχόλια σε διάφορα σημεία του προγράμματος.
4. Σε κάθε εντολή όπου δηλώνεται η τιμή μιάς μεταβλητής (π.χ. Y=3.), αριστερά του συμβόλου = θα πρέπει να υπάρχει το όνομα μιας μεταβλητής και δεξιά του συμβόλου ένας αριθμός ή κάποια υπολογίσιμη ποσότητα, απλή ή πολύπλοκη. Δέν επιτρέπεται η οποιαδήποτε πράξη αριστερά του συμβόλου = , π.χ. η έκφραση X+Y=4. είναι λάθος.

Παράδειγμα

Ζητείται να υπολογιστεί η τιμή της ποσότητας Z=X+Y, όταν X=2. και Y=3. και νά τυπωθούν τα αποτελέσματα.

```
      PROGRAM PPP
C YPOLOGISMOS ATHRISMATOS
      X=2.
      Y=3.
      Z=X+Y
      WRITE(*,*)X,Y,Z
      STOP
      END
```

Το πρόγραμμα αυτό υπολογίζει την τιμή της παράστασης Z=X+Y. Η πρώτη εντολή PROGRAM PPP δεν είναι εκτελέσιμη, αλλά είναι απαραίτητη για τη λειτουργία του προγράμματος, δηλώνει δε οτι το όνομα του προγράμματος είναι PPP. Η δεύτερη γραμμή C YPOLOGISMOS ATHRISMATOS, είναι ένα σχόλιο που πληροφορεί το χρήστη σχετικά με το τί κάνει αυτό το πρόγραμμα και δεν είναι

εκτελέσιμη εντολή. Η τρίτη γραμμή X=2. είναι μία εντολή FORTRAN που δηλώνει στον υπολογιστή ότι η τιμή της μεταβλητής X είναι ίση με 2. και η επόμενη γραμμή δηλώνει ότι η τιμή της μεταβλητής Y είναι 3.. Ας σημειωθεί ότι το σύμβολο = δεν έχει ακριβώς τη σημασία που έχει το αλγεβρικό αντίστοιχο σύμβολο, αλλά την έννοια ότι η μεταβλητή που βρίσκεται αριστερά του =, παίρνει την τιμή της ποσότητας που βρίσκεται δεξιά του συμβόλου. Έτσι λοιπόν οι εντολές X=2. και Y=3. μεταφράζονται σαν: Η μεταβλητή X παίρνει την τιμή 2. και η Y την τιμή 3. . Η επόμενη εντολή Z=X+Y μεταφράζεται σαν: η τιμή της μεταβλητής Z παίρνει την τιμή που έχει το άθροισμα των τιμών των μεταβλητών X και Y.

Οι επιτρεπτές πράξεις στη FORTRAN είναι:

+	**Πρόσθεση**	*	**Πολλαπλασιασμός**
-	**Αφαίρεση**	/	**Διαίρεση**
		**	**Ύψωση σε δύναμη**

Αν και όλες οι πράξεις στους Η/Υ ανάγονται τελικά σε αυτές της πρόσθεσης και της αφαίρεσης, η σειρά με την οποία εκτελούνται εφ' όσον εμφανίζονται όλες στην ίδια έκφραση είναι κατά σειρά: η ύψωση σε δύναμη (**), ο πολλαπλασιασμός (*) και η διαίρεση (/), η πρόσθεση (+) και η αφαίρεση (-). Στην περίπτωση ισοδύναμων τελεστών (π.χ. πολλαπλασιασμού και διαίρεσης) ο συνήθης κανόνας είναι οι πράξεις να εκτελούνται από αριστερά προς τα δεξιά.

Η επόμενη εντολή είναι η WRITE(*,*)X,Y,Z που ζητά από τον υπολογιστή να γράψει τις τιμές των X, Y, Z στην οθόνη του τερματικού. Έτσι στην οθόνη θα εμφανιστούν οι τιμές ως εξής:

2.000000 3.000000 5.000000

Η εντολή STOP δηλώνει στον υπολογιστή ότι πρέπει να σταματήσει η εκτέλεση του προγράμματος. Η επόμενη εντολή END δηλώνει στον υπολογιστή ότι δεν υπάρχει άλλη εκτελέσιμη εντολή στο πρόγραμμα.

Η εντολή PROGRAM PNAME πρέπει να είναι υποχρεωτικά η πρώτη εντολή σε ένα πρόγραμμα και η εντολή END η τελευταία. Η εντολή STOP δεν είναι υποχρεωτικό να αναγράφεται πριν απο την εντολή END σε ένα πρόγραμμα, ενώ μπορούν να υπάρχουν πολλές εντολές STOP στο ίδιο πρόγραμμα. Από τη στιγμή όμως που θα εκτελεστεί ένα από όλα, τότε η ροή του προγράμματος θα σταματήσει αυτόματα στο σημείο αυτό. Σε επόμενα παραδείγματα αυτό θα γίνει πιο κατανοητό.

Επανερχόμενοι στην εντολη WRITE(*,*)X,Y,Z, σημειώνεται ότι η εντολή αυτή είναι ισοδύναμη με τις εντολές

```
WRITE(*,*)X
WRITE(*,*)Y
WRITE(*,*)Z
```

με την έννοια οτι θα τυπωθούν και στις δύο περιπτώσεις οι τιμές των μεταβλητών X, Y, Z. Η ακολουθία όμως των εντολών, στην τελευταία περίπτωση, είναι διαφορετική απο την προηγούμενη και δηλώνει στον υπολογιστή ότι θα τυπώνει την τιμή κάθε μεταβλητής σε μια γραμμή και κατόπιν θα αλλάζει γραμμή για να τυπώσει την επόμενη.

Με το όνομα μεταβλητή νοείται κάθε ποσότητα η οποία μπορεί να αλλάζει κατά τη διάρκεια της εκτέλεσης του προγράμματος. Κάθε μεταβλητή μπορεί να εμφανίζεται με το όνομα που επιλέγει ο προγραμματιστής, υπακούοντας στους εξής τρείς κανόνες: (i) Το όνομα της μεταβλητής πρέπει να αποτελείται από 1 μέχρι 6 αλφαριθμητικούς χαρακτήρες. (ii) ο πρώτος χαρακτήρας δεν πρέπει να είναι άλλος εκτός από γράμμα και (iii) δεν πρέπει να περιέχονται άλλοι χαρακτήρες εκτός απο γράμματα και αριθμούς.

Εφ' όσον η FORTRAN κάνει διαχωρισμό μεταξύ ακεραίων και πραγματικών σταθερών, είναι επόμενο να γίνεται αντίστοιχος διαχωρισμός για ακέραιες και πραγματικές μεταβλητές. Υπάρχουν δύο τρόποι με τους οποίους γίνεται ο διαχωρισμός αυτός. Ο πρώτος άμεσος τρόπος, ο οποίος είναι ο πλέον ασφαλής και συνιστάται ιδιαίτερα σε προγραμματιστές με μικρή εμπειρία, γίνεται από την δήλωση του ονόματος της μεταβλητής. Κάθε όνομα μεταβλητής που αρχίζει με έναν από τους χαρακτήρες I, J, K, L, M, N και ανεξάρτητα ποιοί άλλοι χαρακτήρες ακολουθούν, δηλώνουν ότι η μεταβλητή είναι ακέραιου τύπου. Αν το όνομα της μεταβλητής αρχίζει με οποιοδήποτε άλλο γράμμα, τότε η μεταβλητή είναι πραγματικού τύπου.

Παραδείγματα:

ALPHA	(πραγματική)
BETA	(πραγματική)
K13	(ακέραια)
2BI	(λάθος διότι ο πρώτος χαρακτήρας είναι αριθμός)
VI	(πραγματική)
IB	(ακέραια)
COUNT	(πραγματική)
ICOUNT	(ακέραια)
DERIVATIVE	(λάθος διότι περιέχει περισσότερους από 6 χαρακτήρες)

Υπάρχει επίσης ο έμμεσος τρόπος δήλωσης, με τον οποίο ο τύπος της μεταβλητής καθορίζεται στην αρχή του προγράμματος και ισχύει για όλο τον πρόγραμμα. Γράφοντας την εντολή

INTEGER VARN

δηλώνεται ότι η μεταβλητή με το όνομα VARN θα είναι μεταβλητή ακέραιου τύπου. παρ' όλο που ο πρώτος χαρακτήρας V δηλώνει μεταβλητή πραγματικού τύπου. Αντίστοιχα η εντολή

REAL IVAR, MIN, MAX

δηλώνει ότι οι μεταβλητές IVAR,MIN,MAX είναι πραγματικές ανεξάρτητα από τον πρώτο χαρακτήρα.

3. ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ

Όπως έχει αναφερθεί μέχρι τώρα, οι εντολές ενός προγράμματος εκτελούνται διαδοχικά η μία μετά την άλλη. Πολλές φορές παρουσιάζεται η ανάγκη να μεταφερθεί η ροή του προγράμματος από μία γραμμή σε κάποια άλλη και η οποία δεν είναι η επόμενη. Υπάρχουν δυο εντολές, η GOTO και η IF, καθώς επίσης και ο συνδυασμός αυτών, που δίνουν αυτή τη δυνατότητα στο χρήστη.

Η ΕΝΤΟΛΗ GOTO

Η εντολή GOTO δίνει τη δυνατότητα παράκαμψης κάποιων εντολών και εμφανίζεται συνήθως με τις δύο ακόλουθες μορφές.

UNCONDITIONAL (Χωρίς όρους)

COMPUTED (Μετά από υπολογισμό)

3.1 Η ΕΝΤΟΛΗ UNCONDITIONAL GOTO

Είναι η απλούστερη μορφή της εντολής GOTO και συντάσσεται ως εξής:

GOTO ITARG ή GO TO ITARG

όπου ITARG είναι ένας ακέραιος αριθμός αναφοράς, το πολύ πενταψήφιος, που βρίσκεται σε κάποια εντολή του προγράμματος πριν ή μετά την εντολή GOTO.

Παράδειγμα

Να υπολογιστεί το άθροισμα των τετραγώνων της ακολουθίας $1^2+2^2+3^2+...$

```
      PROGRAM GOTON
C     CAUTION !!!  ENDLESS  PROGRAM
      X=0.
      S=0.
10    CONTINUE
      X=X+1.
      S=S+X**2
      WRITE (*,*)X,S
      GOTO 10
      STOP
      END
```

Στο παράδειγμα κατ'αρχήν μηδενίζονται οι τιμές των μεταβλητών X και S. Η εντολή 10 CONTINUE, δεν εκτελεί καμμία εργασία και μεταβιβάζει την ροή του προγράμματος στην αμέσως επόμενη εντολή X=X+1., όπου το X παίρνει την τιμή 1.. Κατόπιν υπολογίζεται η ποσότητα S=S+X**2 , που θα πάρει την τιμή S=0.+1.**2=1. και με την επόμενη εντολή WRITE(*,*)X,S θα τυπωθούν οι τιμές τους στην οθόνη του τερματικού. Η εντολή GOTO 10 μεταφέρει την ροή του προγράμματος στο 10 CONTINUE και απο εκεί στην X=X+1., όπου τώρα η τιμή του X=1.+1.=2. και αντίστοιχα η τιμή του S γίνεται S=1.+2.**2=5. κ.ο.κ..

Το πρόγραμμα αυτό υπολογίζει το άθροισμα των τετραγώνων των όρων της ακολουθίας 1,2,3,... χωρίς να σταματά ποτέ. Και τούτο διότι δεν υπάρχει περιορισμός ο οποίος να εξασφαλίζει τον ομαλό τερματισμό της εκτέλεσής του. Η εκτέλεση θα σταματήσει όταν η τιμή της μεταβλητής S ξεπεράσει τα όρια που είναι δυνατόν να δεχθεί ο Η/Υ.

Είναι υποχρεωτικό δε ο αριθμός αναφοράς (10 στο παράδειγμα) να βρίσκεται σε κάποια εκτελέσιμη εντολή του προγράμματος.

Παράδειγμα

```
      PROGRAM TESTGO
      A=1.
      B=2.
      C=3.
      Y=A*B*C
      Z=A+B**C
      WRITE(*,*)A,B,C,Y,Z
      GO TO 10
      X=C-B+A
      WRITE(*,*)A,B,C,X
10    CONTINUE
      STOP
      END
```

Η πρώτη εντολή PROGRAM TESTGO ορίζει ότι το όνομα του προγράμματος είναι TESTGO. Οι επόμενες τρείς εντολές ορίζουν τις αντίστοιχες τιμές στις μεταβλητές A, B, C, ενώ οι επόμενες δύο εντολές ορίζουν τις πράξεις που επιθυμεί ο προγραμματιστής να γίνουν μεταξύ των ήδη ορισμένων πραγματικών μεταβλητών A, B, C για να προκύψουν οι τιμές των μεταβλητών Y, Z. Το σύμβολο * στην έκφραση A*B σημαίνει ότι πολλαπλασιάζεται η τιμή της μεταβλητής B με αυτήν της A. Το σύμβολο ** στην έκφραση B**C σημαίνει ύψωση σε δύναμη όπου το B είναι η βάση και C ο εκθέτης (B^C). Η επόμενη εντολή WRITE(*,*)A,B,C,Y,Z

δηλώνει στον υπολογιστή ότι πρέπει να τυπώσει στην οθόνη του τερματικού τις τιμές των πραγματικών μεταβλητών A, B, C, Y, Z με τη σειρά που δηλώνεται. Έτσι στην οθόνη θα εμφανιστούν οι τιμές:

1.000000 2.000000 3.000000 6.000000 9.000000

Η επόμενη εντολή GOTO 10 υποχρεώνει τον υπολογιστή να μην εκτελέσει τις εντολές που ακολουθούν και να βρεθεί στη γραμμή εκείνη η οποία στις στήλες 1-5 περιέχει τον αριθμό 10. Δεν σημαίνει τη 10η γραμμή του προγράμματος ούτε άλλον αριθμό που περιέχει τον 10 (π.χ. 110, 1010 κλπ.). Έτσι η ροή του προγράμματος στο παράδειγμα θα μεταφερθεί στην εντολή 10 CONTINUE, υπερπηδώντας τις εντολές

```
X=C-B+A
WRITE(*,*)A,B,C,X
```

οι οποίες δεν εκτελούνται. Η εντολή CONTINUE είναι μια εκτελέσιμη εντολή, η οποία όμως είναι αδρανής και πληροφορεί τον υπολογιστή να προχωρήσει στην εκτέλεση της επόμενης εντολής που στην προκειμένη περίπτωση είναι η εντολή STOP. Έτσι θα σταματήσει η εκτέλεση του προγράμματος.

Παράδειγμα

```
      PROGRAM TESTGOB
C  CAUTION !!!  ENDLESS  PROGRAM
      X=1.
      Y=3.
10    CONTINUE
      X=X+1.
      Y=Y+1.
      Z=X+2.*Y
      WRITE(*,*)X,Y,Z
      GOTO 10
      STOP
      END
```

Στο παράδειγμα αυτό μετά τη δήλωση του προγράμματος, ορίζονται οι τιμές των μεταβλητών X και Y. Η εντολή 10 CONTINUE δεν εκτελεί καμμία εργασία και εκτελείται η επόμενη εντολή X=X+1.. Η εντολή αυτή ενώ δεν έχει έννοια ως αλγεβρική σχέση, στην FORTRAN σημαίνει: Δώσε στην μεταβλητή X την τιμή που είχε μέχρι τώρα αφού προσθέσεις μία μονάδα. Έτσι λοιπόν αφού αρχικά η τιμή του X ήταν 1. με την εντολή X=X+1. η τιμή του X=2.. Ομοίως η επόμενη εντολή Y=Y+1. ορίζει την τιμή της μεταβλητής Y=4.. Η επόμενη εντολή Z=X+2.*Y δίνει στο Z την τιμή 10. και με την επόμενη εντολή WRITE(*,*)X,Y,Z δηλώνει στον

υπολογιστή να τυπώσει τις τιμές των X,Y,Z στην οθόνη του υπολογιστή, όπου θα εμφανιστούν οι τιμές

2.000000 4.000000 10.000000

Με την επόμενη εντολή GOTO 10, η ροή του προγράμματος μεταφέρεται στην επόμενη εντολή που έχει τον αριθμό 10 που στην προκειμένη περίπτωση είναι η 10 CONTINUE. Η εντολή αυτή μεταφέρει την ροή του προγράμματος στην αμέσως επόμενη γραμμή X=X+1. . Η τιμή του X ήταν 2., έτσι με αυτή την εντολή το X=3. και στην επόμενη το Y=5. και ακολούθως Z=13. και με την εντολή WRITE(*,*)X,Y,Z τυπώνονται οι τιμές των X,Y,Z στην οθόνη του τερματικού

3.000000 5.000000 13.000000

Η εντολή GOTO 10 μεταφέρει και πάλι την ροή στην εντολή 10 CONTINUE. Είναι προφανές ότι το πρόγραμμα αυτό δεν πρόκειται να σταματήσει ποτέ, αφού δεν εκτελείται μία από τις εντολές STOP ή END.

3.2 Η ΕΝΤΟΛΗ COMPUTED GOTO

Η εντολή αυτή συντάσσεται ως εξής

GOTO (I1,I2,I3,....IN)ITOT

όπου οι μεταβλητές I1,I2,....IN είναι ακέραιες σταθερές και υπάρχουν ως αριθμοί αναφοράς στο πρόγραμμα. Το ITOT είναι το όνομα ακέραιας μεταβλητής και οι τιμές που μπορεί να παίρνει είναι μέχρι το πλήθος των I1,I2,I3,....IN. Η εντολή

GOTO(40,50,3,70)IT

θα μεταφέρει την ροή του προγράμματος στην εντολή αναφοράς 40 αν η τιμή του IT=1, στην εντολή 50 αν IT=2, στην 3 αν IT=3 ή στην 70 αν IT=4.

Παράδειγμα

```
      PROGRAM TESTCG
C  CAUTION !!!  ENDLESS  PROGRAM
      X=2.
      Y=3.
      Z=4.
      I=0
5     I=I+1
      GOTO(10,20,30)I
10    W=X+Y+Z
      WRITE(*,*)I,X,Y,Z,W
      GOTO 40
20    W=X+Y-Z
      WRITE(*,*)I,X,Y,Z,W
      GOTO 40
```

```
30  W=X*Y*Z
    WRITE(*,*)I,X,Y,Z,W
40  CONTINUE
    GOTO 5
50  CONTINUE
    STOP
    END
```

Στην αρχή του προγράμματος ορίζονται οι τιμές των μεταβλητών X, Y, Z, I και με την εντολή I=I+1 δίνεται η τιμή I=1. Με την εντολή GOTO(10,20,30) I, η ροή του προγράμματος μεταφέρεται (επειδή I=1) στην εντολή 10 W=X+Y+Z, που παίρνει την τιμή W=2.+3.+4.=9.. Με την εντολή WRITE(*,*)I,X,Y,Z,W τυπώνονται στην οθόνη του τερματικού οι τιμές αυτών των μεταβλητών ως εξής:

1 2.000000 3.000000 4.000000 9.000000

και με την εντολή GOTO 40 η ροή του προγράμματος μεταφέρεται στο 40 CONTINUE, όπου αμέσως μετά είναι η εντολή GOTO 5. Έτσι η ροή μεταφέρεται στην εντολή 5 I=I+1, όπου η τιμή του I γίνεται ίση με 2. Στην περίπτωση αυτή η ροή του προγράμματος θα μεταφερθεί, με βάση την εντολή GOTO(10,20,30) I, στην εντολή 20 W=X+Y-Z όπου η τιμή του είναι W=2.+3.-4.=1. κ.λ.π..

3.3 Η ΕΝΤΟΛΗ IF

Υπάρχουν δύο ειδών εντολές IF . Το αριθμητικό και το λογικό IF.

(i) Το αριθμητικό IF συντάσσεται ως εξής

IF(EXPR) I,J,K

όπου EXPR είναι μια αριθμητική έκφραση και I,J,K αριθμοί αναφοράς εντολών. Η εντολή λειτουργεί ως εξής. Αν η τιμή της EXPR είναι μικρότερη του μηδενός τότε η ροή του προγράμματος μεταφέρεται στην εντολή με αριθμό αναφοράς I. Αν η τιμή του EXPR είναι ίση με μηδέν μεταφέρεται στην εντολή με αριθμό αναφοράς J ενώ αν η τιμή του EXPR είναι μεγαλύτερη από μηδέν, η ροή μεταφέρεται στην εντολή με αριθμό αναφοράς K. Τα I,J ή I,K ή J,K μπορούν να έχουν την ίδια τιμή. Είναι προφανές οτι οι αριθμοί αναφοράς I, J, K πρέπει να εμφανίζονται μέσα στο πρόγραμμα.

Παράδειγμα

```
PROGRAM TESTIN
X=5.
Y=3.
IF(X-Y)10,20,30
```

```
10  Z=X-Y
    GOTO 50
20  Z=X+Y
    GOTO 50
30  Z=X*Y
50  CONTINUE
    WRITE(*,*) X,Y,Z
    STOP
    END
```

Ορίζονται οι τιμές των μεταβλητών X=5., Y=3. και με την εντολή IF(X-Y)10,20,30 η ροή του προγράμματος μεταφέρεται στην εντολή 30 Z=X*Y αφού η τιμή της ποσότητας (X-Y) = 5.-3.=2. είναι θετικός αριθμός.

Παράδειγμα

```
    PROGRAM TESTIN2
    X=3.
    Y=5.
    IF(X-Y)10,10,30
10  Z=X-Y
    GOTO 50
30  Z=X*Y
50  CONTINUE
    WRITE(*,*) X,Y,Z
    STOP
    END
```

Ορίζονται οι τιμές των μεταβλητών X=3., Y=5. και με την εντολή IF(X-Y)10.10,30 η ροή του προγράμματος μεταφέρεται στην εντολή 10 Z=X-Y αφού η τιμή της ποσότητας (X-Y) = 3.-5.=-2. είναι αρνητικός αριθμός.

(ii) Η εντολή για το λογικό IF συντάσσεται ως εξής:

IF(EXPR) ACTION

όπου EXPR είναι μια Λογική(Logical) ή Συγκριτική(Relational) έκφραση και ACTION είναι μια εκτελέσιμη εντολή. Η δράση της εντολής αυτής είναι η εξής: Αν η έκφραση EXPR είναι αληθής (.TRUE.) τότε εκτελείται η εντολή ACTION. Αν η έκφραση EXPR είναι ψευδής (.FALSE.) τότε δεν εκτελείται η εντολή ACTION και η ροή του προγράμματος μεταφέρεται στην αμέσως επόμενη εκτελέσιμη εντολή.

Παράδειγμα

```
      PROGRAM TESTGOC
      X=1.
      Y=3.
10    CONTINUE
      X=X+1.
      Y=Y+1.
      IF(X.GT.20.) GO TO 15
      Z=X+2.*Y
      WRITE(*,*)X,Y,Z
      GOTO 10
15    CONTINUE
      STOP
      END
```

Το παράδειγμα είναι το ίδιο με το πρόγραμμα TESTGOB, μόνον που έχει προστεθεί η εντολή IF(X.GT.20.)GOTO 15. Η έκφραση που βρίσκεται μέσα στην παρένθεση διαβάζεται "το X μεγαλύτερο του 20." και ο τελεστής σύγκρισης πρέπει απαραίτητα να βρίσκεται μεταξύ δύο τελειών. Το .GT. είναι ένας τελεστής που προέρχεται από τα ακρονύμια της αγγλικής έκφρασης Greater Than (Μεγαλύτερο Από) και αλγεβρικά μπορεί να παρασταθεί με το X>20. Η πλήρης εντολή λοιπόν μεταφράζεται ως εξής: Αν (IF) η πρόταση X.GT.20. είναι αληθής, τότε εκτελείται η συνέχεια της εντολής GOTO 15 και η ροή του προγράμματος μεταφέρεται στην εντολή με τον αριθμό 15 CONTINUE. Αν η πρόταση X.GT.20. είναι ψευδής τότε δεν εκτελείται η συνέχεια της εντολής αλλά μεταφέρεται στην αμέσως επόμενη εκτελέσιμη εντολή του προγράμματος.

Οι τελεστές σύγκρισης, όπως αυτός που προηγήθηκε, φαίνονται στον παρακάτω πίνακα με τη σημασίας τους.

Τελεστής	**Αλγεβρική Έκφραση**	**Αγγλική Έκφραση**	**Ελληνική Έκφραση**
.EQ.	=	Equal to	Ίσο με
.NE.	≠	Not Equal to	Όχι ίσο με
.LT.	<	Less Than	Μικρότερο από
.LE.	≤	Less or Equal	Μικρότερο ή Ίσο
.GT.	>	Greater Than	Μεγαλύτερο από
.GE.	≥	Greater or Equal	Μεγαλύτερο ή Ίσο

Η εντολή IF μπορεί να περιλαμβάνει περισσότερες απο μία υποθέσεις, οι οποίες συνδέονται μεταξύ τους με τις λογικές εκφράσεις:

.AND. Η IF ικανοποιείται μόνον αν όλες οι υποθέσεις είναι αληθείς.

.OR. Η IF ικανοποιείται αν έστω και μία απο τις υποθέσεις είναι αληθής.

.XOR. Η IF ικανοποιείται αν η πρώτη υπόθεση είναι αληθής και η δεύτερη ψευδής ή αντίστοφα.

.EQV. Η IF ικανοποιείται αν και οι δύο υποθέσεις έχουν την ίδια λογική τιμή (σωστή ή ψευδής).

Η έκφραση

```
IF((A.EQ.B).AND.(C.EQ.D)) GOTO 10
```

σημαίνει: αν το A είναι ίσο με το B και το C είναι ίσο με το D, τότε GOTO 10. Αν δεν ικανοποιούνται συγχρόνως και οι δύο υποθέσεις, τότε δεν ικανοποιείται το IF και η ροή του προγράμματος μεταφέρεται στην αμέσως επόμενη εκτελέσιμη εντολή του προγράμματος. Αντίστοιχα η έκφραση

```
IF((A.EQ.B).OR.(C.EQ.D)) GOTO 20
```

σημαίνει οτι αν ικανοποιείται μία απο τις δύο συνθήκες (ή και οι δύο), τοτε ικανοποιείται το IF και η ροή του προγράμματος μεταφέρεται στην εντολή με αριθμό αναφοράς 20. Διαφορετικά η ροή μεταφέρεται στην αμέσως επόμενη εκτελέσιμη εντολή.

Θα πρέπει να τονιστεί ιδιαίτερα οτι με την χρήση του λογικού IF συγκρίνονται ομοειδείς ποσότητες. Δεν είναι δυνατόν να γίνει σύγκριση της τιμής μιας πραγματικής μεταβλητής με την τιμή μιας μεταβλητής ακέραιου τύπου, διότι το αποτέλεσμα θα είναι λανθασμένο, έστω και αν ο υπολογιστής δεν δώσει διαγνωστικό μήνυμα.

Μια άλλη μορφή της εντολής IF είναι αυτή στην οποία χρησιμοποιείται το THEN (Τότε) και διευκολύνει όταν είναι επιθυμητή η εκτέλεση μιας σειράς εντολών και όχι μόνον μιάς. Στο παράδειγμα που ακολουθεί αυτό γίνεται πιο κατανοητό.

Παράδειγμα

```
      PROGRAM TIFTH
      X=1.
      Y=2.
10    CONTINUE
      X=X+1.
      Y=Y+1.
      IF (X.GT.10.) GO TO 20
      IF (X.GT.3.) THEN
      Z=2.*X+Y**2
      ELSE
      Z=2.*X+Y**3
      ENDIF
```

```
      WRITE (*,*)X,Y,Z
      GO TO 10
20    CONTINUE
      STOP
      END
```

Στο πρόγραμμα ορίζονται οι τιμές των μεταβλητών X=1. και Y=2.. Μετά το 10 CONTINUE οι τιμές των X και Y αυξάνονται κατά μία μονάδα και στην επόμενη εντολή ελέγχεται αν η τιμή του X ξεπεράσει το 10, οπότε η ροή μεταφέρεται στην εντολή 20 CONTINUE που οδηγεί στο STOP και END. Αν η τιμή του X είναι μικρότερη του 10., με την εντολή

IF(X.GT.3.) THEN

ελέγχεται αν η τιμή του X είναι μεγαλύτερη του 3.. Αν αυτό είναι σωστό τότε (THEN) υπολογίζεται η ποσότητα Z=2.*X+Y**2 . Αν το IF δεν ικανοποιείται τότε (ELSE) υπολογίζεται η τιμή της ποσότητας Z=2.*X+Y**3. Η εντολή ENDIF δηλώνει το τέλος του IF...THEN...ELSE. Η ακολουθία των εντολών IF...THEN...ELSE...ENDIF ονομάζεται BLOCK IF και μπορεί να περιέχει και άλλα IF ή και BLOCK IF. Το τμήμα μεταξύ ELSE και πριν το ENDIF μπορεί να παραληφθεί αν δεν ενδιαφέρει το κομμάτι αυτό. Η ύπαρξη του ENDIF είναι υποχρεωτική εφ' όσον η έναρξη του IF ακολουθείται από το THEN. Σε οποιοδήποτε σημείο του BLOCK IF μπορεί να δοθεί εντολή εξόδου από την BLOCK IF με εντολή GOTO. Δεν επιτρέπεται όμως να μεταφερθεί η ροή του προγράμματος μέσα σε άλλο BLOCK IF. Δίνονται παρακάτω δύο παραδείγματα σωστής και λανθασμένης χρήσης αντίστοιχα.

Παράδειγμα σωστής διατύπωσης

```
      PROGRAM CORBI
      X=2.
      Y=3.
      IF (ΥΠΟΘΕΣΗ 1) THEN
      .
      .
      IF (ΥΠΟΘΕΣΗ 2) THEN
      .
      .
      ELSE ( ανήκει στην υπόθεση 2)
      GO TO 10
      ENDIF (τέλος της υπόθεσης 2)
```

```
      .
      ELSE (ανήκει στην υπόθεση 1)
      .
      ENDIF (τέλος της υπόθεσης 1)
10    CONTINUE
      .
      STOP
      END
```

Παράδειγμα λανθασμένης διατύπωσης

```
      PROGRAM WRONIF
      X=1.
      Y=2.
      IF (ΥΠΟΘΕΣΗ 1) GO TO 10
      IF (ΥΠΟΘΕΣΗ 2) THEN
      .
      ENDIF (τέλος της υπόθεσης 2)
      .
      IF(ΥΠΟΘΕΣΗ3) THEN
      .
10    CONTINUE
      ELSE (ανήκει στην υπόθεση 3)
      .
      ENDIF (τέλος της υπόθεσης 3)
      .
      STOP
      END
```

Στο παράδειγμα αυτό η τοποθέτηση των BLOCK IF είναι σωστή. Το λάθος βρίσκεται στο πρώτο IF (ΥΠΟΘΕΣΗ 1) GO TO 10 το οποίο μεταφέρει την ροή του προγράμματος μέσα σε ένα άλλο BLOCK IF, ενώ τέτοια διαδικασία δεν είναι επιτρεπτή.

3.4 Η ΕΝΤΟΛΗ PAUSE

Η γενική μορφή της ενολής PAUSE είναι:

PAUSE ή

PAUSE n ή

PAUSE ' MESSAGE .....'

όπου n είναι ένας αριθμός με όχι περισσότερα απο πέντε ψηφία , ο οποίος μπορεί να δηλώνει κάτι στον προγραμματιστή, και ' MESSAGE .....' ένα μήνυμα το οποίο ενημερώνει τον προγραμματιστή σχετικά με το που βρίσκεται η εκτέλεση του προγράμματος. Με την εκτέλεση της εντολής PAUSE, η ροή του προγράμματος διακόπτεται και ο χρήστης θα πρέπει να ακολουθήσει τις οδηγίες που εμφανίζονται στην οθόνη του τερματικού για να συνεχιστεί η ροή του προγράμματος. Μπορούν δε να υπάρχουν πολλές εντολές PAUSE σε ένα πρόγραμμα.

3.5 Η ΕΝΤΟΛΗ STOP

Η εντολή STOP συντάσσεται ως εξής:

STOP

STOP n

όπου n ένας αριθμός με όχι περισσότερα απο πέντε ψηφία ή ενα μήνυμα. Όπως έχει αναφερθεί ήδη μπορούν να υπάρχουν πολλές εντολές STOP σε ένα πρόγραμμα. Απο την στιγμή που θα εκτελεστεί ένα απο όλα, τότε η ροή του προγράμματος σταματά αυτόματα. Στην περίπτωση που κάθε εντολή STOP συνοδεύεται απο έναν διαφορετικό αριθμό, είναι εύκολο για τον προγραμματιστή να ξέρει σε ποιό σημείο σταμάτησε η εκτέλεση του προγράμματος.

4. ΜΕΡΙΚΕΣ ΕΣΩΤΕΡΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Μερικές από τις πιο συνηθισμένες μαθηματικές εκφράσεις είναι ενσωματωμένες στην FORTRAN και ο προγραμματιστής μπορεί να τις επικαλεστεί άμεσα. Σε επόμενο κεφάλαιο θα αναφερθεί πώς μπορεί ο καθένας να ορίσει συναρτήσεις (FUNCTIONS) ή υποπρογράμματα (SUBPROUTINES). Οι περισσότερες από τις μαθηματικές συναρτήσεις που είναι ενσωματωμένες στην FORTRAN απαιτούν κατά την κλίση τους τη γνώση της τιμής μιας μεταβλητής και επιστρέφουν την τιμή της συνάρτησης. Πριν από τη χρησιμοποίηση τέτοιων εκφράσεων συνιστάται η ενημέρωση απο το αντίστοιχο βιβλίο οδηγιών της FORTRAN που πρόκειται να χρησιμοποιηθεί. Οι πιό συνηθισμένες εσωτερικές συναρτήσεις (INTRINSIC FUNCTIONS) της FORTRAN φαίνονται στον παρακάτω πίνακα .

SQRT(X)	Τετραγωνική ρίζα
EXP(X)	Δύναμη του e (e^X)
COS(X)	Συνημίτονο X
SIN(X)	Ημίτονο X
TAN(X)	Εφαπτομένη X
ASIN(X)	Τόξο ημιτόνου X
ACOS(X)	Τόξο συνημιτόνου X
ATAN(X)	Τόξο Εφαπτομένης X (arctan(x))
ALOG(X)	Νεπέριος (Φυσικός) λογάριθμος
ALOG10(X)	Δεκαδικός λογάριθμος
ABS(X)	Απόλυτος τιμή
INT(X)	Ακέραιος μέρος(Το δεκαδικό μέρος μηδενίζεται)
NINT(X)	Πλησιέστερος ακέραιος

Προσοχή χρειάζεται κατα την χρήση τριγωνομετρικών συναρτήσεων. διότι οι γωνίες πρέπει να εκφράζονται σε ακτίνια.

Παράδειγμα

```
PROGRAM TESTF
X=6.75
A1=SQRT(X)
E1=EXP(X)
S1=SIN(X)
T1=TAN(X)
I1=INT(X)
```

```
I2=NINT(X)
WRITE(*,*)X,A1,E1,S1,T1,I1,I2
STOP
END
```

Τα αποτελέσματα που θα εμφανιστούν στην οθόνη είναι

6.750000 2.598076 854.0588 0.4500441 0.5039651 6 7

Προσέξτε το διαφορετικό αποτέλεσμα των INT(X) που είναι 6 και NINT(X) που είναι 7 παρ'όλο που προέρχονται απο τον ίδιο αριθμό. Υπενθυμίζεται οτι κατα τον υπολογισμό των τριγωνομετρικών εκφράσεων το X είναι εκφρασμένο σε ακτίνια και όχι σε μοίρες. Τονίζεται ακόμη οτι η ποσότητα της οποίας ζητείται ο υπολογισμός της τετραγωνικής ρίζας, ημιτόνου, κλπ., πρέπει να είναι πραγματική μεταβλητή (πραγματικός αριθμός) και όχι ακέραια μεταβλητή.

Γενικά το πλήθος των εσωτερικών συναρτήσεων και ο τρόπος κλήσεως διαφέρει από υπολογιστή σε υπολογιστή και από την μία έκδοση της FORTRAN σε άλλη. Γι' αυτό συνιστάται στους χρήστες η προηγούμενη ενημέρωσή τους στο σχετικό θέμα από το βιβλίο οδηγιών της FORTRAN την οποία πρόκειται να χρησιμοποιήσουν.

5. ΠΡΑΞΕΙΣ ΜΕΤΑΞΥ ΑΚΕΡΑΙΩΝ ΚΑΙ ΠΡΑΓΜΑΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ ΚΑΙ ΣΥΝΑΡΤΗΣΕΩΝ

Όπως αναφέρθηκε ήδη, η FORTRAN διαχωρίζει τους πραγματικούς από τους ακέραιους, παρ' όλο που οι δεύτεροι αποτελούν υποσύνολο των πρώτων. Ο διαχωρισμός αυτός γίνεται ανάλογα με τον τρόπο με τον οποίο αποθηκεύονται και επεξεργάζονται οι αριθμοί αυτοί στην μνήμη του Η/Υ. Αν και η χρησιμότητα του διαχωρισμού αυτού δεν είναι άμεσα προφανής, η πράξη δείχνει ότι τα πλεονεκτήματα είναι πάρα πολλά. Στις επόμενες παραγράφους παρουσιάζονται οι τρείς δυνατοί συνδυασμοί μεταξύ ακεραίων και πραγματικών.

5.1 ΠΡΑΞΕΙΣ ΜΕΤΑΞΥ ΑΚΕΡΑΙΩΝ

Μια ακέραια σταθερά είναι ένας ακέραιος αριθμός όπως π.χ. 1523, 627, 0, -132, -15327 κλπ. και το κοινό χαρακτηριστικό τους είναι ότι απουσιάζει η δεκαδική τελεία. Η αναγραφή του προσήμου είναι υποχρεωτική, εφ' όσον ο αριθμός είναι αρνητικός, ενώ είναι προαιρετική για τους θετικούς αριθμούς. Κάθε υπολογιστής μπορεί να αποθηκεύσει αριθμούς με κάποια συγκεκριμένη ακρίβεια (πλήθος σημαντικών ψηφίων), που ποικίλει από υπολογιστή σε υπολογιστή και καλό είναι πριν από την χρησιμοποίηση κάποιας εφαρμογής (προγράμματος) να διερευνώνται τα όρια αυτά, ανατρέχοντας στα σχετικά εγχειρίδια.

Οι μεταβλητές στις οποίες μπορεί να αντιστοιχίσει κανείς μια ακέραια σταθερά ονομάζονται ακέραιες μεταβλητές (INTEGER VARIABLES) και είναι αυτές στις οποίες το ονομά τους αρχίζει με ένα από τα γράμματα I, J, K, L, M, N καθώς επίσης και όσες αναφέρονται σε εντολή καθορισμού INTEGER. Η γενική μορφή καθορισμού της εντολής αυτής είναι:

INTEGER VAR1, VAR2

όπου VAR1, VAR2 είναι ονόματα μεταβλητών και παρόλο που δέν αρχίζει το όνομά τους με ένα απο τα γράμματα I, J, K, L, M, N, με την εντολή INTEGER δηλώνονται ως ακέραιοι. Η εντολή καθορισμού είναι μή εκτελέσιμη και γι'αυτό τοποθετείται πριν από οποιαδήποτε εκτελέσιμη εντολή προγράμματος και κάτω από την εντολή PROGRAM PNAME.

Ακέραια έκφραση (integer value expression) ονομάζεται κάθε έκφραση η οποία όταν υπολογιστεί καταλήγει σε μία ακέραια ποσότητα. Ο χρήστης πρέπει να είναι ιδιαίτερα προσεκτικός όσον αφορά στις πράξεις ακεραίων διότι το αποτέλεσμα συνήθως διαφέρει από το αλγεβρικό π.χ. η πράξη μεταξύ ακεραίων

10/3 θα δώσει αποτέλεσμα 3 και όχι 3.3333 και η διαίρεση μεταξύ ακεραίων 2/3 θα δώσει αποτέλεσμα 0.

Για ακέραιες μεταβλητές η γενική μορφή καθορισμού τιμής είναι

Ακέραιου τύπου μεταβλητή = ακέραια έκφραση

π.χ.

```
I32=K*K-4
KAPPA3= KI2+IB
```

Στα παραδείγματα αυτά, υπολογίζεται η τιμή της έκφρασης που βρίσκεται δεξιά του συμβόλου της ισότητας και κατόπιν ορίζεται ότι η τιμή αυτή είναι η τιμή της μεταβλητής.

Παράδειγμα

```
      PROGRAM INTEX
      I=2
      J=3
      K=4
      I1=I/J*K
      I2=K*I/J
      WRITE (*,*)I,J,K,I1,I2
      STOP
      END
```

Το παράδειγμα αυτό δείχνει την ιδιαιτερότητα των πράξεων μεταξύ ακεραίων καθώς και την προτεραιότητα των πράξεων. Εφ'όσον δεν έχει οριστεί διαφορετικά και πρόκειται για ισοδύναμες πράξεις, η σειρά εκτέλεσης των πράξεων διενεργείται απο αριστερά προς τα δεξιά. Στον υπολογισμό της τιμής του I1 το αποτέλεσμα θα είναι 0 διότι πρώτα θα γίνει η διαίρεση I/J (=2/3) η οποία θα δώσει αποτέλεσμα 0 και κατόπιν ο πολλαπλασιασμός με το K (=4), που δίνει τελικό αποτέλεσμα 0. Στον υπολογισμό του I2 το αποτέλεσμα θα είναι 2 διότι γίνεται πρώτα ο πολλαπλασιασμός K*I (4*2=8) και κατόπιν το αποτέλεσμα διαιρείται με το J (=3) που δίνει (8/3=2).

Παράδειγμα

```
      PROGRAM PRIMEN
      I=2
      J=256
      ID=2
      K=I
    1 CONTINUE
```

```
      IF(K/ID*ID.EQ.K)GOTO 2
      ID=ID+1
      GOTO 1
2     CONTINUE
      IF (ID.EQ.K)GOTO 3
4     CONTINUE
      K=K+1
      ID=2
      IF (K.GT.J) GOTO 5
      GOTO 1
3     WRITE (*,*)K
      GOTO 4
5     STOP
      END
```

Το πρόγραμμα αυτό βρίσκει τους πρώτους αριθμούς από το 2 μέχρι και το 256 και εκμεταλλεύεται την ιδιαιτερότητα των πράξεων μεταξύ ακεραίων. Αυτό που πρέπει να προσεχθεί ιδιαίτερα στο παράδειγμα είναι η εντολή

IF(K/ID*ID.EQ.K) GOTO 2

όπου η έκφραση K/ID*ID είναι ίση με K μόνον όταν η αλγεβρική διαίρεση είναι ακριβής (δηλ. το υπόλοιπο είναι μηδέν). Στην περίπτωση αυτή μένει να ελεγχθεί ποιός είναι ο διαιρέτης. Αν αυτός δεν είναι ο K, τότε ο K δεν είναι πρώτος αριθμός και ο έλεγχος προχωρά στον επόμενο αριθμό. Αν ο διαιρέτης είναι ο K τότε αυτό σημαίνει ότι όλοι οι αριθμοί από 2 έως K-1 δεν διαιρούν ακριβώς τον αριθμό αυτό, το οποίο σημαίνει ότι ο K είναι πρώτος.

Σε κάθε έκδοση της γλώσσας FORTRAN υπάρχουν μερικές ακέραιες εσωτερικές συναρτήσεις, τις οποίες μπορεί να χρησιμοποιήσει άμεσα ο προγραμματιστής. Μια χρήσιμη συνάρτηση είναι η IABS η οποία υπολογίζει την απόλυτη τιμή μιας ακέραιας έκφρασης. Μια πολύ χρήσιμη συνάρτηση είναι η MOD(I,J) η οποία υπολογίζει το υπόλοιπο τις διαίρεσης I/J. Αν I=K*J, τότε MOD(I,J)=0 σε κάθε άλλη περίπτωση MOD(I,J) ≠ 0.

5.2 ΠΡΑΞΕΙΣ ΜΕΤΑΞΥ ΠΡΑΓΜΑΤΙΚΩΝ

Μια πραγματική σταθερά είναι ένας αριθμός που περιέχει τη δεκαδική τελεία. Οι παρακάτω εκφράσεις είναι πραγματικές σταθερές.

2.35	0.132E+04	(=0.132×10^{4})
-127.2	0.327E-12	(=0.327×10^{-12})
3001.0	0.998E5	(=0.998×10^{5})
627.	-3.127E-05	(=-3.127×10^{-5})

Όπως φαίνεται απο τα παραπάνω, κάθε πραγματική σταθερά μπορεί να έχει τη γενική μορφή nnn.dd ή .dd ή n. ή .nE±ee ή n.E±ee ή n.nE±ee όπου E±ee δηλώνει τον εκθέτη δύναμης του 10.

Όλοι οι πραγματικοί αριθμοί πρέπει να περιλαμβάνουν την δεκαδική τελεία μία και μόνον φορά. Στην περίπτωση που η αναγραφή τους γίνεται με χρήση του εκθετικού (E±ee), αυτό που πρέπει να τονιστεί είναι ότι μετά από το γράμμα E ακολουθεί η δύναμη του 10, όπου η αναγραφή του προσήμου είναι υποχρεωτική εφ' όσον ο εκθέτης είναι αρνητικός αριθμός και προαιρετική αν είναι θετικός. Ο εκθέτης ανήκει στις ακέραιες σταθερές και δεν επιτρέπεται η αναγραφή της δεκαδικής τελείας. Το σύμβολο E μπορεί να χρησιμοποιηθεί μόνον με πραγματικούς αριθμούς και χρησιμοποιείται για την αναγραφή πολύ μεγάλων ή πολύ μικρών αριθμών όπως π.χ. ο αριθμός 6.023×10^{23}, που θα έπρεπε διαφορετικά να γραφεί ως 602,300,000,000,000,000,000,000. ή ο αριθμός 3.5×10^{-15} που θα έπρεπε να γραφεί ώς 0.0000000000000035. Όπως αναφέρθηκε προηγουμένως, κάθε υπολογιστής μπορεί να δεχθεί αριθμούς μέχρι έναν ορισμένο αριθμό σημαντικών ψηφίων. Αν δοθεί ένας αριθμός με περισσότερα ψηφία, τότε ο υπολογιστής κατ' αρχήν ελέγχει αν τα επιπλέον ψηφία στο συμβολισμό nnn.dd βρίσκονται δεξιά ή αριστερά από την δεκαδική τελεία. Στην πρώτη περίπτωση απλώς θα αγνοήσει τα επιπλέον ψηφία. Στην δεύτερη, θα καταμετρήσει πόσα ψηφία είναι αριστερά από την τελεία και κατόπιν θα την μεταφέρει προς τα αριστερά κατά τόσες θέσεις, ώστε η αναπαράσταση που θα προκύψει να είναι μέσα στην επιτρεπτή ακρίβεια.

Παραδείγματα

Στα παρακάτω παραδείγματα φαίνονται μερικά από τα συνηθισμένα λάθη που συμβαίνουν στη χρήση πραγματικών σταθερών.

3.25E8.	(Δεν επιτρέπεται η δεκαδική τελεία στο εκθετικό)
-4725	(Δεν έχει δεκαδική τελεία)
374,525.3	(Δεν επιτρέπεται η χρήση του κόμμα)
127.E540	(Το εκθετικό είναι έξω από την περιοχή των πραγματικών που μπορεί να δεχθεί ένα υπολογιστής)

Στις περιπτώσεις όπου ο χρήστης απαιτεί μεγαλύτερη ακρίβεια στα αποτελέσματα, μπορεί να χρησιμοποιήσει την έκφραση διπλής ακρίβειας. Να δηλωθεί δηλαδή στον υπολογιστή ότι ο χρήστης επιθυμεί ο αριθμός αυτός να

υπολογιστεί και αναγραφεί με διπλή ακρίβεια. Ο τρόπος αναγραφής είναι ο ίδιος με τον τρόπο αναγραφής των πραγματικών αριθμών n.nE±ee όπου το E αντικαθίσταται με το γράμμα D δηλ. n.nD±ee.

Με τον όρο πραγματική μεταβλητή εννοείται μια ποσότητα, η τιμή της οποίας μπορεί να μεταβάλεται κατά τη διάρκεια εκτέλεσης του προγράμματος. Κάθε πραγματική μεταβλητή παριστάνεται με ένα όνομα που αποτελείται από 1 έως 6 αλφαριθμητικούς χαρακτήρες με τους περιορισμούς ότι: (i) ο πρώτος χαρακτήρας δεν είναι αριθμός (ii) δεν περιέχεται κάποιος από τους ειδικούς χαρακτήρες και (iii) ο πρώτος χαρακτήρας δεν είναι κάποιος από τους I, J, K, L, M, N. Πραγματική μεταβλητή μπορεί να είναι κάποια που το όνομά της αρχίζει με έναν από τα I, J, K, L, M, N αν στην αρχή του προγράμματος δηλωθεί με την εντολή καθορισμού

REAL IGOR,JTOT

οπότε σε όλο το πρόγραμμα οι μεταβλητές IGOR και JTOT, παρ' όλο που αρχίζουν με I και J αντίστοιχα, θα είναι πραγματικές μεταβλητές.

Παραδείγματα

ALPHA	Πραγματική μεταβλητή
VAR1	Πραγματική μεταβλητή
VAR2	Πραγματική μεταβλητή
2BA	Λάθος ο πρώτος χαρακτήρας είναι αριθμός
.W3	Λάθος ο πρώτος χαρακτήρας είναι η τελεία
XY.	Λάθος περιέχεται η τελεία
COP	Πραγματική μεταβλητή

Κάθε έκφραση η οποία όταν υπολογιστεί δίνει έναν πραγματικό αριθμό ως αποτέλεσμα, ονομάζεται πραγματική έκφραση. Για παράδειγμα οι παρακάτω εκφράσεις είναι πραγματικές.

(i) X+10.7	(ii) (X*Y)/3	(iii) A**B
(iv) A**B**C	(v) (A-1.3)*(B+4.7)	(vi) SQRT((A*B)**C)

Προσοχή ιδιαίτερη χρειάζεται στην περίπτωση (iv) όπου διάφοροι μεταφραστές (compilers) εκτελούν την πράξη διαφορετικά. Το αποτέλεσμα θα είναι διαφορετικό αν η πράξη γίνει από αριστερά προς τα δεξιά A^{B^C} από το αν γίνει από δεξιά προς τα αριστερά, όπου πρώτα γίνεται ο υπολογισμός του D=B**C και κατόπιν το A^D. Για την αποφυγή λαθών είναι προτιμότερο να χρησιμοποιούνται παρενθέσεις με τρόπο που να καθορίζουν την σειρά των πράξεων. Στο παράδειγμα (vi) πρώτα γίνεται ο υπολογισμός της ποσότητας (A*B) και κατόπιν η ύψωση στη δύναμη C πριν τον υπολογισμό της τετραγωνικής ρίζας της ποσότητας. Θα πρέπει να

σημειωθεί ακόμη ότι απαγορεύεται η ύψωση του μηδέν σε μηδενική δύναμη και η χρήση δύο διαδοχικών πράξεων (π.χ. A*-B). Επίσης πρέπει να δίνεται προσοχή στη χρήση των παρενθέσεων, όπου κάθε αριστερή παρένθεση πρέπει να έχει την αντίστοιχη δεξιά.

Ασκήσεις

1. Γράψτε ένα πρόγραμμα FORTRAN το οποίο αφού διαβάσει τις τιμές τριών μεταβλητών A, B και C να υπολογίζει το A+B+C αν A*B*C>0 ή το $A^2+B^2+C^2$ αν A*B*C<0.
2. Γράψτε ένα πρόγραμμα FORTRAN το οποίο να υπολογίζει το άθροισμα των άρτιων αριθμών που περιέχονται μεταξύ των αριθμών 25 και 105.
3. Γράψτε ένα πρόγραμμα FORTRAN το οποίο αφού διαβάσει 5 ζεύγη αριθμών, να υπολογίζει την απόλυτη τιμή της διαφοράς των στην περίπτωση που ο ένας εκ των ζευγών είναι αρνητικός.
4. Γράψτε ένα πρόγραμμα FORTRAN το οποίο να υπολογίζει τις τιμές της συνάρτησης

$$Y=(X^2+X+3)/(X-2)$$

όταν το X παίρνει ακέραιες τιμές απο 3 έως και 10.
5. Μετατρέψτε το παραπάνω πρόγραμμα ώστε να υπολογίζει την ίδια συνάρτηση αλλά για τιμές του X απο 0 έως και 10 σε βήματα ΔX=1. Προσέξτε οτι ο παρονομαστής μηδενίζεται για X=2.

5.3 ΕΚΦΡΑΣΕΙΣ ΜΙΚΤΟΥ ΤΥΠΟΥ

Στις πράξεις που γίνονται μεταξύ δύο μεταβλητών όπου η μία είναι ακέραιου και η άλλη πραγματικού τύπου, το αποτέλεσμα είναι πραγματικός αριθμός. Αν σε μια έκφραση υπάρχουν πράξεις μεταξύ πραγματικών και ακεραίων, τότε κατά τον υπολογισμό της έκφρασης γίνεται μετατροπή τύπου. Πριν από την εκτέλεση της πράξης μεταξύ μεταβλητών μικτού τύπου, οι ακέραιοι μετατρέπονται σε πραγματικούς (REAL) που είναι και η πλέον ισχυρή έκφραση όπως φαίνεται και στον παρακάτω πίνακα.

ΤΥΠΟΣ ΤΟΥ Α	ΤΥΠΟΣ ΤΟΥ Β	ΤΥΠΟΣ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ
Α	**Β**	**Α+Β ή Α-Β ή Α*Β ή Α/Β**
INTEGER	INTEGER	INTEGER
INTEGER	REAL	REAL
REAL	INTEGER	REAL
REAL	REAL	REAL

΄Ετσι λοιπόν κατά την εκτέλεση της πράξης X*I, που είναι πολλαπλασιασμός της πραγματικής μεταβλητής X με την ακέραια μεταβλητή I, γίνεται μετατροπή της ακέραιας I σε πραγματική και κατόπιν ακολουθεί η πράξη του πολλαπλασιασμού. Το αποτέλεσμα βέβαια είναι πραγματικός αριθμός.

Θα πρέπει να αναφερθεί ότι ακολουθούνται οι συνηθισμένοι κανόνες προτεραιότητας κατά την εκτέλεση των πράξεων. 'Ετσι χρειάζεται προσοχή στη χρήση των τελεστών και εξασφάλιση της επιθυμητής, από το χρήστη, σειράς εκτέλεσης των πράξεων με την χρησιμοποίηση παρενθέσεων. Ως παράδειγμα αναφέρεται: 3/4*2.8. Εφ' όσον δεν έχει οριστεί διαφορετικά, ακολουθείται η εξής σειρά. Πρώτα γίνεται η διαίρεση του 3 δια 4. Επειδή οι αριθμοί είναι ακέραιοι, το αποτέλεσμα είναι 0 (ακέραιος). Κατόπιν το ακέραιο μηδέν μετατρέπεται σε πραγματικο 0. και ακολούθως γίνεται ο πολλαπλασιασμός με το 2.8 . Το αποτέλεσμα βέβαια είναι μηδέν (0.).

Όπως στην περίπτωση των τεσσάρων βασικών πράξεων, έτσι και στην ύψωση σε δύναμη ακολουθούνται οι ίδιοι κανόνες. Στον παρακάτω πίνακα φαίνονται οι τύποι βάσης και δύναμης που επιτρέπονται στη FORTRAN.

ΒΑΣΗ	ΔΥΝΑΜΗ	ΑΠΟΤΕΛΕΣΜΑ
X	A	X**A
INTEGER	INTEGER	INTEGER
REAL	INTEGER	REAL
INTEGER	REAL	REAL
REAL	REAL	REAL

Ζητώντας από τον Η/Υ να υπολογίσει το αποτέλεσμα της πράξης X**5 δεν γίνεται καμμία μετατροπή τύπου. Το X εξακολουθεί να είναι πραγματικός και η δύναμη 5 να είναι ακέραια. Αυτό που γίνεται στην πραγματικότητα είναι ότι δίνεται εντολή στον Η/Υ να πολλαπλασιάσει 5 φορές διαδοχικά την τιμή του X με τον εαυτό της. Έτσι λοιπόν όταν η δύναμη είναι ακέραιος αριθμός, δηλώνει τον αριθμό των διαδοχικών πολλαπλασιασμών που θα γίνουν.

Όταν η δύναμη δεν είναι ακέραιος τότε ο υπολογισμός γίνεται χρησιμοποιώντας την εσωτερική ενσωματωμένη ρουτίνα της FORTRAN για τον υπολογισμό του νεπέριου (φυσικού) λογάριθμου. Για τον υπολογισμό της ποσότητας X**A χρησιμοποιείται η σχέση $X^A = e^{A(\ln X)}$. Από τη σχέση αυτή είναι προφανές ότι δεν είναι δυνατή η ύψωση ενός αρνητικού πραγματικού αριθμού σε πραγματική δύναμη. Άλλωστε πως θα μπορούσε να οριστεί το πρόσημο του αποτελέσματος της πράξης -2.3**1.7. Έτσι πράξεις αυτού του τύπου δεν επιτρέπονται. Αν η βάση είναι ακέραιος και η δύναμη είναι πραγματικός, γίνεται μετατροπή της βάσης σε πραγματικό και στη συνέχεια υπολογίζεται η τιμή της έκφρασης που βέβαια θα είναι πραγματική. Γενικά, η χρήση παρενθέσεων σε περιπτώσεις διαδοχικών πράξεων δίνει την δυνατότητα ελέγχου της σειράς των πράξεων και την αποφυγή λαθών. Στις περιπτώσεις που η δύναμη είναι αρνητικός αριθμός, η χρησιμοποίηση παρενθέσεων είναι απαραίτητη, για την αποφυγή λάθους π.χ.

X**(-A) $\rightarrow (X^{-A})$

X**(Y**(Z**W))

X**(A-2.*I)

5.4 ΕΝΤΟΛΕΣ ΚΑΘΟΡΙΣΜΟΥ ΤΙΜΗΣ ΜΙΚΤΟΥ ΤΥΠΟΥ

Υπάρχουν δύο μορφές για την εντολή καθορισμού τιμής μικτού τύπου.

1. Πραγματική Μεταβλητή = Ακέραια Έκφραση
2. Ακέραια Μεταβλητή = Πραγματική Έκφραση

Στην πρώτη περίπτωση (π.χ. X=2*I+5) αφού γίνει ο υπολογισμός του δεύτερου μέλους της σχέσης, που είναι ακέραιος, στη συνέχεια μετατρέπεται σε πραγματική σταθερά.

Στη δεύτερη περίπτωση (π.χ. I=A*X+B) αφού γίνει ο υπολογισμός του δεύτερου μέλους, που είναι πραγματικός, στη συνέχεια μετατρέπεται σε ακέραιο, απορρίπτοντας το δεκαδικό μέρος, αν υπάρχει και οτιδήποτε και αν είναι. Π.χ η τιμή 12.2 δίνει την ακέραια τιμή 12 όπως και η 12.9 δίνει την ακέραια τιμή 12. Πολύ μεγάλοι πραγματικοί αριθμοί, π.χ. 3.5E40 δεν μπορούν να μετατραπούν σε ακέραιους. Πολλοί μεταφραστές της FORTRAN (Compilers) θα δώσουν διαγνωστικό μύνημα ότι η μετατροπή είναι λανθασμένη, υπάρχουν όμως μερικοί που δεν θα δώσουν διαγνωστικό μύνημα λάθους αλλά το αποτέλεσμα θα είναι λανθασμένο. Είναι ευθύνη του προγραμματιστή να χρησιμοποιήσει σωστά τις μετατροπές αυτού του τύπου.

5.5 ΜΕΡΙΚΕΣ ΕΝΣΩΜΑΤΩΜΕΝΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ΜΙΚΤΟΥ ΤΥΠΟΥ

Υπάρχουν μερικές ενσωματωμένες συναρτήσεις στην FORTRAN που αναλαμβάνουν να κάνουν μετατροπές του τύπου αυτού.

Η εντολή

IFIX(X+3.5*Y)

μετατρέπει το αποτέλεσμα των πράξεων που βρίσκονται στην παρένθεση σε ακέραια σταθερά. Η μετατροπή γίνεται κρατώντας μόνο το ακέραιο μέρος του αποτελέσματος, απορρίπτοντας το δεκαδικό μέρος, οποιοδήποτε και αν είναι.

Η εντολή

FLOAT(I*3-5)

κάνει την αντίστροφη διαδικασία. Υπολογίζει το αποτέλεσμα των πράξεων της παρένθεσης, που είναι ακέραιος και κατόπιν το μετατρέπει σε πραγματικό.

Η εντολή

INT(A+4.*B)

ενεργεί με τον ίδιο ακριβώς τρόπο όπως και η IFIX, κρατώντας δηλαδή μόνον το ακέραιο μέρος, απορρίπτοντας το δεκαδικό, όσο και αν είναι αυτό .

Η εντολή NINT(X-B**2) μετατρέπει επίσης την πραγματική τιμή της παρένθεσης σε ακέραιο. Στη μετατροπή όμως λαμβάνεται υπόψη και η τιμή του δεκαδικού μέρους. Αν το δεκαδικό μέρος είναι μεγαλύτερο από 0.5, τότε το ακέραιο μέρος αυξάνει κατά μία μονάδα και το αποτέλεσμα είναι το ακέραιο μέρος του πραγματικού αριθμού, αυξημένο κατά μία μονάδα. Αν το δεκαδικό μέρος είναι

μικρότερο από 0.5 τότε το αποτέλεσμα είναι ακριβώς το ακέραιο μέρος χωρίς καμμία αλλαγή. Στην περίπτωση που το δεκαδικό μέρος είναι ακριβώς 0.5 τότε το αποτέλεσμα εξαρτάται από τον compiler που χρησιμοποιείται. Μερικοί μεταφραστές της FORTRAN ακολουθούν τον ακόλουθο κανόνα, στην περίπτωση που το δεκαδικό μέρος είναι ακριβώς 0.5. Αν το τελευταίο ψηφίο του ακέραιου μέρους είναι άρτιος αριθμός, τότε το αποτέλεσμα είναι το ακέραιο μέρος της πραγματικής έκφρασης. Αν το τελευταίο ψηφίο του ακέραιου μέρους είναι περιττός αριθμός, τότε στο ακέραιο μέρος προστίθεται μία μονάδα ώστε το τελευταίο ψηφίο να είναι πάντοτε άρτιος αριθμός. Στο μεταφραστή FORTRAN του συστήματος EP/IX(UNIX), όταν το δεκαδικό μέρος είναι ακριβώς 0.5, η στρογγύλευση γίνεται προσθέτοντας μία μονάδα στον τελευταίο ακέραιο ψηφίο ανεξάρτητα αν προκύπτει περιττός ή άρτιος αριθμός.

6. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ- ΕΞΟΔΟΥ

Στο κεφάλειο αυτό αναφέρονται οι σχετικές εντολές οι οποίες επιτρέπουν τον καθορισμό των τιμών κάποιων μεταβλητών του προγράμματος κατά τη διάρκεια εκτέλεσης. Επίσης αναφέρονται οι σχετικές εντολές εκτύπωσης ή αποθήκευσης τιμών μεταβλητών και ο τρόπος δημιουργίας ενός αρχείου μέσα από πρόγραμμα.

6.1 Η ΕΝΤΟΛΗ OPEN

Κάθε διαδικασία ανάγνωσης ή καταχώρισης πληροφοριών μπορεί να γίνεται σε ένα αρχείο του Η/Υ. Το αρχείο αυτό για να είναι αναγνωρίσιμο, σαν οντότητα από τον Η/Υ, πρέπει να ορισθεί με βάση κάποια διαδικασία. Η εντολή OPEN ορίζει την διαδικασία που ακολουθείται για την δημιουργία ή την αναγνώριση της ύπαρξης ενός αρχείου. Η εντολή OPEN ακολουθείται από κάποιες παραμέτρους, οι οποίες δεν κρίνεται σκόπιμο να αναφερθούν όλες, αλλά αυτές οι οποίες είναι απαραίτητες και καλύπτουν τις περισσότερες ανάγκες, συντάσσεται δε ως ακολούθως

```
OPEN((UNIT=)un,FILE=fname,(ERR=itar))
```

Κάθε αρχείο, το οποίο για το χρήστη μπορεί να έχει οποιοδήποτε όνομα και να περιέχει διάφορες πληροφορίες, για τον υπολογιστή αναγνωρίζεται με έναν ακέραιο αριθμό, ο οποίος μπορεί να είναι μονοψήφιος ή διψήφιος. Ο αριθμός αυτός δηλώνεται από το un, ενώ το όνομα του αρχείου, το οποίο είναι κατανοητό για το χρήστη, δηλώνεται από το fname. Αν τυχόν συμβεί κάποιο λάθος, π.χ. το όνομα του αρχείου δεν υπάρχει, με την παράμετρο ERR=itar μεταφέρεται η ροή του προγράμματος στον αριθμό που δηλώνει το tar. Θα πρέπει να σημειωθεί ότι οι τιμές των un, fname και tar δεν θα πρέπει να αλλάζουν κατά τη διάρκεια εκτέλεσης ενός προγράμματος, δηλαδή δεν μπορεί να είναι μεταβλητές. Ο καθορισμός του αριθμού (un) και του ονόματος του αρχείου (fname) είναι υποχρεωτικός, ενώ μπορεί να παραληφθεί η καταχώρηση ERR.

```
OPEN(10,file='INDAT')
OPEN(99,file='OUTPUT')
OPEN(file='TEST')
```

Τα δύο πρώτα παραδείγματα αποτελούν σωστή διατύπωση της εντολής OPEN ενώ το τρίτο όχι, διότι δεν δηλώνεται ο αριθμός. Το όνομα του αρχείου (INDAT, OUTPUT κλπ) μπορεί να είναι γραμμένο με κεφαλαία ή πεζά γράμματα και θα πρέπει οπωσδήποτε να παρικλείεται σε αποστρόφους.

6.2 Η ΕΝΤΟΛΗ READ

Μέχρι τώρα στα παραδείγματα που παρατέθηκαν, ο καθορισμός της τιμής μεταβλητών γινόταν μέσα στο πρόγραμμα. Αυτό όμως δεν είναι η συνήθης πρακτική. Συνήθως ο προγραμματιστής γράφει έτσι το πρόγραμμα ώστε να μπορεί να αντιμετωπίσει διάφορες περιπτώσεις, χωρίς να χρειάζεται να ξαναγράψει το πρόγραμμα. Έτσι π.χ. για την εύρεση των ριζών μιας εξίσωσης δευτέρου βαθμού, το πρόγραμμα γράφεται έτσι ώστε να "διαβάζει" τις τιμές των συντελεστών, οι οποίες κάθε φορά μπορεί να είναι διαφορετικές. Η εντολή READ επιτρέπει αυτή τη διαδικασία και συντάσσεται όπως παρακάτω

READ((unit=)un,f,(ERR=itar),(END=itar1))VAR

Όπου unit=un είναι ένας μονοψήφιος ή διψήφιος αριθμός που χαρακτηρίζει το αρχείο από όπου πρόκειται να διαβαστούν στοιχεία και ο οποίος καθορίστηκε από την εντολή OPEN. Το f θα πρέπει να είναι ένας αριθμός ο οποίος θα καθορίζει τον τρόπο με τον οποίο θα γίνει η ανάγνωση (FORMAT) και προς το παρόν θα χρησιμοποιηθεί το σύμβολο * στην θέση του. Το ERR=itar δηλώνει ότι σε περίπτωση λάθους, η ροή του προγράμματος θα πρέπει να μεταφερθεί εκεί όπου δηλώνει ο αριθμός tar και το END=itar1 δηλώνει ότι αν η ανάγνωση φθάσει στο τέλος του αρχείου, η ροή του προγράμματος θα πρέπει να μεταφερθεί στον αριθμό που δηλώνει ο αριθμός itar1. Οι παράμετροι ERR και END δεν είναι υποχρεωτικό να αναγράφονται εφ'όσον δεν το επιθυμεί ο χρήστης, ενώ VAR είναι το όνομα της μεταβλητής (ή μεταβλητών) που θα αντιστοιχηθούν στην τιμή (ή τις τιμές). Στην περίπτωση όπου η ανάγνωση γίνεται από το τερματικό (όπου στην περίπτωση αυτή δεν χρειάζεται προηγουμένως να χρησιμοποιηθεί η εντολή OPEN), η εντολή READ συντάσσεται ως εξής

READ(*,*)VAR1

Το πρώτο * δηλώνει ότι η ανάγνωση θα γίνει από το τερματικό και το δεύτερο δηλώνει ότι ο αριθμός που θα διαβαστεί θα πρέπει να μετατραπεί στον

τύπο που δηλώνει ο όνομα της μεταβλητής VAR1 που ακολουθεί. Έτσι λοιπόν ο αριθμός που θα διαβαστεί από το τερματικό, θα είναι η τιμή της μεταβλητής VAR1.

Στην περίπτωση όμως που πρόκειται να δοθεί μια ακολουθία αριθμών, π.χ. η βαθμολογία φοιτητών σε κάποιο μάθημα με σκοπό την στατιστική επεξεργασία των, τότε η ανάγνωση της βαθμολογίας απο το τερματικό δέν είναι πρόσφορη διαδικασία, διότι δεν είναι εύκολη η διόρθωση μιας λανθασμένης πληκτρολόγησης ή ανάγνωσης. Πιό εύκολη διαδικασία είναι η δημιουργία ενός αρχείου με την βαθμολογία των φοιτητών, με την βοήθεια ενός συντάκτη κειμένου (editor) όπου υπάρχει η δυνατότητα εύκολης διόρθωσης, και κατόπιν η ανάγνωση των στοιχείων απο το πρόγραμμα.

Παράδειγμα

```
PROGRAM TRE
OPEN(10,FILE='INPUT')
READ(*,*)N
READ(10,*)X
READ(10,*)Y
WRITE(*,*)N,X,Y
STOP
END
```

Με την εντολή OPEN(10,FILE='INPUT') το αρχείο INPUT παίρνει τον αριθμό 10. Στην ακόλουθη εντολή READ(*,*)N ο Η/Υ περιμένει από το χρήση να γράψει στο τερματικό έναν αριθμό, ο οποίος θα είναι η τιμή της μεταβλητής N. Με τις εντολές READ(10,*)X και READ(10,*)Y ο Η/Υ διαβάζει από το αρχείο 10 (όνομα INPUT) δυο τιμές που τις αντιστοιχεί στις μεταβλητές X και Y. Το σύμβολο * δηλώνει στον υπολογιστή οτι θα διαβάσει μια τιμή απο το αρχείο 10, την οποία θα αναθέσει στην μεταβλητή X. Αν η τιμή που θα διαβάσει είναι ακέραιος αριθμός, τότε θα την μετατρέψει σε πραγματική, ώστε να ταιριάζει με τον τύπο του ονόματος της μεταβλητής στην οποία θα ανατεθεί η τιμή. Είναι προφανές ότι πριν από την εκτέλεση ενός τέτοιου προγράμματος πρέπει να έχει δημιουργηθεί το αρχείο INPUT στο οποίο να περιέχονται τα ανάλογα στοιχεία. Η σειρά των εντολών READ(10,*)X και READ(10,*)Y έχει διαφορετική λειτουργία από την εντολή READ(10,*)X,Y. Στην πρώτη περίπτωση που αναφέρεται στο παράδειγμα ο υπολογιστής διαβάζει από το αρχείο INPUT την τιμή της μεταβλητής X απο μια γραμμή και από την επόμενη γραμμή του αρχείου INPUT την τιμή της μεταβλητής Y. Δηλαδή διαβάζει μία τιμή από κάθε γραμμή, ανεξάρτητα από το αν ακολουθούν άλλες τιμές στην ίδια γραμμή ή όχι. Με την εντολή READ(10,*)X,Y ο Η/Υ

περιμένει να βρεί δύο τιμές στην ίδια γραμμή για να διαβάσει. Γενικά, τα ονόματα των μεταβλητών που ακολουθούν την εντολή READ μπορούν να είναι περισσότερα από ένα, χωρίς να υπάρχει ανώτατο όριο. Πρέπει όμως να δίνεται προσοχή και στον τρόπο με τον οποίο είναι γραμμένες οι τιμές των μεταβλητών στο αρχείο από όπου το πρόγραμμα θα διαβάσει τις τιμές.

Κρίνεται σκόπιμο στο σημείο αυτό, να μην αναφερθεί η διαδικασία ανάγνωσης με συγκεκριμένο τρόπο (FORMAT) η οποία όμως θα εξηγηθεί σε επόμενο κεφάλαιο.

6.3 Η ΕΝΤΟΛΗ ACCEPT

Η εντολή ACCEPT λειτουργεί κατα τον ίδιο τρόπο όπως και η READ μόνο που χρησιμοποιείται για είσοδο τιμών απο τερματικό. Η γενική μορφή της είναι

ACCEPT f,VAR ή ACCEPT *, VAR

όπου f είναι ένας αριθμός ο οποίος καθορίζει τον τρόπο (FORMAT) με τον οποίο θα γίνει η ανάγνωση της τιμής που θα ανατεθεί η μεταβλητή. Αν αντί του f χρησιμοποιηθεί το σύμβολο *, τότε δεν έχει σημασία αν η τιμή που θα γραφεί στο τερματικό είναι πραγματικού ή ακέραιου τύπου, καθόσον η ανάθεση της τιμής στην μεταβλητή θα καθοριστεί απο το όνομά της και μόνον.

6.4 Η ΕΝΤΟΛΗ WRITE

Η εντολή WRITE είναι παρόμοια με την εντολή READ μόνον που γράφει (τυπώνει) τις τιμές κάποιων μεταβλητών ή κάποιο μήνυμα. Η εντολή WRITE συντάσσεται ως εξής

WRITE((unit=un),f)VAR$_i$

Όπου unit=un είναι ένας αριθμός ο οποίος χαρακτηρίζει το αρχείο, όπου πρόκειται να γραφούν τα στοιχεία. Είναι προφανές ότι το αρχείο αυτό πρέπει να έχει δημιουργηθεί προηγούμενα με την εντολή OPEN. Το f είναι ένας αριθμός που χαρακτηρίζει τον τρόπο με τον οποίο θα γραφούν τα αποτελέσματα στο αρχείο (FORMAT). Στην περίπτωση που η εκτύπωση πρόκειται να γίνει στην οθόνη του τερματικού, ο αριθμός un αντικαθίσταται από το σύμβολο *. Επίσης το f μπορεί να αντικατασταθεί από το * όταν αφήνεται ο Η/Υ να επιλέξει τον τρόπο εκτύπωσης των τιμών των παραμέτρων VAR$_i$, οι οποίες μπορεί να είναι περισσότερες από μία και ανάλογα με τον τύπο της κάθε μιάς. Αν το μήκος μιας γραμμής δεν επαρκεί για την εκτύπωση των τιμών όλων των μεταβλητών που ορίστηκαν να τυπωθούν, τότε

θα χρησιμοποιηθούν και επόμενες γραμμές. Το VARi είναι σύνολο ονομάτων μεταβλητών ή/και αριθμητικών εκφράσεων οι οποίες χωρίζονται μεταξύ τους με κόμμα π.χ. A, B, CAT, D, ICON, TER. Στην περίπτωση όπου επιθυμείται η αναγραφή και ενός μηνύματος, τότε το μήνυμα θα πρέπει να βρίσκεται μεταξύ αποστρόφων π.χ.

```
WRITE(*,*) THESE ARE THE VALUES'
```

Όταν η ροή του προγράμματος θα φθάσει στην εντολή αυτή, στην οθόνη του τερματικού θα εμφανιστεί το μύνημα :
THESE ARE THE VALUES
Είναι δυνατή η εκτύπωση μηνυμάτων και τιμών μεταβλητών συγχρόνως, όχι όμως και η εντολή αλλαγής γραμμής κατά την εκτύπωση. Π.χ. Έστω ότι οι ακόλουθες μεταβλητές έχουν τις τιμές I=3, J=7, X=32., Y=147.3 . Η εντολή

```
      WRITE(*,*)'FOR I= ',I,' AND  J=',J,/, THE VALUES',
     > ' OF X AND Y ARE: X=',X,' Y'=Y
```

θα προκαλέσει διαγνωστικό μήνυμα λάθους κατα την μετάφραση (compilation), διότι υπάρχει το σύμβολο / , το οποίο δηλώνει αλλαγή γραμμής.

6.5 Η ΕΝΤΟΛΗ REWIND

Με την εντολή OPEN ορίζεται ένα αρχείο σαν μονάδα εισόδου-εξόδου ενός προγράμματος και με τις εντολές READ και WRITE επιτυγχάνεται εγγραφή ή ανάγνωση στοιχείων από το αρχείο αυτό. Για την διαχείρηση των αρχείων σειρακής προσπέλασης, ο υπολογιστής χρησιμοποιεί έναν δείκτη στο κάθε αρχείο, με τον οποίο γνωρίζει ανά πάσα στιγμή σε ποιό σημείο του αρχείου βρίσκεται η διαδικασία εγγραφής ή ανάγνωσης. Κάθε φορά που εκτελείται μια εντολή READ ή WRITE, ο δείκτης προχωρά στην επόμενη εγγραφή.
Η εντολή

```
      REWIND un
```

μεταφέρει τον δείκτη του αρχείου, το οποίο έχει τον αριθμό un, στην αρχή του αρχείου. Με τον τρόπο αυτό είναι δυνατόν να διαβαστεί το περιεχόμενο ενός αρχείου περισσότερες από μία φορές.

6.6 Η ΕΝΤΟΛΗ BACKSPACE

Η εντολή συντάσσεται ως

BACKSPACE un

όπου το un είναι ένας αριθμός που χαρακτηρίζει το αρχείο το οποίο προηγούμενα έχει δημιουργηθεί με την εντολή OPEN και χρησιμοποιείται σε αρχείο σειρακής προσπέλασης.

Η δράση της εντολή BACKSPACE είναι παρόμοια με αυτήν της εντολής REWIND, μόνον που στην προκειμένη περίπτωση, ο δείκτης του αρχείου μεταφέρεται στην αμέσως προηγούμενη εγγραφή (γραμμή) και όχι στην αρχή του αρχείου.

6.7 Η ΕΝΤΟΛΗ ENDFILE

Η γενική της μορφή είναι

ENDFILE UN

όπου το un είναι ένας αριθμός που χαρακτηρίζει το αρχείο, το οποίο προηγούμενα έχει δημιουργηθεί με την εντολή OPEN και χρησιμοποιείται σε αρχεία σειριακής προσπέλασης.

Όταν εκτελείται η εντολή ENDFILE, γράφεται στο αρχειο un και στο σημείο που βρίσκεται αυτή τη στιγμή ο δείκτης του αρχείου, μια πληροφορία που δηλώνει ότι αυτό το σημείο είναι το τέλος του αρχείου. Κάθε προσπάθεια ανάγνωσης πληροφορίας μετά από το σημείο αυτό σε συγκεκριμένο αρχείο, θα αποτύχει.

6.8 Η ΕΝΤΟΛΗ CLOSE

Η εντολή CLOSE εκτελεί την αντίθετη ακριβώς διαδικασία από αυτήν που εκτελείται με την εντολή OPEN. Με άλλα λόγια μια εντολή CLOSE καταργεί μια προηγούμενη εντολή OPEN.

Η γενική της μορφή είναι

CLOSE(un)

όπου un είναι ένας αριθμός ο οποίος χαρακτηρίζει ένα αρχείο το οποίο έχει προηγούμενα δημιοργηθεί με την εντολή OPEN. Η εντολή CLOSE δεν είναι απαραίτητη σε ένα πρόγραμμα, διότι ο υπολογιστής όταν εκτελέσει μια εντολή STOP, τότε αυτόματα κλείνει όλες τις μονάδες και αρχεία τα οποία έχουν δημιουργηθεί με εντολή OPEN.

Παράδειγμα

Εύρεση ριζών δευτεροβάθμιας εξίσωσης της μορφής $AX^2+BX+C=0$.

Ζητείται να γραφεί ένα πρόγραμμα το οποίο:

(i) Θα διαβάζει τις τιμές τριών πραγματικών μεταβλητών A, B, C.

(ii) Θα υπολογίζει τις ρίζες της εξίσωσης

(iii) Θα τυπώνει τις τιμές των A, B, C και των ριζών.

Διαδικασία:

Για τον υπολογισμό των ριζών της εξίσωσης απαιτείται υπολογισμός του προσήμου της διακρίνουσας D=B**2-4.*A*C. Διακρίνονται οι παρακάτω περιπτώσεις.

Αν D≥0. τότε υπάρχουν δύο πραγματικές ρίζες.

Αν D<0. τότε υπάρχουν δύο μιγαδικές ρίζες.

Έτσι υπολογίζονται τέσσερις ποσότητες που θα είναι :

RR1: Πραγματικό μέρος της πρώτης ρίζας

RI1 : Φανταστικό μέρος της πρώτης ρίζας

RR2: Πραγματικό μέρος της δεύτερης ρίζας

RI2: Φανταστικό μέρος της δεύτερης ρίζας

Είναι προφανές ότι αν D≥ 0. τότε τα RI1 και RI2 θα έχουν μηδενική τιμή.

Το πρόγραμμα για την επίλυση είναι το παρακάτω:

```
      PROGRAM ROOTS
10    WRITE(*,*) 'ENTER VALUES FOR A,B,C'
      READ (*,*) A,B,C
      IF(A.EQ.0.) GO TO 10
      D=B**2-4.*A*C
      IF(D.GE.0.)THEN
      RR1=(-B-SQRT(D))/(2.*A)
      RI1=0.
      RR2=(-B+SQRT(D))/(2.*A)
      RI2=0.
      ELSE
      RR1=-B/(2.*A)
      RI1=-SQRT(-D)/(2.*A)
      RR2=RR1
      RI2=-RI1
```

```
      ENDIF
      WRITE(*,*)'THE ROOTS OF A*X**2+B*X+C=0.  WHEN'
      WRITE(*,*)'A=',A, 'B=',B, 'C=',C,'ARE:'
      WRITE(*,*) 'REAL ROOT1=',RR1, 'IMAG:PART=',RI1
      WRITE(*,*) 'REAL ROOT2=',RR2, 'IMAG.RART=',RI2
      STOP
      END
```

7. Η ΕΝΤΟΛΗ DO

Υπάρχουν πολλές περιπτώσεις όπου μια εντολή ή μια σειρά εντολών χρειάζεται να εκτελεστούν περισσότερες από μία φορές επαναληπτικά. Στις περιπτώσεις αυτές χρησιμοποιείται η εντολή DO η οποία σε συνδυασμό με τις μεταβλητές με δείκτες που θα αναφερθούν στο επόμενο κεφάλαιο, διευκολύνει τον προγραμματισμό και συγχρόνως αποφεύγονται λάθη.

Η εντολή DO συντάσσεται ως εξής:

DO NUM IVAR=LL,LR,IST

όπου το NUM είναι ένας ακέραιος αριθμός που μπορεί να έχει μέχρι 5 ψηφία και δηλώνει το σημείο όπου τελειώνει η εντολή DO. Το IVAR είναι το όνομα μιας μεταβλητής, η οποία συνήθως είναι ακέραιου τύπου, η τιμή της οποίας αλλάζει κάθε φορά που εκτελείται το περιεχόμενο της εντολής DO. Η αρχική τιμή της μεταβλητής IVAR είναι ίση με την τιμή LL. Το LL πρέπει να είναι ένας αριθμός ή μία μεταβλητή της οποίας η τιμή έχει οριστεί προηγούμενα μέσα στο πρόγραμμα. Το LR δηλώνει τη μέγιστη τιμή που θα πάρει η μεταβλητή IVAR και θα πρέπει να είναι ένας αριθμός ή μια μεταβλητή της οποίας η τιμή έχει οριστεί προηγούμενα μέστα στο πρόγραμμα. Το IST είναι ένας αριθμός που δηλώνει το βήμα με το οποίο θα αλλάξει η τιμή της μεταβλητής IVAR. Στην περίπτωση που το βήμα είναι ίσο με 1, τότε η τιμή του IST μπορεί να παραληφθεί. Αυτό που πρέπει να τονιστεί είναι ότι το LL δηλώνει τήν τιμή εκκίνησης της IVAR και το LR την τελική τιμή. Έτσι παραδεκτά όρια για τα LL και LR μπορούν να είναι τα παρακάτω ζεύγη τιμών 1,10 ή 20,100 ή -10,10 όπου βέβαια το βήμα είναι 1. Παραδεκτά επίσης είναι και τα 100, 10, -2 που σημαίνει ότι η πρώτη τιμή της IVAR είναι 100, η τελική 10 και το βήμα -2. Τα παρακάτω είναι λανθασμένα.

10, 3, 1 -10, 0, -1 1, 10, -1

Στις περιπτώσεις αυτές το DO LOOP θα εκτελεστεί μόνον μία φορά, για την αρχική τιμή του IVAR.

Ένα σημαντικό σημείο το οποίο πρέπει να τονιστεί είναι ότι οι τιμές των παραμέτρων της DO (δηλ. οι IVAR, LL, LR, IST) δεν πρέπει να αλλάζουν μέσα στο πεδίο ενεργειών της εντολής DO.

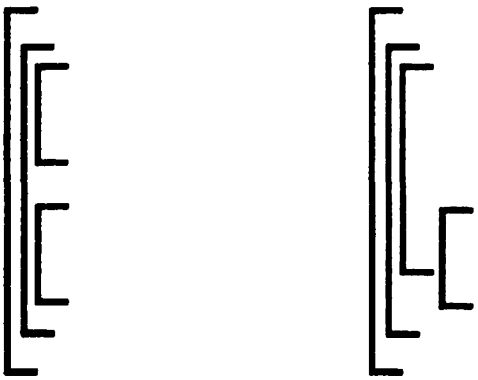
Το τέλος της ενέργειας της εντολής DO πρέπει οπωσδήποτε να είναι μια εκτελέσιμη εντολή και δεν πρέπει να είναι κάποια από τις STOP, PAUSE, RETURN, END, DIMENSION, GOTO ή IF. Για την αποφυγή λαθών συνιστάται η χρησιμοποίηση της εντολής CONTINUE ως τέλος της ενέργειας της εντολής DO.

Σε οποιοδήποτε σημείο μιας ανακύκλωσης DO, επιτρέπεται η μεταφορά της ροής του προγράμματος έξω από το πεδίο της DO με χρήση π.χ. της εντολής IF... GOTO. Σε μιά τέτοια περίπτωση η μεταβλητή ελέγχου (IVAR) έχει ορισμένη τιμή

και είναι αυτή που είχε την στιγμή που βγήκε απο την ανακύκλωση DO. Αν η ανακύκλωση DO ικανοποιηθεί πλήρως, στο επόμενο εκτελέσιμο βήμα η τιμή της μεταβλητής IVAR είναι αόριστη.

Η τιμή της μεταβλητής IVAR ορίζεται τη στιγμή που εκτελείται η εντολή DO. Έτσι δεν επιτρέπεται η είσοδος στο πεδίο ενεργειών μιας ανακύκλωσης DO, από άλλο σημείο του προγράμματος. Και τούτο διότι σε μια τέτοια περίπτωση δεν έχει οριστεί αρχική τιμή για τη μεταβλητή IVAR.

Μέσα σε μια ανακύκλωση DO μπορούν να υπάρχουν και άλλες ανακυκλώσεις DO. Πρέπει όμως η εσωτερική ανακύκλωση DO (ή αντίστοιχα οι εσωτερικές ανακυκλώσεις DO) να ικανοποιηθεί πριν από την ικανοποιήση της εξωτερικής ανακύκλωσης DO. Στο παρακάτω διάγραμμα φαίνονται οι επιτρεπτές και μη τοποθετήσεις ανακυκλώσεων DO.



Επιτρεπτή τοποθέτηση Μή επιτρεπτή τοποθέτηση

Στην FORTRAN V (ή FORTRAN 77) είναι δυνατόν οι τιμές των παραμέτρων της εντολής DO (δηλ. οι IVAR, LL, LR, IST) να είναι πραγματικοί. Έτσι η έκφραση

```
DO 10 X=0.2, 3.5, 0.1
```

είναι σωστή και αποδεκτή. Αυτή η δυνατότητα δίνει ένα σοβαρό πλεονέκτημα στις περιπτώσεις όπου χρειάζεται η μεταβολή των τιμών κατά δεκαδικά και όχι ακέραια μέρη, για τον υπολογισμό κάποιων ποσοτήτων.

Παράδειγμα
Δίνεται η έλλειψη

$$\frac{X^2}{5^2} + \frac{Y^2}{3^2} = 1$$

Να βρεθούν τα σημεία με ακέραιες τιμές των X και Y που βρίσκονται μέσα ή πάνω στην έλλειψη και να τυπώνονται τα αντίστοιχα σημεία. Θεωρείστε ότι το X παίρνει τιμές απο 1 έως 10 και το Y απο 2 έως 8.

Διαδικασία:

Το οτι ζητείται να βρεθούν τα ζεύγη τιμών τα οποία βρίσκονται μέσα ή πάνω στην έλλειψη σημαίνει οτι η τιμή του πρώτου μέλους πρέπει να είναι μικρότερη ή ίση με την τιμή του δεύτερου μέλους.

```
      PROGRAM TELLIP
      DO 10 I=1,10
      DO 15 J=2,8
      X=FLOAT(I)
      Y=FLOAT(J)
      F=((X*X)/(5.*5.))+((Y*Y)/(3.*3.))-1.
      IF(F.LE.0.)THEN
      WRITE(*,*)' X=',X,'  Y=',Y
      ENDIF
15    CONTINUE
10    CONTINUE
      STOP
      END
```

Οι εντολές

```
      DO 10 I=1,10
      DO 15 J=2,8
      X=FLOAT(I)
      Y=FLOAT(J)
```

θα μπορούσαν να αντικατασταθούν απο τις εντολές

```
      DO 10 X=1.,10.
      DO 15 Y=2.,8.
```

χωρίς να αλλάξει τίποτε άλλο στο πρόγραμμα και να προκύπτουν σωστά αποτελέσματα.

Η εντολή DO εμφανίζεται και με την ακόλουθη μορφή. η οποία θα πρέπει να αποφεύγεται απο μαθητευόμενους προγραμματιστές.

```
      DO IVAR=LL,LR,ST
```

και τελειώνει με την εντολή ENDDO. Δεν εμφανίζεται δηλαδή ο αριθμός που δηλώνει το τέλος της εντολής DO, αλλά τούτο δηλώνεται απο την εντολή ENDDO. Υπακούει στους ίδιους ακριβώς κανόνες όπως και η μορφή της εντολής DO που έχει αναφερθεί παραπάνω.

8. ΜΕΤΑΒΛΗΤΕΣ ΜΕ ΔΕΙΚΤΕΣ

Η γλώσσα FORTRAN έχει τη δυνατότητα να αποθηκεύει στο όνομα μιας μεταβλητής, η οποία έχει τη μορφή πίνακα, ένα μεγάλο πλήθος πληροφοριών. Έτσι η δομή του προγράμματος γίνεται πιο απλή, ενώ ένα μεγάλο πλήθος εντολών αντικαθίσταται από πολύ λίγες.

Σε πολλές μαθηματικές εκφράσεις συναντώνται διανύσματα, πίνακες, τελεστές κλπ. σε μια ή περισσότερες διαστάσεις, όπως επίσης και πράξεις μεταξύ αυτών. Τα πλεονεκτήματα της χρησιμοποίησης μεταβλητών με δείκτες καθώς και η απλότητα των προγραμμάτων θα γίνει περισσότερο κατανοητή με τα επόμενα.

Ένας πίνακας Α διάστασης 3x3, παριστάνεται στην άλγεβρα με τον ακόλουθο τρόπο

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

Για την καταχώρηση των στοιχείων του πίνακα αυτού σε ένα πρόγραμμα χρειάζεται να δηλωθούν τα ονόματα 9 μεταβλητών. Με την δυνατότητα που προσφέρεται απο την FORTRAN , ο πίνακας αυτός μπορεί να παρασταθεί με τη χρήση του ονόματος μιας μεταβλητής Α, η οποία θα έχει δύο διαστάσεις και συμβολίζεται σαν A(3,3). Η δήλωση αυτή σημαίνει ότι η μεταβλητή Α είναι ένας πίνακας με στοιχεία 3x3 (συνολικά 9 στοιχεία), τα οποία είναι τα:

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)

και όπου ο πρώτος δείκτης δηλώνει τη γραμμή και ο δεύτερος τη στήλη. Είναι προφανές ότι ένας πίνακας μπορεί να έχει μεγαλύτερη διάσταση nxn, όπου η μέγιστη διάσταση του πίνακα περιορίζεται μόνον από τις δυνατότητες του υπολογιστή και την έκδοση της γλώσσας FORTRAN που χρησιμοποιούνται. Επίσης ένας πίνακας μπορεί να έχει διάσταση (n*n*n), όπου η μέγιστη τιμή του γινομένου (n*n*n) εξαρτάται απο την διαθέσιμη μνήμη του υπολογιστή, ενώ δέν μπορεί να δηλωθεί πίνακας (n*n*n*n).

Τα στοιχεία ενός πίνακα μπορεί να είναι ακέραιοι ή πραγματικοί, ανάλογα με τον πρώτο χαρακτήρα της δήλωσης του συμβολισμού του πίνακα. Ισχύουν οι ίδιοι κανόνες όπως και στις μεταβλητές, δηλ. αν ο πρώτος χαρακτήρας είναι ένα από τα γράμματα I, J, K, L, M, N τότε τα στοιχεία του πίνακα θα είναι ακέραιοι. Διαφορετικά, τα στοιχεία του πίνακα είναι πραγματικοί. Είναι προφανές ότι δεν μπορούν να υπάρχουν στον ίδιο πίνακα μερικά στοιχεία τα οποία είναι ακέραιοι

και τα υπόλοιπα πραγματικοί. Από τη δήλωση του ονόματος του πίνακα καθορίζεται για όλα τα στοιχεία αν θα είναι ακέραιοι ή πραγματικοί.

Για να γνωρίζει ο υπολογιστής ότι το όνομα μιας μεταβλητής με δείκτες δεν είναι μια απλή μεταβλητή, πρέπει τούτο να δηλωθεί στην αρχή του προγράμματος. Ο συνηθέστερος τρόπος δήλωσης είναι με τη χρήση της εντολής DIMENSION την οποία ακολουθεί το όνομα της μεταβλητής και σε παρένθεση ο αριθμός των στοιχείων της.

Η εντολή DIMENSION A(100) μπαίνει ακριβώς κάτω απο την εντολή PROGRAM PNAME. δηλαδή πρίν απο οποιαδήποτε εκτελέσιμη εντολή του προγράμματος. Η εντολή

DIMENSION A(50)

δηλώνει στον υπολογιστή ότι η μεταβλητή Α περιλαμβάνει 50 στοιχεία, τα οποία θα συμβολίζονται ως A(1), A(2),....A(50). Με αυτό τον τρόπο δήλωσης, η αρίθμηση των στοιχείων αρχίζει από το 1. Σε ειδικές περιπτώσεις, αυτού του είδους η δήλωση μπορεί να μήν ικανοποιεί. Αν επιθυμείται η αρίθμηση να γίνεται από το στοιχείο 0 (μηδέν) τότε η δήλωση πρέπει να γίνει ως εξής

DIMENSION A(0:50).

Στην περίπτωση αυτή ο πίνακας Α περιλαμβάνει 51 στοιχεία. Αντίστοιχα αν επιθυμείται η αρίθμηση να ξεκινά από το 100, τότε η αντίστοιχη δήλωση γίνεται: DIMENSION A(100:150). Στην περίπτωση αυτή το πρώτο στοιχείο είναι το A(100) και το τελευταίο A(150), ενώ το σύνολο των στοιχείων που μπορούν να αποθηκευθούν στον πίνακα Α είναι 51. Στη δήλωση της διάστασης μιας μεταβλητής μπορούν να χρησιμοποιηθούν και αρνητικοί ακέραιοι. Π.χ. η δήλωση

DIMENSION A(-100:500)

σημαίνει ότι το πρώτο στοιχείο του πίνακα Α είναι το A(-100), το δεύτερο το A(-99) κλπ., ενώ το σύνολο των στοιχείων του πίνακα Α είναι 601.

Αντίστοιχα μπορεί να δηλωθεί η διάσταση ενός πίνακα με τις εντολές REAL και INTEGER, χωρίς να δηλώνεται με την εντολή DIMENSION. Για παράδειγμα, οι εντολές

REAL A(100)
INTEGER K(200)

είναι ισοδύναμες με τις εντολές:

DIMENSION A(100)
DIMENSION K(200)

ή με την εντολή

DIMENSION A(100), K(200)

Υπάρχουν πολλές περιπτώσεις όπου δεν είναι δυνατή η προηγούμενη γνώση του πλήθους των στοιχείων που επιθυμούμε να μπούν στα στοιχεία ενός πίνακα.

Για να είναι δυνατή η εκτέλεση ενός τέτοιου προγράμματος πρέπει η διάσταση του πίνακα να οριστεί εξ αρχής μεγάλη ώστε να μπορούν να περιληφθούν όλα τα στοιχεία, ανεξάρτητα αν μερικές θέσεις του πίνακα παραμείνουν κενές. Για παράδειγμα ας θεωρηθεί ότι σε κάποιο αρχείο βρίσκονται αριθμοί, των οποίων το πλήθος είναι μεγαλύτερο από 800 και μικρότερο από 1000. Σε μια τέτοια περίπτωση συνιστάται η διάσταση του πίνακα, στον οποίο θα αποθηκευτούν τα στοιχεία αυτά, να είναι τουλάχιστον 1000.

Ειδικά στις εντολές READ και WRITE είναι δυνατόν να χρησιμοποιείται μια διαφορετική μορφή της εντολής DO, η οποία διευκολύνει στον τρόπο γραφής και ανάγνωσης δεδομένων. Για παράδειγμα οι εντολές:

```
READ(10,*)(A(I),I=1,50)
WRITE(*,*)(A(I),I=1,50)
```

δηλώνουν, η μεν πρώτη οτι θα πρέπει να διαβαστούν απο το αρχείο με τον αριθμό 10, τα στοιχεία του πίνακα A(I) και το I θα παίρνει τιμές απο 1 έως και 50 και η δεύτερη οτι θα πρέπει να τυπωθούν οι τιμές του πίνακα A(I) για τιμές του I απο 1 έως 50. Με τον τρόπο αυτό αναγραφής ενός "κρυφού" DO LOOP (Implied DO), ο υπολογιστής θα διαβάσει (αντίστοιχα θα γράψει) τα στοιχεία απο μια γραμμή. Αν τυχόν τα στοιχεία βρίσκονται και σε επόμενες γραμμές, η ανάγνωση θα συνεχιστεί μέχρι να διαβαστεί όλο το πλήθος των στοιχείων που δηλώνεται στο "κρυφό" DO LOOP. Είναι δυνατόν να υπάρχουν περισσότερα απο ένα τέτοια "κρυφά" DO LOOPS όπως στην εντολή:

```
READ(10,*)((A(I,J),I=1,10),J=1,8)
```

Στην περίπτωση αυτή, πρώτα εκτελείται η ανακύκλωση ως προς J και κατόπιν η ανακύκλωση ως προς I. Είναι προφανές οτι για κάθε τιμή του I, ικανοποιούνται όλα τα J, κλπ.

Παράδειγμα

Σε ένα αρχείο με το όνομα INDAT βρίσκονται καταχωρημένες 50 μετρήσεις ενός μεγέθους και είναι καταχωρημένες μία μέτρηση (τιμή) ανά γραμμή. Ζητείται να υπολογιστεί η μέση τιμή των μετρήσεων.

Εφ' όσον πρόκειται για ομοειδείς μετρήσεις, αρκεί να δηλωθεί ένας πίνακας X με 50 στοιχεία. Έτσι,

```
      PROGRAM AVER
      DIMENSION X(50)
C     ΜΗΔΕΝΙΣΜΟΣ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΠΙΝΑΚΑ X
      DO 10 I=1,50
      X(I)=0.
```

```
10    CONTINUE
      OPEN (12, FILE='INDAT')
      DO 15 I=1,50
      READ(12,*)X(I)
15    CONTINUE
C TO S EINAI BOHΘHTIKH METABΛHTH ΠOY ΘA ΠEPIEXEI TO
C AΘPOIΣMA TΩN ΣTOIXEIΩN TOY ΠINAKA X
      S=0.
      DO 20 I=1,50
      S=S+X(I)
20    CONTINUE
      AV=S/50.
      WRITE (*,*)'MEΣH TIMH=',AV
      STOP
      END
```

Παράδειγμα: Να βρεθεί το γινόμενο C ενός πίνακα A(3,3) με ένα μονοδιάστατο πίνακα B(3,1)=B(3) όπου

$$A=\begin{matrix}1&2&3\\4&5&6\\7&8&9\end{matrix} \quad \text{και} \quad B=\begin{matrix}1\\2\\3\end{matrix}$$

και να τυπωθούν οι πίνακες A,B,C.

```
      PROGRAM MULT
      DIMENSION A(3,3), B(3), C(3)
      A(1,1)=1.
      A(1,2)=2.
      A(1,3)=3.
      A(2,1)=4.
      A(2,2)=5.
      A(2,3)=6.
      A(3,1)=7.
      A(3,2)=8.
      A(3,3)=9.
      B(1)=1.
      B(2)=2.
      B(3)=3.
```

```
      DO 10 I=1,3
      C(I)=0.
      DO 20 J=1,3
20    C(I) =C(I)+A(I,J)*B(J)
10    CONTINUE
      DO 30 I=1,3
30    WRITE(*,*) (A(I,J),J=1,3),'   ',B(I),'    ',C(I))
      STOP
      END
```

Τα αποτελέσματα θα είναι

1.	2.	3.	1.	14.
4.	5.	6.	2.	32.
7.	8.	9.	3.	50.

Η εντολή 30 WRITE(*,*) (A(I,J),J=1,3),' ',B(I),' ',C(I)) βρίσκεται μέσα στο DO 30. Κατα την εκτέλεση του DO σαρρώνονται οι δείκτες I, ενώ μεσα στην εντολή WRITE σαρρώνονται οι δείκτες J. Για κάθε τιμή του I θα ικανοποιηθούν όλες οι τιμές για το J και κατόπιν θα αλλάξει η τιμή του I.

Παράδειγμα:

Δίνεται ένα αρχείο INDAT στο οποίο είναι γραμμένοι τυχαίοι ακέραιοι αριθμοί, ανά ένας σε κάθε γραμμή και το πλήθος τους είναι άγνωστο αλλά μικρότερο από 1000. Ζητείται να βρεθεί το πλήθος των αριθμών και κατόπιν να τοποθετηθούν σε αύξουσα τάξη.

Εφ' όσον το πλήθος των τυχαίων αριθμών είναι μικρότερο του 1000 η διάσταση του πίνακα IA στον οποίο θα αποθηκευτούν οι αριθμοί μπορεί να είναι IA(1000).

```
      PROGRAM ARNUM
      DIMENSION IA(1000), IB(1000)
      DO 3  I=1,1000
      IA(I)=0.
3     IB(I)=0.
      OPEN (10,FILE='INDAT')
      DO 12 I=1,1000
      READ(10,*,ERR=15)IA(I)
12    CONTINUE
15    NUM =I-1
      DO 16 I=1,NUM
16    IB(I)=IA(I)
```

```
      DO 20  I1=1,NUM-1
      DO 20 I2=I1+1,NUM
      IF(IB(I1).LE.IB(I2))GO TO 19
      IE=IB(I1)
      IB(I1)=IB(I2)
      IB(I2)=IE
19    CONTINUE
20    CONTINUE
      DO 25 I=1,NUM
25    WRITE(*,*)I, IA(I),IB(I)
      STOP
      END
```

Επεξήγηση: Παρ' όλο που ένας πίνακας είναι αρκετός για την αποθήκευση των στοιχείων, ο δεύτερος (IB) χρησιμοποιείται για την αποθήκευση των στοιχείων κατά αύξουσα τάξη. Αρχικά με την ανακύκλωση DO 3 μηδενίζονται όλα τα στοιχεία των πινάκων IA και IB. Με την ανακύκλωση DO12 διαβάζονται τα στοιχεία του αρχείου INDAT και καταχωρούνται στον πίνακα IA. Σε περίπτωση που συμβεί λάθος κατά την ανάγνωση (ERR=15) η ροή του προγράμματος μεταφέρεται έξω από το πεδίο της DO 12. Στην περίπτωση αυτή, η τελευταία τιμή που είχε το I είναι γνωστή και είναι αυτή στην οποία συνέβη το λάθος κατά την ανάγνωση. Έτσι αφαιρώντας μία μονάδα από την τελευταία τιμή του I προκύπτει το πλήθος των στοιχείων που βρίσκονται στο αρχείο INDAT και είναι η τιμή της μεταβλητής NUM. Κατόπιν τα στοιχεία του IA αντιγράφονται στον πίνακα IB. Στο σημείο αυτό οι δύο πίνακες περιέχουν τον ίδιο αριθμό μή μηδενικών στοιχείων και στις ίδιες ακριβώς θέσεις. Ακολουθεί η διπλή ανακύκλωση DO 20 όπου ο δείκτης I1 παίρνει τιμές από 1 έως NUM-1 ενώ ο I2 παίρνει τιμές από την εκάστοτε τιμή του I1 μέχρι το NUM. Εφ' όσον η εσωτερική ανακύκλωση βρίσκεται κάτω από τον έλεγχο της εξωτερικής, σημαίνει ότι για κάθε τιμή του πίνακα IB με δείκτη I1 ελέγχονται με σύγκριση όλες οι τιμές του πίνακα IB με δείκτη μεγαλύτερο του I1. Η μεταβλητή IE είναι βοηθητική μεταβλητή. Αν ικανοποιείται το IF αυτό σημαίνει ότι ο αριθμός αυτός είναι μικρότερος από τους λοιπούς και παραμένει στη θέση του. Αν αυτό δεν συμβαίνει τότε γίνεται η εναλλαγή που ακολουθεί. Αφού γίνει η σύγκριση για όλους τους αριθμούς ακολουθεί η εκτύπωση του αύξοντα αριθμού, της αρχικής και τελικής σειράς σε ζεύγη.

Παράδειγμα

Δίνονται τα στοιχεία δύο πινάκων A(3,3) και B(3,3) και ζητείται ο πίνακας που είναι το γινόμενο των δύο πινάκων. Υποτίθεται ότι η ανάγνωση των στοιχείων των δύο πινάκων γίνεται από το πληκτρολόγιο (εναλλακτικά το πρόγραμμα μπορεί να διαμορφωθεί έτσι ώστε να διαβάζει τα στοιχεία από ένα ή δύο αρχεία).

```
      PROGRAM POLPI
      DIMENSION A(3,3), B(3,3), C(3,3)
      DO 10 I=1,3
      DO 11 J=1,3
      C(I,J)=0.
      READ(*,*)A(I,J)
11    CONTINUE
10    CONTINUE
      DO 12 I=1,3
      DO 12 J=1,3
      READ (*,*)B(I,J)
12    CONTINUE
      DO 15 I=1,3
      DO 15 J=1,3
      DO 15 L=1,3
      C(I,J)=C(I,J)+A(I,L)*B(L,J)
15    CONTINUE
      DO 20 I=1,3
      WRITE(*,*)(A(I,J),J=1,3),' ',(B(I,J),J=1,3),' ',(C(I,J),J=1,3)
20    CONTINUE
      STOP
      END
```

Στα DO-LOOPS 10 και 11 γίνεται η ανάγνωση των στοιχείων του πίνακα A(3,3) και συγχρόνως ο μηδενισμός των στοιχείων του πίνακα C. Στο διπλό DO LOOP 12 γίνεται η ανάγνωση των στοιχείων του πίνακα B. Τα DO LOOPS 10 και 11 με το διπλό DO LOOP 12 κάνουν ακριβώς την ίδια εργασία. Στο συνθετο DO LOOP 15 γίνεται ο υπολογισμός του πίνακα C και ακολουθεί η εκτύπωση των στοιχείων των πινάκων A, B, C.

9. ΜΕΤΑΒΛΗΤΕΣ CHARACTER

Όπως αναφέρθηκε ήδη, στο όνομα μιας μεταβλητής μπορεί να καταχωρηθεί ένας αριθμός και σε έναν πίνακα μια σειρά αριθμών, το πλήθος των οποίων εξαρτάται απο την διάσταση του πίνακα. Σε αντιστοιχία, στο όνομα μιας μεταβλητής τύπου CHARACTER μπορούν να καταχωρηθούν χαρακτήρες, γράμματα, αριθμοί και τα ειδικά σύμβολα. Ως αλφαριθμητικοί χαρακτήρες νοούνται τα 26 γράμματα του λατινικού αλφαβήτου (A-Z), τα δέκα αριθμητικά σύμβολα (0.-9), οι ειδικοί χαρακτήρες (+, -, *, /, =, (,), ., $, :) και ο χαρακτήρας του κενού σημείου.

Σε αντιστοιχία με τις μεταβλητές με δείκτες, οι μεταβλητές τύπου CHARACTER δηλώνονται στην αρχή του προγράμματος, πρίν ή μετά την εντολή DIMENSION και οπωσδήποτε πρίν απο οποιαδήποτε εκτελέσιμη εντολή του προγράμματος, και δηλώνεται ως εξής:

CHARACTER*W A,B(10), C*n, D*1,E

όπου W είναι ένας ακέραιος αριθμός που δηλώνει το μήκος του πεδίου κάθε μεταβλητής

A, C, D, E είναι μεταβλητές τύπου CHARACTER.

B είναι πίνακας τύπου CHARACTER με διάσταση 10 και όπου κάθε στοιχείο του πίνακα (B(1), B(2),....., B(10)) χωρά W χαρακτήρες.

n είναι το μήκος του πεδίου της μεταβλητής C

η δε μεταβλητή D έχει έναν χαρακτήρα.

Ο καθορισμός τιμής μιας μεταβλητής A τύπου CHARACTER, μέσα στο πρόγραμμα γίνεται ως εξής:

A='TIME'

Είναι υποχρεωτική η εμφάνιση της τιμής μεταξύ δύο αποστρόφων στην περίπτωση καθορισμού τιμής μιας μεταβλητής CHARACTER. Είναι δυνατόν μια μεταβλητή τύπου CHARACTER να πάρει την τιμή μιας άλλης μεταβλητής τύπου CHARACTER οπότε στην περίπτωση αυτή δεν πρέπει να χρησιμοποιηθούν απόστροφοι.

Παράδειγμα

```
PROGRAM CHRT
CHARACTER*5 A,B
A='HOUR'
B=A
...
```

οπότε η μεταβλητή B περιέχει τα στοιχεία HOUR.

Παράδειγμα

```
PROGRAM CHRE
CHARACTER A*8, B*7, C*15
A='ABCDE'
B='IJKLMN'
C=A//B
...
```

με την εντολή C=A//B δηλώνεται ότι γίνεται συνένωση των χαρακτήρων που περιέχονται στις μεταβλητές A και B. Η συνένωση δηλώνεται με τη διπλή κάθετο (//). Γενικά η εντολή

CHAR=CHAR1//CHAR2//.....//CHARN

συνενώνει τις τιμές των μεταβλητών CHAR1, CHAR2,..... και η τελική τιμή δίνεται στη μεταβλητή CHAR, εφ'όσον έχει αρκετό μήκος πεδίου για να χωρέσουν όλοι οι χαρακτήρες μετά την συνένωση.

Επιτρέπεται επίσης η έκφραση

CHAR=A//'XY'//B

Αν A='ABC' και B='DEF' με την εντολή αυτή (εφ' όσον η CHAR έχει αρκετό μήκος πεδίου), η μεταβλητή CHAR έχει την τιμή

ABCXYDEF

Είναι δυνατόν να γίνουν πράξεις σε μέρος ή σε όλο το μήκος του πεδίου μιας μεταβλητής CHARACTER.

Παράδειγμα

```
      PROGRAM TESTCH
      CHARACTER*10 W,B,C*12,D*4,E*5,F*6,X*11
      W='DEMOKRITOS'
      B='SOCRATES'
      C='ARISTOTELES'
      D=W(1:4)
      E=B(3:7)
      F=C(1:6)
      WRITE (*,4)D
      WRITE (*,5)E
      WRITE (*,6)F
4     FORMAT(A4)
5     FORMAT(A5)
6     FORMAT(A6)
      X=F//E
```

```
      WRITE (*,7)X
7     FORMAT (A12)
      STOP
      END
```

Η μεταβλητή D με την εντολή D=W(1:4) παίρνει την τιμή DEMO, η E την CRATE, η F την ARISTO και η X την ARISTOCRATE.

Στις περιπτώσεις όπου η ανάγνωση της τιμής μιας μεταβλητής CHARACTER γίνεται από το πληκτρολόγιο, η είσοδος είναι λίγο διαφορετική

Παράδειγμα

```
      PROGRAM TERCH
      CHARACTER INP*10
      WRITE(*,10)
10    FORMAT ('ENTER INPUT FILE-NAME')
      READ (*,20)INP
20    FORMAT (A10)
      OPEN (10,FILE=INP)
      .
      .
      STOP
      END
```

Η μεταβλητή INP τύπου CHARACTER θα διαβαστεί από το πληκτρολόγιο και στη δήλωση του ονόματος του αρχείου που θα ανοίξει η εντολή OPEN, τούτο δεν πρέπει να γραφεί μεταξύ αποστρόφων.

Οι δύο εντολές

```
      READ(*,20)INP
20    FORMAT (A10)
```

είναι δυνατόν να αντικατασταθούν από την ισοδύναμη εντολή

```
      READ(*,'(A10)')INP
```

Στις μεταβλητές τύπου CHARACTER μπορούν να αντιστοιχηθούν μόνον χαρακτήρες και είναι διαφορετικές απο τις πραγματικές και ακέραιες μεταβλητές, στις οποίες αντιστοιχούνται μόνον πραγματικές ή ακέραιες τιμές. Λόγω της ιδιαιτερότητας των μεταβλητών τύπου CHARACTER δεν είναι δυνατόν να εκτελεστούν οι αλγεβικές πράξεις (πρόσθεση, αφαίρεση, κλπ.). Επίσης κατα την

σύγκριση (με την εντολή IF) πρέπει μια μεταβλητή CHARACTER να συγκρίνεται μόνον με κάποια άλλη ομοιειδή της και με τίποτε άλλο. Σε μία μεταβλητή τύπου CHARACTER μπορεί να αντιστοιχηθεί μία ακολουθία αριθμών, δεν είναι δυνατόν όμως να γίνει κάποια αλγεβική πράξη με την μεταβλητή αυτή., Για παράδειγμα με τις εντολές:

```
PROGRAM CHRP
CHARACTER B*10
...
B=' 12345'
...
Y=12345
...
```

ορίζονται οι τιμές των μεταβλητών B (που είναι μεταβλητή CHARACTER και της Y (που είναι πραγματική μεταβλητή). Δέν είναι δυνατόν όμως να γίνει καμμία σύγγριση ή αλγεβρική πράξη μεταξύ των μεταβλητών B και Y, διότι πρόκειται για διαφορετικού τύπου μεταβλητές.

10. ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ ΜΕ ΠΡΟΚΑΘΟΡΙΣΜΕΝΗ ΜΟΡΦΗ (FORMAT)

Με τις εντολές READ(IN,*) και WRITE(I0,*) όπως αναφέρθηκαν μέχρι τώρα, η ανάγνωση στοιχείων και η αναγραφή αποτελεσμάτων, γινόταν ελεύθερα από τον υπολογιστή. Σε πολλές περιπτώσεις όμως ο προγραμματιστής επιθυμεί η ανάγνωση δεδομένων και η εκτύπωση αποτελεσμάτων να γίνεται με συγκεκριμένο τρόπο.

Η εντολή FORMAT επιτρέπει το καθορισμό του τρόπου ανάγνωσης και εκτύπωσης. Κάθε εντολή FORMAT πρέπει να χαρακτηρίζεται από έναν αριθμό π.χ.

```
10      FORMAT (TE,TE,TE)
```

Ο αριθμός που χαρακτηρίζει τη FORMAT αναφέρεται σε εντολή READ ή WRITE αντικαθιστώντας το δεύτερο στοιχείο ορισμού το οποίο μέχρι τώρα ήταν το σύμβολο *. π.χ.

```
        READ(*,12)X,Y
    12  FORMAT(TE1,TE2,.).
```

όπου το 12 είναι ο αριθμός που αντιστοιχεί στην εντολή αυτή. Η εντολή FORMAT δεν είναι εκτελέσιμη και κατά συνέπεια δεν επιτρέπεται η μεταφορά της ροής του προγράμματος στην εντολή αυτή μέσω μιας εντολής GOTO.

Οι παράμετροι TE1,TE2 κλπ., καθορίζουν τον τρόπο με τον οποίο θα διαβαστούν ή θα εκτυπωθούν οι τιμές των μεταβλητών.

10.1 ΚΩΔΙΚΑΣ I

Ο κώδικας I χρησιμοποιείται για την ανάγνωση ή την εκτύπωση ακέραιων αριθμών. Το πλήθος των χαρακτήρων (μήκος πεδίου) που θα διαβαστούν ή θα τυπωθούν ορίζεται από έναν αριθμό που γράφεται αμέσως μετά το I. Η γενική μορφή της εντολής είναι

```
        FORMAT (Iw)
```

όπου w είναι το πλήθος των ψηφίων που θα τυπωθούν (ή θα διαβαστούν) η δε εκτύπωση εμφανίζεται ως ±nnn....n, όπου τυπώνεται το πρόσημο αν ο αριθμός είναι αρνητικός και παραμένει ένα κενό αν ο αριθμός είναι θετικός και nnn.....n είναι ο αριθμός με τόσες θέσεις, όσες ορίζει ο αριθμός w. Ο αριθμός που εκτυπώνεται τοποθετείται στο δεξιό μέρος του πεδίου, έτσι ώστε αν παραμείνουν κενές θέσεις αυτές να είναι στο αριστερό μέρος του πεδίου εκτύπωσης. Στην περίπτωση που ο αριθμός αποτελείται από περισσότερα ψηφία από όσα έχουν οριστεί στο πεδίο, τότε θα τυπωθούν αστεράκια στο πεδίο εκτύπωσης.

Παράδειγμα

```
      PROGRAM TESTI
      I=127
      J=I*9252
      K=-532
      WRITE(*,10)I,J,K
10    FORMAT(I5,I6,I5)
      STOP
      END
```

Τα αποτελέσματα θα τυπωθούν στην οθόνη ως εξής

##127******#-532 (Το σύμβολο # δηλώνει κενό χαρακτήρα). Στην εκτύπωση της τιμής του J το πεδίο εκτύπωσης που ορίστηκε (I6) δεν είναι αρκετό για να χωρέσει ο αριθμός ο οποίος είναι επταψήφιος.

Ο κανόνας που ακολουθείται είναι ο εξής: Ο αριθμός που καθορίζει το πεδίο εκτύπωσης (ή ανάγνωσης) πρέπει να είναι μεγαλύτερος από το πλήθος των ψηφίων του αριθμού που πρόκειται να εκτυπωθεί. Θα πρέπει επίσης να λαμβάνεται υπόψη και μία θέση για το πρόσημο του αριθμού.

Πολλές φορές, για λόγους ευκρίνειας και ευκολίας στην ανάγνωση των αποτέλεσμάτων, ο προγραμματιστής θέλει να παρεμβάλει κενά μεταξύ των τιμών που τυπώνονται. Ο κώδικας X δίνει αυτή τη δυνατότητα μέσω της εντολής FORMAT. Στο προηγούμενο παράδειγμα, αν η εντολή FORMAT αντικατασταθεί από την

```
10    FORMAT(2X,I5,3X,I8,5X,I5)
```

τα αποτελέσματα θα τυπωθούν στην οθόνη ως

####127####1175004######-532

Με την εντολή FORMAT μπορούν να τυπωθούν και επεξηγήσεις στην ίδια ή σε συνεχόμενες γραμμές, μαζί με τις τιμές των παραμέτρων.

Παράδειγμα.

```
      PROGRAM TEST3
      I=5
      J=2
      K=3*5+J**2
      WRITE(*,10)I,J,K
10    FORMAT('FOR I=',I3,'AND J=',I3,/,
     >' THE VALUE OF THE EXPRESSION',/,
     >' K=3*5+J**2' IS EQUAL TO: ',I6)
      STOP
      END
```

Τα αποτελέσματα θα εμφανιστούν στην οθόνη ως εξής:

```
FOR I=5 AND J=2
THE VALUE OF THE EXPRESSION
K=3*5+J**2 IS EQUAL TO:   19
```

Κάθε έκφραση, μέσα στην εντολή FORMAT, που περιλαμβάνεται μεταξύ δύο αποστρόφων ('), τυπώνεται όπως ακριβώς βρίσκεται. Στην περίπτωση που το μήνυμα ή οι τιμές καταλαμβάνουν μήκος περισσότερο από μία γραμμή, τότε χρησιμοποιείται το σύμβολο / μεταξύ δύο κόμμα (,) το οποίο δίνει εντολή στον υπολογιστή να συνεχίσει την εκτύπωση στην επόμενη γραμμή. Είναι προφανές ότι δεν μπορεί να διακοπεί η εκτύπωση της τιμής μιας μεταβλητής και να συνεχιστεί στην επόμενη γραμμή.

10.2 ΚΩΔΙΚΑΣ F

Ο κώδικας F χρησιμοποιείται για την ανάγνωση και εκτύπωση πραγματικών αριθμών, στις περιπτώσεις εκείνες όπου δε χρειάζεται εκτύπωση με δύναμη του 10. Στη γενική μορφή ο κώδικας F συντάσσεται ως Fw.d, όπου w είναι το ολικό μήκος του πεδίου εκτύπωσης και d είναι ο αριθμός των δεκαδικών ψηφίων που θα διαβαστούν ή θα τυπωθούν.

Στο ολικό μήκος του πεδίου εκτύπωσης (ή ανάγνωσης) περιλαμβάνεται μία θέση για το πρόσημο του αριθμού και μία θέση για τη δεκαδική τελεία. Έτσι γενικά θα πρέπει w-d>2. Αν ο αριθμός που ζητείται να τυπωθεί, έχει περισσότερα δεκαδικά ψηφία από όσα δηλώνεται στον κώδικα F, τότε τυπώνονται τόσα δεκαδικά όσα ορίζει ο κώδικας F, με στρογγύλευση των υπολοίπων δεκαδικών που απομένουν. Η δήλωση F12.6 σημαίνει ότι ο αριθμός θα τυπωθεί σε ένα ολικό πεδίο 12 χαρακτήρων, με 6 δεκαδικά ψηφία. Αφαιρώντας δύο χαρακτήρες για το πρόσημο και τη δεκαδική τελεία, μένουν 4 θέσεις για την εκτύπωση του ακέραιου μέρους του πραγματικού αριθμού. Έτσι λοιπόν ο μέγιστος (αντίστοιχα μικρότερος) αριθμός, που μπορεί να τυπωθεί με τη δήλωση F12.6, είναι ±9999.999999.

Παράδειγμα

```
      PROGRAM TESTF
      A=125.67
      B=-25365.3267
      C=123456789.123458
      WRITE(*,10)A,B,C
10    FORMAT(F4.3,3X,F18.6,2X,F20.5)
      END
```

Τα αποτελέσματα θα εμφανιστούν ως εξής:
****########-25365.326700######123456789.12346

Το πρώτο πεδίο εκτύπωσης, για τη μεταβλητή A, δεν είναι αρκετό για να γίνει η εκτύπωση του αριθμού 125.67, με συνέπεια να τυπωθούν 4 αστεράκια, όσα και το μήκος του πεδίου (F4.3). Για τη μεταβλητή B το πεδίο έχει περισσότερο χώρο, ο οποίος θα μείνει κενός πρίν από τον αριθμό. Η απαίτηση για εκτύπωση με 6 δεκαδικά ψηφία, αναγκάζει τον υπολογιστή να προσθέσει δύο μηδενικά στο τέλος του δεκαδικού μέρους. Στη μεταβλητή C, το πεδίο έχει αρκετό μήκος για να χωρέσει ο αριθμός, αλλά με την απαίτηση να τυπωθούν 5 δεκαδικά ψηφία. Έτσι λοιπόν ο υπολογιστής στρογγυλοποιεί το αποτέλεσμα στα 5 δεκαδικά. Σε πολλές περιπτώσεις όμως το πλήθος των ψηφίων που αποθηκεύονται στον υπολογιστή είναι μικρότερο από το πλήθος που έχει αναγραφεί. Αυτό συμβαίνει στην τελευταία περίπτωση όπου τα τελευταία ψηφία που θα τυπωθούν θα είναι διαφορετικά από αυτά που δόθηκαν.

10.3 ΚΩΔΙΚΑΣ E

Ο κώδικας E χρησιμοποιείται όταν πρόκειται για πολύ μεγάλους ή πολύ μικρούς αριθμούς και η αναπαράστασή τους απαιτεί τη χρήση εκθετικού (δύναμη του 10). Χρησιμοποιείται επίσης σε περιπτώσεις όπου η τιμή ενός αποτελέσματος, που θα εκτυπωθεί, δεν είναι γνωστή κατά τη συγγραφή του προγράμματος. Η γενική μορφή του κώδικα E είναι Ew.d όπου:

- το E δηλώνει ότι θα χρησιμοποιηθεί συμβολισμός με χρήση δύναμης του 10
- Το w δηλώνει το ολικό μήκος του πεδίου που θα χρησιμοποιηθεί και όπου συμπεριλαμβάνονται: Το πρόσημο του αριθμού, η δεκαδική τελεία, το σύμβολο E, το πρόσημο του εκθετικού και η δύναμη του εκθετικού.
- Ο αριθμός d δηλώνει πόσα δεκαδικά ψηφία θα τυπωθούν στην αναπαράσταση του αριθμού.
- Η τιμή του w θα πρέπει να είναι μεγαλύτερη από το άθροισμα d+6

Αν το πεδίο δεν είναι αρκετό για να χωρέσει ο αριθμός, τότε θα τυπωθούν αστεράκια. Αν το πεδίο είναι μεγαλύτερο, τότε όσα κενά υπάρξουν θα είναι αριστερά του αριθμού. Στην περίπτωση που η τιμή της μεταβλητής, που ζητείται να τυπωθεί, δεν έχει οριστεί προηγουμένως, τότε θα τυπωθεί ο χαρακτήρας I. στο χώρο του πεδίου. Αν δε ο αριθμός είναι έξω από τα όρια που δέχεται ο υπολογιστής, τότε θα τυπωθεί ένα R.

Παράδειγμα

```
      PROGRAM TESTE
      A=123.627
      B=12345678.92
      C=-0.0000000125
      D=0.000000328
      WRITE(*,10)A,B,C,D
10    FORMAT(E8.7,2X,E12.4,2X,E15.5,2X,E12.5)
      STOP
      END
```

Τα αποτελέσματα θα εμφανιστούν ως εξής

********####0.1235E08#####-0.12500E-07###0.32800E-06

Στην πρώτη περίπτωση της εκτύπωσης της τιμής του A, το ολικό πεδίο εκτύπωσης δεν είναι αρκετό για να τυπωθεί ο αριθμός. Στην εκτύπωση της τιμής του B γίνεται στρογγύλευση και στις άλλες δυο περιπτώσεις απλή αλλαγή του τρόπου αναπαράστασης.

Στις περιπτώσεις όπου η δύναμη του εκθετικού είναι τριψήφιος αριθμός, τότε καλύπτεται μία ακόμη θέση για το εκθετικό από το ολικό μήκος του πεδίου εκτύπωσης.

10.4 ΚΩΔΙΚΑΣ D ΚΑΙ G

Ο κώδικας D χρησιμοποιείται για την ανάγνωση και εκτύπωση αριθμών διπλής ακρίβειας και υπακούει στους ίδιους κανόνες όπως και ο κώδικας E.

Ο κώδικας G θα μπορούσε να θεωρηθεί ότι καλύπτει τις περιπτώσεις των κωδίκων E και F. Η γενική του μορφή είναι Gw.d όπου w δηλώνει το ολικό μήκος του πεδίου εκτύπωσης και d ο αριθμός των σημαντικών ψηφίων που θα τυπωθούν ανεξάρτητα αν είναι πριν ή μετά από τη δεκαδική τελεία. Στην περίπτωση αυτή επίσης πρέπει να ικανοποιείται η σχέση w>d+6.

Με τον κώδικα G, η εκτύπωση εξαρτάται από την τάξη μεγέθους του αριθμού που πρόκειται να τυπωθεί και από το ολικό μήκος του πεδίου. Αν το μήκος του πεδίου καλύπτει τον αριθμό, τότε θα χρησιμοποιηθεί ο κώδικας F, σε άλλη περίπτωση θα χρησιμοποιηθεί ο κώδικας E.

Παράδειγμα

```
      PROGRAM TESTG
      A=132.4532
      B=0.127345E-10
      WRITE (*,10)A,B
10    FORMAT (G10.3,2X,G10.3)
      STOP
      END
```

Τα αποτελέσματα θα εμφανιστούν ως εξής:

####132.453#0.127E-10

Όπου στην πρώτη περίπτωση χρησιμοποιήθηκε κώδικας F και στην δεύτερη κώδικας E.

10.5 ΚΩΔΙΚΑΣ A

Ο κώδικας αυτός χρησιμοποιείται για την ανάγνωση και εκτύπωση αλφαριθμητικών χαρακτήρων. Το μήκος του πεδίου που θα χρησιμοποιηθεί δηλώνεται από έναν αριθμό που γράφεται δίπλα στο A. Με τη δήλωση αυτή, όλο το πεδίο γεμίζει με κενά. Αν ο αριθμός των χαρακτήρων που θα καταχωρηθεί είναι μικρότερος από το μήκος του πεδίου, τα στοιχεία καταχωρούνται αρχίζοντας από αριστερά έτσι ώστε αν μείνουν κενά, αυτά να είναι μαζεμένα στο δεξιό μέρος του πεδίου. Αν το μήκος του πεδίου είναι μικρότερο από τον αριθμό των χαρακτήρων που καταχωρούνται, παραμένουν μόνον οι από αριστερά προς τα δεξιά χαρακτήρες που ικανοποιούν το μήκος του πεδίου. Οι υπόλοιποι χαρακτήρες αγνοούνται.

Παράδειγμα

```
      PROGRAM TESTA
      CHARACTER*10 C, B*12
      READ(*,20)C
      READ(*,'(A)')B
20    FORMAT(A10)
      WRITE(*,20)C
      WRITE(*,25)B
25    FORMAT(A15)
      STOP
      END
```

Αν υποτεθεί ότι στο πρώτο READ(*,20)C δίνεται η φράση VENUS 127 και στο δεύτερο η φράση HERMES IS A PLANET τα στοιχεία που θα καταχωρηθούν στις μεταβλητές τύπου CHARACTER θα είναι:

A: VENUS 127

B: HERMES IS A

Σημειώνεται ότι και τα κενά αντιμετωπίζονται ως χαρακτήρες. Έτσι η λέξη PLANET δεν θα καταχωρηθεί πουθενά. Κατά την εκτύπωση λοιπόν θα εμφανιστούν τα εξής

VENUS 127

HERMES IS A

10.6 ΚΩΔΙΚΑΣ T

Στις περιπτώσεις που είναι απαίτηση του χρήστη, η ανάγνωση ή η εκτύπωση να αρχίσει από συγκεκριμένη στήλη, χρησιμοποιείται ο κώδικας T. Η γενική μορφή σύνταξης είναι

Tn

όπου n ένας ακέραιος θετικός αριθμός. Μετά από τον κώδικα T, πρέπει να οριστεί με ποιόν τρόπο (κώδικας F,G,E,κλπ) θα γίνει η ανάγνωση ή εκτύπωση.

Παράδειγμα

```
      PROGRAM TESTT
      OPEN(5,FILE='INPUT')
      READ(5,10)IA,B,C
10    FORMAT(T2,I4,T8,F8.1,T25,F9.2)
      WRITE (*,20)B
20    FORMAT (T10,F12.2)
      .
      .
      .
      STOP
      END
```

Αν υποτεθεί ότι στο αρχείο INPUT είναι γραμμένα τα ακόλουθα,

```
##127####1234567.2########34567.1
12345678901234567890123456789 0
      1       2       3
```

τότε το IA θα πάρει την τιμή 127 το B την τιμή 1234567. και το C την τιμή 4567.1. (Οι δύο γραμμές με τους αριθμούς σε πλάγια γραφή είναι βοηθητικές για να είναι εύκολη η καταμέτρηση του πλήθους των στοιχείων και η θέση τους). Με την εντολή ο δείκτης μετακινείται κατα δύο θέσεις και διαβάζει, με αρχή την θέση 2, τον πρώτο αριθμό. Στη συνέχεια μετακινείται στην θέση 8 και διαβάζει, με αρχή την θέση 8,

τον δεύτερο αριθμό, και κατόπιν στην θέση 25, όπου διαβάζει, με αρχή την θέση 25, τον τρίτο αριθμό. Στην εκτύπωση της τιμής του Β θα παραμείνουν 10 κενά και κατόπιν θα γραφεί η τιμή του Β, αρχίζοντας απο την δέκατη θέση.

10.7 ΚΩΔΙΚΑΣ HOLLERITH

Ένας διαφορετικός τρόπος αναγραφής ενός μηνύματος, από αυτόν που το μήνυμα τοποθετείται μεταξύ αποστρόφων, είναι η χρησιμοποίηση του κώδικα Η. Η γενική μορφή σύνταξής του είναι nH, όπου n ένας ακέραιος θετικός αριθμός, ο οποίος δηλώνει ότι θα τυπωθούν οι n χαρακτήρες που ακολουθούν το Η, όπως έχουν Παράδειγμα

```
      WRITE(*,20)
20    FORMAT(31H THIS IS THE END OF THE PROGRAM)
```

Πρέπει να δίνεται ιδιαίτερη προσοχή στη χρήση του κώδικα Η, διότι ο αριθμός n πρέπει να είναι ίσος με τους χαρακτήρες που ακολουθούν το Η. Σημειώνεται ότι τα κενά μετρώνται κανονικά ως χαρακτήρες. Αυτό σημαίνει ότι ο προγραμματιστής πρέπει πρώτα να γράψει το μήνυμα και κατόπιν να μετρήσει τους χαρακτήρες για να σημπληρωθεί σωστά η εντολή.

Το παρακάτω παράδειγμα είναι λάθος διότι ο αριθμός των χαρακτήρων που δηλώθηκε στον κώδικα Η είναι μικρότερος από το πλήθος των χαρακτήρων που ακολουθούν το Η

```
      WRITE (*,10)
10    FORMAT (12H END OF MAIN PROGRAM)
```

11. ΕΓΓΡΑΦΗ ΚΑΙ ΑΝΑΓΝΩΣΗ ΧΩΡΙΣ ΚΩΔΙΚΑ

Μερικές φορές προκύπτει η ανάγκη εγγραφής και ανάγνωσης στοιχείων σε ένα αρχείο, χωρίς να ενδιαφέρει άμεσα το χρήστη τί είναι γραμμένο στο αρχείο αυτό. Μια τέτοια διαδικασία μπορεί να χρειαστεί σε περιπτώσεις εκτέλεσης διαδοχικών προγραμμάτων, όταν η έξοδος του ενός προγράμματος αποτελεί είσοδο για το άλλο. Υπάρχουν δε περιπτώσεις όπου το μήκος των αρχείων αυτών είναι πολύ μεγάλο, αν εγγραφεί με τις ως τώρα γνωστές μεθόδους, με συνέπεια να δημιουργείται πρόβλημα χώρου αποθήκευσης. Το αρχείο αυτό μπορεί να γραφεί με δυαδική (binary) μορφή και δεν είναι δυνατόν βέβαια να το διαβάσει άμεσα ο χρήστης. Μπορεί όμως να το διαβάσει μέσω ενός άλλου προγράμματος. Ένα αρχείο σε μορφή binary καταλαμβάνει πολύ λιγότερο χώρο από ότι η αντίστοιχη αναγνώσιμη μορφή του ίδιου αρχείου. Η γενική μορφή της εντολής εγγραφής χωρίς κώδικα είναι

WRITE(n)var

και η αντίστοιχη εντολή ανάγνωσης είναι

READ(n)var

όπου n είναι ο αριθμός ενός αρχείου και var τα ονόματα των μεταβλητών που θα καταχωρηθούν

Παράδειγμα

```
PROGRAM TESTUWR
DIMENSION A(1000),B(1000)
OPEN(10,FILE='TEMOUT')
WRITE(10)(A(I),I=1,1000)
.
READ(10)(B(I),I=1,1000)
.
.
STOP
END
```

Ότι πληροφορία έχει γραφεί στο αρχείο TEMOUT, σύμφωνα με όσα αναφέρθηκαν, δεν μπορεί να διαβαστεί άμεσα από το χρήστη με τη βοήθεια κάποιου επεξεργαστή κειμένου. Μπορούν όμως να διαβαστούν από πρόγραμμα, από το ίδιο ή από άλλο, και να χρησιμοποιηθούν. Στο παράδειγμα, κατά την εγγραφή, καταχωρούνται 1000 στοιχεία του πίνακα A στο αρχείο TEMOUT. Αν μετέπειτα ζητηθεί να αναγνωστούν περισσότερα από 1000 στοιχεία, τότε υπάρχει λάθος και σταματά η εκτέλεση του προγράμματος. Στην περίπτωση που ζητηθεί να αναγνωστούν λιγότερα στοιχεία,

από όσα είναι καταγραμμένα, τότε δεν υπάρχει πρόβλημα και η εκτέλεση του προγράμματος θα προχωρήσει κανονικά.

12. ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Υπάρχουν πολλές περιπτώσεις όπου μία διαδικασία χρειάζεται να επαναληφθεί πολλές φορές κατά τη διάρκεια εκτέλεσης ενός προγράμματος. Για την αποφυγή αυτών των επαναλήψεων, οι οποίες ενέχουν πολλούς κινδύνους λαθών, είναι δυνατόν η συγκεκριμένη διαδικασία να γραφεί μία φορά μόνον με τη μορφή υποπρογράμματος. Η δυνατότητα αυτή προσφέρει πολλά πλεονεκτήματα στη συγγραφή προγραμμάτων για την επίλυση πολύπλοκων προβλημάτων, τα οποία μπορούν να επιμεριστούν σε μικρά προβλήματα.

Μερικά παραδείγματα, όπου κάποια διαδικασία είναι προτιμότερο να βρίσκεται σε χωριστή ενότητα, φαίνονται παρακάτω

1. Διάταξη στοιχείων κατά αύξουσα ή φθίνουσα σειρά.
2. Επίλυση εξισώσεων
3. Επίλυση συστημάτων γραμμικών εξισώσεων
4. Υπολογισμός ολοκληρωμάτων
5. Υπολογισμός οριζουσών
6. Διαγωνοποίηση πινάκων

κλπ.

Τα υποπρογράμματα χωρίζονται σε τρείς κατηγορίες

- Υποπρογράμματα Υπορουτίνες (Subroutine Subprograms)
- Υποπρογράμματα Συναρτήσεις (Function Subprograms)
- Εντολές Συναρτήσεων (Statement Functions)

12.1 ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ ΥΠΟΡΟΥΤΙΝΕΣ
Subroutine Subprograms

Η υπορουτίνα (Subroutine) είναι η γενικώτερη μορφή υποπρογράμματος. Θα πρέπει να θεωρείται ως ένα αυτοτελές, αλλά όχι ανεξάρτητο πρόγραμμα, το οποίο επιτελεί ορισμένη εργασία, ελεγχόμενη από το κυρίως πρόγραμμα. Μια υπορουτίνα αποτελείται από πολλές εντολές FORTRAN και η πρώτη γραμμή πρέπει οπωσδήποτε να δηλώνει την ιδιότητά της. Έτσι η πρώτη εντολή είναι:

```
SUBROUTINE SNAME(VAR1,VAR2,VARi)
```

και θα πρέπει να τελειώνει οπωσδήποτε με τις εντολές

```
RETURN
END
```

Ας θεωρηθεί ένα πρόγραμμα το οποίο διαβάζει 50 τυχαίους αριθμούς και τους κατατάσσει σε αύξουσα τάξη. Το πρόγραμμα αυτό μπορεί πολύ εύκολα να

μετασχηματιστεί σε κυρίως πρόγραμμα και υπορουτίνα, όπως φαίνεται στο παρακάτω παράδειγμα. Σε μια τέτοια περίπτωση το κυρίως πρόγραμμα θα αναλαμβάνει την ανάγνωση των τυχαίων αριθμών και η υπορουτίνα θα αναλαμβάνει την ταξινόμηση των αριθμών αυτών σε αύξουσα τάξη. Στο κατάλληλο σημείο του κυρίως προγράμματος θα πρέπει να τεθεί η εντολή CALL SNAME, κατά την εκτέλεση της οποίας, η ροή του προγράμματος μεταβιβάζεται στην υπορουτίνα. Μετά την εκτέλεση των εντολών RETURN και END της υπορουτίνας, η ροή του προγράμματος μεταφέρεται και πάλι στο κυρίως πρόγραμμα αμέσως μετά την εντολή CALL SNAME.

Παράδειγμα

```
      PROGRAM TESTS
      DIMENSION X(50)
      DO 10 I=1,50
10    X(I)=0.
      WRITE (*,12)50
12    FORMAT ('ENTER',I3,'NUMBERS')
      DO 15 I=1,50
15    READ(*,20)X(I)
20    FORMAT(F8.0)
      DO 100 I1=1,50-1
      DO 101 I2=I1.+1,50
      IF(X(I1).LE.X(I2))GOTO102
      TM=X(I1)
      X(I1)=X(I2)
      X(I2)=TM
102   CONTINUE
101   CONTINUE
100   CONTINUE
      WRITE(*,110)
      DO 120 I=1,50
120   WRITE(*,130)X(I)
110   FORMAT('OUTPUT. NUMBERS IN ASCENDING ORDER')
130   FORMAT (2X,F8.0)
      STOP
      END
```

Το πρόγραμμα αυτό μετασχηματίζεται σε κυρίως πρόγραμμα και υπορουτίνα ως ακολούθως

```
      PROGRAM TESTSU
      DIMENSION X(50)
      DO 10 I=1,50
10    X(I)=0.
      DO 12 I=1,50
      WRITE (*,15)I
      READ(*,18)X(I)
12    CONTINUE
15    FORMAT ('ENTER REAL NUMBER NO=',I3)
18    FORMAT (F8.0)
      CALL ARANG(X,50)
      WRITE (*,20)
      DO 25 I=1,50
25    WRITE (*,30)X(I)
20    FORMAT ('OUTPUT NUMBERS IN ASCENDING ORDER')
30    FORMAT (2X,F8.0)
      STOP
      END
      SUBROUTINE ARANG(Y,J)
      DIMENSION Y(J)        ( ή  DIMENSION Y(50))
      DO 1 I1=1,J-1
      DO 2 I2=I1+1,J
      IF(Y(I1).LE.Y(I2))GOTO4
      TR=Y(I1)
      Y(I1)=Y(I2)
      Y(I2)=TR
4     CONTINUE
2     CONTINUE
1     CONTINUE
      RETURN
      END
```

Τα δύο προγράμματα εκτελούν την ίδια εργασία. Η διαφορά τους έγκειται στην παρουσία της υπορουτίνας ARANG στο δεύτερο παράδειγμα. Η κλίση της υπορουτίνας γίνεται με την εντολή κλήσης CALL ARANG(X,50) και η ροή του προγράμματος μεταφέρεται στην υπορουτίνα, η οποία δηλώνεται ως SUBROUTINE ARANG(Y,J) (εντολή δήλωσης). Οι τιμές , ή μεταβλητές, που βρίσκονται στην

εντολή κλήσης και δήλωσης έχουν αντιστοιχία ένα προς ένα. Είναι δυνατόν, εφ' όσον διευκολύνει τον προγραμματιστή, το όνομα των μεταβλητών μέσα στην υπορουτίνα να είναι διαφορετικό από αυτό της κλήσεως. Στην προκειμένη περίπτωση ο πίνακας X του κυρίως προγράμματος έχει μετονομαστεί σε Y στην υπορουτίνα και έτσι όλα τα στοιχεία του πίνακα X μεταβιβάζονται στον πίνακα Y κατά την κλήση της υπορουτίνας. Αντίστοιχα στην επιστροφή της ροής στο κυρίως πρόγραμμα, τα στοιχεία του πίνακα Y μεταβιβάζονται στον πίνακα X. Η τιμή 50 στην κλήση της υπορουτίνας, μεταβιβάζεται ως τιμή της μεταβλητής J στην δήλωση της υπορουτίνας. Ένα σημείο άξιο προσοχής είναι ότι δεν είναι απαραίτητο μια εντολή FORMAT να ακολουθεί την εντολή READ ή WRITE στην οποία αναφέρεται, αλλά μπορεί να βρίσκεται οπουδήποτε μέσα στο πρόγραμμα. Είναι σηνήθης πρακτική, πολλοί προγραμματιστές να συγκεντρώνουν όλα τα FORMAT στο τέλος του προγράμματος.

Η δεύτερη εντολή στην υπορουτίνα, μετά τη δήλωσή της, είναι η εντολή DIMENSION Y(J). Κανονικά θα έπρεπε να είχε δηλωθεί ως DIMENSION Y(50). Εφ' όσον όμως η τιμή του J είναι γνωστή και ίση με 50, από την κλήση της υπορουτίνας, η εντολή μεταβλητής διάστασης, όπως είναι η εντολή DIMENSION Y(J), στην προκειμένη περίπτωση, να είναι δυνατόν να χρησιμοποιηθεί. Γενικά όμως δεν μπορεί να υπάρξει δήλωση μεταβλητής διάστασης ενός πίνακα στην FORTRAN. Μπορεί να δηλωθεί μεταβλητή διάσταση πίνακα μόνον στην περίπτωση που όλες οι απαραίτητες τιμές των μεταβλητών μεταβιβάζονται σε υπορουτίνα, όπως στην συγκεκριμένη περίπτωση του παραδείγματος.

Οι εντολές που ακολουθούν, είναι ακριβώς οι ίδιες όπως και στο προηγούμενο παράδειγμα που δεν υπάρχει υπορουτίνα. Αφού ικανοποιηθούν οι εντολές της υπορουτίνας, και πριν εκτελεστούν οι εντολές RETURN και END, στον πίνακα Y βρίσκονται οι αριθμοί που εδόθησαν διατεταγμένοι κατά αύξουσα σειρά. Με την εκτέλεση των εντολών RETURN και END, η ροή του προγράμματος επιστρέφει στο κυρίως πρόγραμμα όπου όμως ο πίνακας Y είναι άγνωστος. Δεν πρέπει να διαφεύγει όμως το γεγονός ότι η υπορουτίνα κλήθηκε από το κυρίως πρόγραμμα με το όνομα του πίνακα X. Έτσι λοιπόν οι νέες τιμές του πίνακα Y της υπορουτίνας μεταφέρονται στον πίνακα X του κυρίως προγράμματος.

Ο τρόπος δήλωσης μεταβλητής διάστασης στον πίνακα Y στην υπορουτίνα, καθιστά την υπορουτίνα, όπως είναι γραμμένη, γενικώτερης χρήσης και μπορεί να κληθεί από οποιοδήποτε πρόγραμμα. Η διάσταση του πίνακα μπορεί να είναι οποιαδήποτε, μέσα στα όρια βέβαια της μνήμης του υπολογιστή που χρησιμοποιείται.

Παράδειγμα

Ζητείται να υπολογιστεί το γινόμενο δύο πινάκων διάστασης 3x3.

```
      PROGRAM TEST25
      DIMENSTION A(3,3),B(3,3),C(3,3)
      DO 5 I=1,3
      DO 5 J=1,3
      A(I,J)=0.
      B(I,J)=0.
5     C(I,J)=0.
      WRITE (*,6)
6     FORMAT('ENTER THE ELEMENTS OF THE TWO MATRICES')
      DO 8 I=1,3
      DO 8 J=1,3
      READ(*,*)A(I,J),B(I,J)
8     CONTINUE
      CALL SMUL (A,B,C,3)
      DO 10 I=1,3
      WRITE(*,12)(C(I,J),J=1,3)
10    CONTINUE
12    FORMAT(3(2X,F8.0))
      STOP
      END
      SUBROUTINE SMUL(X,Y,Z,M)
      DIMENSION X(M,M),Y(M,M),Z(M,M)
      DO 1 I=1,M
      DO 2 J=1,M
      DO 3 K=1,M
      Z(I,J)=Z(I,J)+X(I,K)*Y(K,J)
3     CONTINUE
2     CONTINUE
1     CONTINUE
      RETURN
      END
```

Το πρόγραμμα διαβάζει τα στοιχεία των πινάκων A και B και καταχωρεί στον πίνακα C το αποτέλεσμα του γινομένου των δύο πινάκων. Το όνομα των πινάκων A, B, C, έχει μεταφερθεί στην υπορουτίνα με τα ονόματα X, Y, Z. Η μεταβλητή

διάσταση των πινάκων X, Y, Z στην υπορουτίνα, την καθιστά ικανή να χρησιμοποιηθεί σε οποιαδήποτε περίπτωση πολλαπλασιασμού πινάκων τάξης nxn, αφού προηγουμένως εξασφαλισθεί ο μηδενισμός των στοιχείων του πίνακα Z(M,M), πρίν απο την κλήση της υπορουτίνας.

Κάθε υπορουτίνα θεωρείται σαν ξεχωριστό υποπρόγραμμα και με την έννοια αυτή μπορούν να υπάρχουν ίδια ονόματα παραμέτρων όπως και στο κυρίως πρόγραμμα ή άλλα υποπρογράμματα, όπως επίσης και ίδιοι αριθμοί αναφοράς εντολών. Ένα πρόγραμμα μπορεί να αποτελείται από πολλές υπορουτίνες, οι οποίες είναι δυνατόν να καλούνται από το κυρίως πρόγραμμα ή και από υπορουτίνα ή υπορουτίνες. Εκτυπώσεις τιμών παραμέτρων μπορούν να γίνονται από το κυρίως πρόγραμμα ή από υπορουτίνες, εφ' όσον βέβαια οι τιμές των παραμέτρων είναι γνωστές στην υπορουτίνα.

Παράδειγμα: Γράψτε πρόγραμμα σε γλώσσα FORTRAN, το οποίο να υπολογίζει το εμβαδόν ενός τριγώνου, όταν είναι γνωστά τα μήκη των πλευρών του.

```
      PROGRAM TRIG
      WRITE(*,10)
10    FORMAT ('YPOLOGISMOS EMBADOY TRIGONOY'/,
     >' DOSE TA MHKH TWN PLEYRWN A,B,C')
      READ(*,*) A,B,C,
      CALL EMV(A,B,C,S)
      WRITE(*,20)A,B,C,S
20    FORMAT('PLEYRES A=',F12.3,'B=',F12.3,'C=',F12.3/,
     >' EMVADON=',F12.3)
      STOP
      END
      SUBROUTINE EMV(X,Y,Z,T)
      P=(X+Y+Z)/2.
      T=SQRT(P*(P-X)*(P-Y)*(P-Z))
      RETURN
      END
```

Ανακεφαλαιώνοντας τα σχετικά με τις υπορουτίνες σημειώνουμε τα ακόλουθα:

1. Κάθε υποπρόγραμμα υπορουτίνα (SUBROUTINE) έχει σαν πρώτη εντολή την SUBROUTINE SNAME, όπου SNAME το όνομα της υπορουτίνας και τελειώνουν πάντα με τις εντολές RETURN και END.
2. Η γενική μορφή της πρώτης εντολής είναι

SUBROUTINE SNAME (VAR1,VAR2,....VARN)

όπου VAR1,VAR2,....VARN είναι ονόματα μεταβλητών ή πινάκων. Η εντολή αυτή ονομάζεται εντολή δήλωσης ή εντολή ορισμού της υπορουτίνας.

3. Η κλήση της υπορουτίνας, από κυρίως πρόγραμμα ή από υποπρόγραμμα γίνεται με την εντολή CALL SNAME(VAR1, VAR2,...VARN), η οποία ονομάζεται εντολή κλήσης της υπορουτίνας. Θα πρέπει να δίνεται ιδιαίτερη προσοχή στον αριθμό και τη σειρά των παραμέτρων κλήσης και ορισμού καθ' όσον υπάρχει μία προς μία αντιστοιχία. Ενώ οι παράμετροι κλήσης μπορεί να είναι και αριθμητικές εκφράσεις, οι παράμετροι ορισμού συνήθως είναι μόνον ονόματα μεταβλητών ή πινάκων.

4. Μερικές από τις παραμέτρους κλήσης και ορισμού μπορούν να είναι παράμετροι εισόδου.

12.2 ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ ΤΥΠΟΥ ΣΥΝΑΡΤΗΣΗΣ FUNCTION SUBPROGRAMS

Τα υποπρογράμματα συνάρτησης (FUNCTION) είναι απλούστερη μορφή των υπορουτινών. Τα υποπρογράμματα συναρτήσεις αποτελούνται από ένα σύνολο εντολών FORTRAN και οι οποίες καλούνται από οποιοδήποτε σημείο του κυρίως προγράμματος ή από υποπρογράμματα. Τα υποπρογράμματα συνάρτησης χρησιμοποιούνται συνήθως όταν από την FUNCTION επιστρέφεται η τιμή μίας μόνον μεταβλητής. Μερικές φορές είναι δυνατόν να επιστρέψουν περισσότερες από μια τιμές μεταβλητών.

Το προηγούμενο παράδειγμα που υπολογίζει το εμβαδόν τριγώνου, όταν είναι γνωστά τα μήκη των πλευρών του μετασχηματίζεται με χρήση FUNCTION ως εξής:

```
      PROGRAM TRIG
      WRITE(*,5)
5     FORMAT ('YPOLOGISMOS EMVADOY TRIGWNOY',/,
     >' DOSE TA MHKH TWN PLEYRWN A,B,C')
      READ(*,*)A,B,C
      S=EMV(A,B,C)
      WRITE(*,10)A,B,C,S
10    FORMAT('PLEYRES A=',F12.3,'B=',F12.3,'C=',F12.3,/,
     >' EMVADON=',F12.3)
      STOP
      END
```

```
      FUNCTION EMV(X,Y,Z)
      P=(X+Y+Z)/2.
      EMV=SQRT(P*(P-X)*(P-Y)*(P-Z))
      RETURN
      END
```

Σημειώνεται ότι το όνομα της FUNCTION (στην προκειμένη περίπτωση EMV) εμφανίζεται και σαν όνομα μεταβλητής μέσα στην FUNCTION και επιστρέφει μια τιμή στο σημείο κλήσης από το κυρίως πρόγραμμα, που είναι η τιμή της μεταβλητής EMV, και η οποία υπολογίστηκε μέσα στο υποπρόγραμμα FUNCTION. Η εμφάνιση του ονόματος της FUNCTION είναι υποχρεωτική για τα υποπρογράμματα τύπου function, ενώ αυτό δεν μπορεί να συμβαίνει για υποπρογράμματα τύπου SUBROUTINE.

Το πρώτο γράμμα του ονόματος της FUNCTION, ορίζει αν η τιμή της FUNCTION θα είναι ακέραια (αν είναι ένα από τα γράμματα I, J, K, L, M, N) ή πραγματική (αν αρχίζει με ένα από τα λοιπά γράμματα).

Παράδειγμα:

Να υπολογιστούν οι τιμές της συνάρτησης $Y=3X^2-2Z$ για τις ακόλουθες τιμές των X και Z

X=1, 2, 3, 4, 5 και Z=2, 4, 6, 8, 10

Τα αποτελέσματα να τυπώνονται από το κυρίως πρόγραμμα.

```
      PROGRAM PAR2
      DO 10 I=1,5
      DO 20 J=2,10,2
      Y=F(I,J)
      WRITE (*,30)I,J,F
20    CONTINUE
10    CONTINUE
      STOP
      END
      FUNCTION F(K,L)
      F=3.*K*K-2.*L
      RETURN
      END
```

Το πρόγραμμα είναι απλό και δεν χρειάζονται επεξηγήσεις. Θα πρέπει μόνο να τονιστεί το γεγονός ότι ενώ είναι απαραίτητο να μήν αλλάζει το όνομα της

συνάρτησης, δεν είναι απαραίτητο να είναι τα ίδια τα ονόματα των παραμέτρων ορισμού και κλήσης της FUNCTION. Στο παράδειγμα η FUNCTION καλείται με τις μεταβλητές I,J ενώ στον ορισμό της έχουν αντικατασταθεί με τα K και L. Είναι προφανές ότι τα I και J δεν υπάρχουν ως μεταβλητές στο υποπρόγραμμα FUNCTION και αν ήταν απαραίτητο θα μπορούσαν να χρησιμοποιηθούν μέσα στην FUNCTION, ως ονόματα μεταβλητών, χωρίς να δημιουργηθεί πρόβλημα. Σε μιά τέτοια περίπτωση, οι τιμές των μεταβλητών αυτών μέσα στο υποπρόγραμμα είναι τελείως άσχετες και ανεξάρτητες απο τις τιμές των ίδιων μεταβλητών άλλων υποπρογραμμάτων ή του κυρίως προγράμματος. Γενικά στα υποπρογράμματα, τα ονόματα των μεταβλητών που χρησιμοποιούνται είναι τοπικά και δεν έχουν καμμία σχέση με τα ονόματα ίδιων μεταβλητών που χρησιμοποιούνται στο κυρίως πρόγραμμα ή άλλα υποπρογράμματα. Το ίδιο ισχύει βέβαια και για τους αριθμούς αναφοράς εντολών.

Το κυρίως πρόγραμμα μπορεί να περιλαμβάνει εκφράσεις όπως

```
DIMENSION A(100), B(50)
.
.
SSS=F(A,B,...)
.
.
```

Στην περίπτωση αυτή η FUNCTION θα αρχίζει με την εντολή ορισμού της

FUNCTION SSS(A,B,...), (ή FUNCTION SSS(X,Y,...))

οπότε θα πρέπει να ακολουθεί η δήλωση.

DIMENSION A(100), B(50) (ή DIMENSION X(100),Y(50))

έτσι ώστε οι διαστάσεις των πινάκων να είναι ίδιες με αυτές που ορίστηκαν στο κυρίως πρόγραμμα και ακόμη να είναι του ίδιου τύπου (ακέραιες ή πραγματικές) σε πλήρη αντιστοιχία.

Ανακεφαλαιώνοντας τα σημαντικώτερα σημεία αναφέρονται:

1. Κάθε υποπρόγραμμα FUNCTION πρέπει να έχει τουλάχιστον μια παράμετρο στην παρένθεση που ακολουθεί το όνομα της FUNCTION.
2. Η κλήση της FUNCTION γίνεται αυτόματα με την αναφορά του ονόματός της και με τις κατάλληλες παραμέτρους. Η ροή του προγράμματος μεταφέρεται στην FUNCTION, η οποία αφού ικανοποιηθεί και αντιστοιχηθεί μία τιμή στο όνομά της, μεταφέρει τη ροή του προγράμματος στο σημείο όπου εκλήθη η FUNCTION. Τονίζεται και πάλι οτι ο τρόπος κλήσης ενός υποπρογράμματος FUNCTION είναι διαφορετικός απο αυτόν ενός υποπρογράμματος SUBROUTINE.

3. Συνήθως τα υποπρογράμματα FUNCTION χρησιμοποιούνται στις περιπτώσεις που ζητείται να επιστραφεί η τιμή μιας μεταβλητής, που αντιστοιχεί στο όνομα της FUNCTION.

Παράδειγμα

Να υπολογιστούν οι συνδυασμοί N αντικειμένων ανά K

$$\binom{N}{K} = \frac{N!}{K!(N-K)!}$$

```
      PROGRAM TTWOF
      WRITE(*,5)
5     FORMAT('ENTER TWO INTEGERS N AND K, N>K')
      READ(*,6)N,K
6     FORMAT (2(I3))
      CB=CO(N,K)
      WRITE(*,10)CB
10    FORMAT('COMBINATIONS=',F10.1)
      STOP
      END
      FUNCTION CO(N,K)
      X=FC(N)
      Y=FC(K)
      I=N-K
      Z=FC(I)
      CO=X/(Y*Z)
      RETURN
      END
      FUNCTION FC(J)
      IF(J.LT.0)GOTO 5
      FC=1.
      IF(J.EQ.0)GOTO 10
      DO 20 I=1,J
      FC=FC*I
20    CONTINUE
      GO TO 10
5     CONTINUE
      WRITE(*,*)'NEGATIVE ARGUMENT'
```

```
10      CONTINUE
        RETURN
        END
```

Στο παράδειγμα αυτό φαίνεται πώς γίνεται η κλήση μιας FUNCTION από μια άλλη FUNCTION.

12.3 ΕΝΤΟΛΗ ΣΥΝΑΡΤΗΣΗΣ STATEMENT FUNCTION

Στις περιπτώσεις που ένα υποπρόγραμμα είναι πολύ απλό και μπορεί να εκφραστεί με μία εντολή, είναι δυνατόν η συνάρτηση-εντολή να γραφεί στο κυρίως πρόγραμμα, πριν από την πρώτη εκτελέσιμη εντολή χωρίς να χρειάζεται υποπρόγραμμα SUBROUTINE ή FUNCTION.
Το παράδειγμα του υπολογισμού του εμβαδού ενός τριγώνου, όταν είναι γνωστά τα μήκη των πλευρών μετασχηματίζεται ως εξής:

```
        PROGRAM TRIGSF
        EMV(X,Y,Z)=SQRT(P*(P-X)*(P-Y)*(P-Z))
        WRITE(*,5)
5       FORMAT ('YPOLOGISMOS EMVADOY TRIGWNOY',/,
       >' DOSE TA MHKH TWN PLEYRWN A,B,C')
        READ(*,*)A,B,C
        P=(A+B+C)/2.
        S=EMV(X,Y,Z)
        WRITE(*,10)A,B,C,S
10      FORMAT('PLEYRES A=',F12.3,'B=',F12.3,'C=',F12.3,/,
       >' EMVADON=',F12.3)
        STOP
        END
```

Η εντολή

```
        EMV(X,Y,Z)=SQRT(P*(P-X)*(P-Y)*(P-Z))
```

ορίζει την συνάρτηση EMV με τρείς παραμέτρους X, Y, Z. Το πρόγραμμα διαβάζει τα μήκη των τριών πλευρών του τριγώνου και υπολογίζει την ημιπερίμετρο P. Η επόμενη εντολή καλεί την Statement FUNCTION EMV, η οποία υπολογίζει το εμβαδόν.

Το πρώτο γράμμα του ονόματος της εντολής συνάρτησης ορίζει τον τύπο του αποτελέσματος. Αν είναι ένα απο τα γράμματα I, J, K, L, M, N τότε το αποτέλεσμα θα είναι ακέραιος, σε διαφορετική περίπτωση θα είναι πραγματικός. Στα ονόματα

των παραμέτρων που βρίσκονται μέσα στην παρένθεση μετά το όνομα της συνάρτησης, μπορεί να βρίσκεται οποιοδήποτε όνομα μεταβλητής, ιδίου τύπου όμως με αυτό που πρόκειται να χρησιμοποιηθεί. Δεν επιτρέπεται όμως να υπάρχουν ονόματα μεταβλητών τα οποία έχουν δηλωθεί ως πίνακες.

Είναι προφανές οτι μια εντολή συνάρτησης έχει έννοια μόνον μέσα στο πρόγραμμα ή υποπρόγραμμα στο οποίο έχει οριστεί και όχι σε οποιοδήποτε άλλο υποπρόγραμμα έστω και του ιδίου προγράμματος.

ΑΛΛΕΣ ΙΔΙΟΤΗΤΕΣ ΤΩΝ ΥΠΟΠΡΟΓΡΑΜΜΑΤΩΝ

Στα προηγούμενα αναφέρθηκαν τα βασικά σημεία της δομής των υποπρογραμμάτων. Υπάρχουν κάποιες εντολές ακόμη οι οποίες δίνουν μερικές πρόσθετες δυνατότητες στην χρήση υποπρογραμμάτων στην FORTRAN.

Όπως αναφέρθηκε ήδη, το πρώτο γράμμα του ονόματος μιάς εντολής συνάρτησης (Statement FUNCTION) ή μιάς FUNCTION δηλώνει τον τύπο του αποτελέσματος. Υπάρχει η δυνατότητα να αλλάξει ο τύπος του αποτελέσματος, χρησιμοποιώντας τις εντολές REAL και INTEGER όπως και στα ονόματα μεταβλητών.

Έτσι οι εντολές

```
REAL FUNCTION IAPL(X,Y,Z)
REAL FUNCTION KVAX(A,B)
INTEGER FUNCTION COUP(A,B,C,D)
INTEGER FUNCTION WMAX(A,C)
```

αλλάζουν τον τύπο του αποτελέσματος στον επιθυμητό τύπο.

13. COMMON BLOCKS

Για κάθε μεταβλητή που χρησιμοποιείται σε ένα πρόγραμμα, ο υπολογιστής κρατά μια θέση μνήμης για κάθε μεταβλητή. Το ίδιο συμβαίνει για κάθε στοιχείο ενός πίνακα που δηλώνεται με την εντολή DIMENSION, REAL ή INTEGER. Αν πρόκειται να μεταφερθούν στοιχεία ενός πίνακα ή μεταβλητές σε κάποιο υποπρόγραμμα, τότε πρέπει να δηλωθούν στην κλήση και τη δήλωση του υποπρογράμματος. Αν μεταφέρονται στοιχεία πίνακα, τότε πρέπει να δηλωθεί και η διάσταση του πίνακα με την εντολή DIMENSION. Αυτό σημαίνει ότι τα στοιχεία του ίδιου πίνακα, καταλαμβάνουν δύο (ή περισσότερες) θέσεις το καθένα στη μνήμη του υπολογιστή. Αν πρόκειται για πρόγραμμα μικρών απαιτήσεων σε μνήμη, δεν πρόκειται να υπάρξει πρόβλημα. Στις περιπτώσεις όμως που οι απαιτήσεις σε μνήμη ενός προγράμματος είναι μεγάλες, τότε κάθε δυνατή οικονομία χώρου αποθήκευσης και χρόνου εκτέλεσης είναι σημαντική.

Η γλώσσα FORTRAN δίνει τη δυνατότητα, μέσω της εντολής COMMON,να αποθηκευθούν ορισμένες μεταβλητές σε ένα μέρος της μνήμης, όπου θα έχουν πρόσβαση και θα μπορούν να χρησιμοποιηθούν από το κύριο πρόγραμμα και από οποιοδήποτε άλλο υποπρόγραμμα του προγράμματος. Η κοινή αυτή περιοχή μνήμης του Η/Υ ονομάζεται COMMON BLOCK. Η εντολή COMMON είναι μή εκτελέσιμη εντολή καθορισμού, η οποία πρέπει να εμφανίζεται με τον ίδιο ακριβώς τρόπο στο κυρίως πρόγραμμα και σε κάθε υποπρόγραμμα που θα κάνει χρήση των μεταβλητών που ορίζονται στην COMMON. Στα ονόματα των μεταβλητών που εμφανίζονται στην COMMON μπορούν να περιλαμβάνονται και ονόματα πινάκων χωρίς να αναφέρεται η διάσταση του πίνακα, η οποία βέβαια πρέπει να έχει οριστεί προηγουμένως με την εντολή DIMENSION.

Παράδειγμα

Να γραφεί πρόγραμμα FORTRAN το οποίο αφού διαβάζει ένα πλήθος αριθμών, θα υπολογίζει τη μέση τιμή αυτών και κατόπιν θα βρίσκει τον πλησιέστερο, από τους δοθέντες αριθμούς, στη μέση τιμή.

Αφού δεν δίδεται πληροφορία για το πόσοι θα είναι οι αριθμοί, θα πρέπει η διάσταση των πινάκων που θα χρησιμοποιηθούν να είναι μεγαλύτερη από το πιθανό πλήθος των δοθέντων αριθμών. Ας υποτεθεί οτι δεν μπορεί να είναι περισσότεροι απο 1000 αριθμοί. Γίνεται επίσης η υπόθεση ότι οι αριθμοί αυτοί βρίσκονται σε ένα αρχείο και το πρόγραμμα θα ρωτήσει σχετικά με το όνομα του αρχείου.

```
      PROGRAM TESTGO
      CHARACTER*20 INPF
      COMMON A(1000),VM,VMN,NP
      WRITE(*,*)'ENTER INPUT FILENAME'
      READ(*,'(A20)')INPF
      OPEN(10,FILE=INPF)
      DO 5 I=1,1000
      READ(10,*,ERR=100)A(I)
5     CONTINUE
100   NP=I-1
      CALL FMEAN
      WRITE(*,20)NP,VM,VMN
20    FORMAT('TOTAL POINTS=', I5./,
     >' MEAN VALUE=',F12.5,/,
     >' NEAREST TO MEAN VALUE=',F12.5)
      STOP
      END
      SUBROUTINE FMEAN
      COMMON A(1000),VM,VMN,NP
      S=0.
      DO 5 I=1,NP
      S=S+A(I)
5     CONTINUE
      VM=S/NP
      CALL FNMEAN
      RETURN
      END
      SUBROUTINE FNMEAN
      COMMON A(1000),VM,VMN,NP
      VMN=A(I)
      DO 10 I=2,NP
      V1=ABS(A(I)-VM)
      V2=ABS(VMN-VM)
      IF(V1.LT.V2)VMN=X(I)
10    CONTINUE
      RETURN
      END
```

Η διάσταση του πίνακα ορίζεται από την εντολή DIMENSION, ενώ στην εντολή COMMON εμφανίζεται μόνον το όνομα της μεταβλητής Α. Με το DO 5 και την εντολή READ(10,*,ERR=100)A(I) διαβάζονται στοιχεία από το αρχείο INPF με ελεύθερο FORMAT. Αν συμβεί λάθος στην ανάγνωση, το οποίο σίγουρα θα συμβεί όταν η ανάγνωση φθάσει στο τέλος του αρχείου όπου δεν υπάρχει άλλο στοιχείο για να διαβαστεί, τότε η ροή του προγράμματος θα μεταφερθεί στην εντολή με αριθμό αναφοράς 100 όπου ορίζεται η τιμή της μεταβλητής NP. Εφ' όσον διακόπτεται με αυτόν τον τρόπο το DO LOOP, πριν ολοκληρωθεί, η τιμή του I είναι ορισμένη. Έστω ίση με Ν. Τα στοιχεία που έχουν διαβαστεί όμως είναι Ν-1. Έτσι το σύνολο των στοιχείων είναι NP=I-1. Με την CALL FMEAN η ροή του προγράμματος μεταφέρεται στην υπορουτίνα FMEAN. Δεν χρειάζεται η αναγραφή των παραμέτρων στην προκειμένη περίπτωση διότι οι παράμετροι μεταφέρονται από και προς την υπορουτίνα FMEAN μέσω της COMMON. Μέσα από την υπορουτίνα FMEAN η ροή μεταφέρεται στην υπορουτίνα FNMEAN, όπου και πάλι οι παράμετροι μεταφέρονται μέσω της COMMON. Στην υπορουτίνα FNMEAN γίνεται σύγκριση της διαφοράς κάθε αριθμού από τη μέση τιμή. Δεν ενδιαφέρει αν είναι μεγαλύτερη ή μικρότερη από τη μέση τιμή, αλλά απλώς η πλησιέστερη και γι' αυτό χρησιμοποιείται η ABS (απόλυτη τιμή) της διαφοράς.

Είναι δυνατόν, αλλά γενικά θα πρέπει να αποφεύγεται, να αλλάξουν τα ονόματα των μεταβλητών στο COMMON μιας υπορουτίνας, αρκεί να είναι του ίδιου τύπου, όπως στο κυρίως πρόγραμμα. Στο προηγούμενο παράδειγμα, θα μπορούσε στην SUBROUTINE FMEAN να γραφεί COMMON X,Y,Z,K με τις ανάλογες αλλαγές βέβαια και στις λοιπές εντολές της υπορουτίνας. Έτσι θα υπάρχει μία προς μία αντιστοιχία με τις μεταβλητές του COMMON στο κυρίως πρόγραμμα. Η μεταβλητή Χ αντιστοιχεί στον πίνακα Α, η Υ στην VM, η Ζ στην VMN και η Κ στην NP. Οι μεταβλητές αυτές χρησιμοποιούν τις ίδιες ακριβώς θέσεις μνήμης που κατελήφθησαν από τη δήλωση των μεταβλητών στο COMMON του κυρίως προγράμματος.

Η παρουσία του COMMON δεν σημαίνει ότι οπωσδήποτε δεν θα υπάρχουν μεταβλητές που θα συνοδεύουν την κλήση μιας υπορουτίνας. Ο προγραμματιστής έχει τη δυνατότητα μερικές μεταβλητές να τις εντάξει σε ένα ή περισσότερα COMMON και άλλες να τις συμπεριλάβει στις παραμέτρους κλήσης και ορισμού μιας υπορουτίνας. Στις περιπτώσεις αυτές θα πρέπει να δίνεται ιδιαίτερη προσοχή ώστε μια μεταβλητή που βρίσκεται στις παραμέτρους κλήσης να μην υπάρχει και στις μεταβλητές του COMMON.

Υπάρχει επίσης η δυνατότητα να οριστούν περισσότερα του ενός COMMON BLOCKS. Στις περιπτώσεις αυτές πρέπει να δοθεί ένα όνομα σε κάθε COMMON

BLOCK έτσι ώστε να γίνεται ο διαχωρισμός του από τα υπόλοιπα. Η γενική μορφή σύνταξης είναι η εξής

COMMON/NAME1/VAR1,VARW.....VARN

COMMON/NAME2/PAR1,PAR2,....,PARN

Ο παραπάνω τρόπος δήλωσης των COMMON BLOCK λέγεται LABELLED COMMON. Στον τρόπο αυτό οι μεταβλητές μπορούν να είναι πίνακες, των οποίων η διάσταση ορίζεται στο COMMON και όχι στο DIMENSION. π.χ.

COMMON/ONE/A(100),B,C,D,E

δηλώνεται ότι το COMMON BLOCK με το όνομα ONE καταλαμβάνει 104 θέσεις μνήμης, 100 για τα στοιχεία του πίνακα A και τέσσερις για τις μεταβλητές B,C,D,E. Ένα ή περισσότερα LABELLED COMMON μπορούν να δηλωθούν σε μία ή περισσότερες υπορουτίνες, από όπου οι μεταβλητές αυτές αναφέρονται στις ίδιες θέσεις μνήμης.

Με τη χρήση των COMMON BLOCKS δεν επιτυγχάνεται μόνον οικονομία στις θέσεις μνήμης, πράγμα πολύ σημαντικό σε περιπτώσεις που η διαθέσιμη μνήμη είναι περιορισμένη ή οι απαιτήσεις του προγράμματος είναι μεγάλες, αλλά επιτυγχάνεται και αύξηση της ταχύτητας εκτέλεσης του προγράμματος. Αυτή η αύξηση της ταχύτητας οφείλεται στο γεγονός ότι οι μεταβιβαζόμενες τιμές παραμέτρων από ένα πρόγραμμα ή υποπρόγραμμα σε ένα άλλο, γίνεται χωρίς αλλαγή ονόματος των παραμέτρων.

14. ΑΛΛΕΣ ΕΝΤΟΛΕΣ

14.1 ΕΝΤΟΛΗ EQUIVALENCE

Όπως έχει αναφερθεί μέχρι τώρα, οι τιμές των μεταβλητών είναι ανεξάρτητες μεταξύ των. Αυτό σημαίνει ότι κάθε μεταβλητή έχει τη δική της θέση στη μνήμη του υπολογιστή, ενώ η ανάθεση τιμής σε μια μεταβλητή, δεν επηρρεάζει τις τιμές άλλων μεταβλητών.

Σε πολλά μεγάλα προγράμματα μια μεταβλητή χρησιμοποιείται σε συγκεκριμένο μέρος του προγράμματος, κάποια άλλη σε άλλο κλπ. Ας υποτεθεί ότι μια μεταβλητή Α χρησιμοποιείται μέχρι το μέσον ενός προγράμματος και από εκεί και κάτω χρησιμοποιείται μια άλλη άλλη μεταβλητή , έστω η Β. Οι μεταβλητές Α και Β έχουν διαφορετικό νόημα, διαφορετικές τιμές και βέβαια καταλαμβάνουν διαφορετικές θέσεις μνήμης. Εφ' όσον η λογική του προγράμματος είναι τέτοια που να μήν χρησιμοποιούνται συγχρόνως οι μεταβλητές Α και Β, είναι δυνατόν να χρησιμοποιηθεί μόνον μία μεταβλητή, π.χ. η Α, σε όλο το πρόγραμμα, χωρίς να επηρεαστούν τα αποτελέσματα. Αυτό επιτυγχάνεται χωρίς να γίνει καμμία αλλαγή στο πρόγραμμα, που σημαίνει ότι υπάρχουν ακόμη ως ονόματα οι μεταβλητές Α και Β, χρησιμοποιώντας την εντολή

EQUIVALENCE (A,B)

στην αρχή του προγράμματος και μετά τις εντολές PROGRAM, DIMENSION, CHARACTER και COMMON.

Η εντολή EQUIVALENCE(A,B) δηλώνει ότι οι μεταβλητές Α και Β είναι ισοδύναμες και εννοούν το ίδιο πράγμα, καθώς επίσης και ότι καταλαμβάνουν την ίδια θέση μνήμης. Με την εντολή EQUIVALCE μπορεί να δηλωθεί ισοδυναμία όχι μόνον ζευγών, αλλά και τριάδων, τετράδων κλπ. Μπορούν δε να δηλωθούν πολλές ομάδες ισοδυναμίας (EQUIVALENCE) σε ένα πρόγραμμα αν αυτό διευκολύνει τον προγραμματιστή.

14.2 Η ΕΝΤΟΛΗ DATA

Σε πολλές περιπτώσεις χρειάζεται να ανατεθούν αρχικές τιμές ή σταθερές τιμές σε κάποιες μεταβλητές. Μέχρι τώρα αναφέρθηκε ο τρόπος ανάγνωσης τιμών από το τερματικό ή από κάποιο αρχείο. Είναι δυνατόν η ανάθεση αρχικών τιμών σε παραμέτρους να γίνει μέσα από το πρόγραμμα με την εντολή DATA της οποίας η γενική μορφή είναι

DATA PAR1/VALUE1,PAR2/VALUE2,..... ή

DATA PAR1,PAR2,PAR3..../VALUE1,VALUE2,VALUE3,....

όπου PAR1,PAR2,..... είναι ονόματα μεταβλητών και VALUE1,VALUE2,.... είναι οι τιμές που θα ανατεθούν στις μεταβλητές. Η μεταβλητή PAR1 θα πάρει την τιμή VALUE1, η PAR2 την τιμή VALUE2 κλπ. Είναι δυνατόν να υπάρχουν πολλές εντολές DATA σε ένα πρόγραμμα και οι οποίες πρέπει να αναγράφονται στην αρχή του προγράμματος ή υποπρογράμματος, μετά τις εντολές PROGRAM, DIMENSION, CHARACTER, και COMMON.

Με την εντολή DATA είναι εύκολο να ανατεθούν τιμές στα στοιχεία ενός πίνακα. Οι εντολές

```
DIMENSION A(100)
DATA A/50*0., 20*1., 20*2., 10*3/
```

αναθέτουν στα πρώτα 50 στοιχεία του πίνακα Α την τιμή 0., στα επόμενα 20 την τιμή 1., στα επόμενα 20 την τιμή 2. και στα τελευταία 10 την τιμή 3.

Με την εντολή DATA μπορούν να ανατεθούν χαρακτήρες σε μία μεταβλητή τύπου CHARACTER. Π.χ. η εντολή

```
CHARACTER*20 VAR
DATA VAR/ABCDEFGHIJKLMNOPQRST/
```

αναθέτει τα γράμματα A-T στη μεταβλητή VAR.

Οι εντολές

```
DIMENSION A(20)
DATA A(5)/10*0./
```

αναθέτουν σε δέκα (10) στοιχεία του πίνακα Α από Α(5) έως και Α(14) την τιμή 0. Τα υπόλοιπα στοιχεία του Α είναι απροσδιόριστα.

ΜΕΡΙΚΕΣ ΑΚΟΜΗ ΕΣΩΤΕΡΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Η γλώσσα FORTRAN είναι εξοπλισμένη με ένα σύνολο συναρτήσεων, τις οποίες μπορεί άμεσα να χρησιμοποιήσει ο χρήστης. Έχουν αναφερθεί σε προηγούμενα παραδείγματα μερικές όπως η SQRT, EXP κλπ.

Επειδή κάθε έκδοση (version) μιας γλώσσας έχει διαφορές, ιδιαίτερα αν ο κατασκευαστής διαφέρει, είναι σημαντικό για τον προγραμματιστή να ανατρέχει στα εγχειρίδια της γλώσσας που πρόκειται να χρησιμοποιήσει για το σωστό τρόπο κλήσης των εσωτερικών συναρτήσεων.

Ο προγραμματιστής θα πρέπει να είναι ιδιαίτερα προσεκτικός κατά τη χρήση τριγωνομετρικών συναρτήσεων (SIN, COS κλπ) διότι η γωνία θα πρέπει να δίνεται σε ακτίνια και όχι σε μοίρες. Αντίστοιχα η έξοδος του arctan δίνεται σε ακτίνια και όχι σε μοίρες. Θα πρέπει λοιπόν ο προγραμματιστής να κάνει τον απαραίτητο μετασχηματισμό από μοίρες σε ακτίνια και αντίστροφα, κατά περίπτωση.

Ο προγραμματιστής θα πρέπει να αποφεύγει γενικά τη χρησιμοποίηση ονομάτων που είναι ίδια με τα ονόματα των εσωτερικών συναρτήσεων του συστήματος. Αν και πολλές φορές το όνομα που έχει δώσει ο χρήστης έχει προτεραιότητα απέναντι στις εσωτερικές συναρτήσεις, αυτό δεν είναι κανόνας με συνέπεια το πρόγραμμα να δίνει λάθος αποτελέσματα, χωρίς καμμία προειδοποίηση προς τον χρήστη. Επειδή ο εντοπισμός τέτοιου είδους λαθών είναι πάρα πολύ δύσκολος, θα πρέπει να τηρείται αυστηρά ο κανόνας της αποφυγής ονόματος που είναι ίδιος με κάποιο όνομα εσωτερικής συνάρτησης του συστήματος. Οι εντολές INTRINSIC και EXTERNAL μπορούν να περιορίσουν τα παραπάνω προβλήματα, χωρίς όμως να τα εξαλείφουν τελείως.

Το τόξο μιάς τριγωνομετρικής ποσότητας υπολογίζεται χρησιμοποιώντας μία απο τις παρακάτω εσωτερικές συναρτήσεις του συστήματος.

ASIN(X) (Τόξο ημιτόνου X)

ACOS(X) (Τόξο συνημιτόνου X)

ATAN(X) (Τόξο εφαπτομένης X)

Στην περίπτωση που η γωνία εκφράζεται σε μοίρες χρησιμοποιούνται οι παρακάτω συναρτήσεις

SIND(X) (Ημίτονο της γωνίας X, όταν το X εκφράζεται σε μοίρες)

COSD(X) (Συνημίτονο της γωνίας X, όταν το X εκφράζεται σε μοίρες)

TAND(X) (Εφαπτομένη της γωνίας X, όταν το X εκφράζεται σε μοίρες)

ASIND(X) (Τόξο ημιτόνου της γωνίας X, όταν το X εκφράζεται σε μοίρες)

ACOSD(X) (Τόξο συνημιτόνου της γωνίας X, όταν το X εκφράζεται σε μοίρες)

ATAND(X) (Τόξο εφαπτομένης της γωνίας X, όταν το X εκφράζεται σε μοίρες)

14.3 Η ΕΝΤΟΛΗ EXTERNAL

Η εντολή EXTERNAL είναι μια μή εκτελέσιμη εντολή καθορισμού και ως εκ τούτου τοποθετείται στην αρχή του προγράμματος.

Η γενική μορφή σύνταξής της είναι

EXTERNAL FNAM1,FNAM2,.....

όπου FNAM1,FNAM2,..... είναι ονόματα υποπρογραμμάτων και γράφεται μόνον στο πρόγραμμα (ή υποπρόγραμμα) που καλεί το υποπρόγραμμα FNAM. Η διαδικασία κλήσης με χρησιμοποίηση της EXTERNAL γίνεται κατανοητή με το παράδειγμα

.

.

EXTERNAL THETA

.

```
.
A=FUNF(B.C.THETA)
.
.
END
FUNCTION FUNF(X,Y,ZI)
.
.
W=ZI(R,S,T)
.
.
END
```

Η εντολή EXTERNAL πληροφορεί τον υπολογιστή ότι υπάρχει ένα υποπρόγραμμα στη συνέχεια, με το όνομα THETA. Όταν η ροή του προγράμματος φθάσει στην εντολή A=FUNF(B,C,THETA), η ροή μεταφέρεται στην FUNCTION FUNF όπου γίνεται η αντιστοίχιση σων παραμέτρων. Η τιμή της B αντιστοιχείται στην X και της C στην Y. Για την THETA δεν αντιστοιχεί τιμή επειδή υπάρχει η EXTERNAL. Έτσι το όνομα THETA μεταβιβάζεται στην ZI και η FUNCTION FUNF εκτελείται έχοντας το όνομα THETA στη θέση του ZI. Όταν η ροή φθάσει στην W=ZI(R,S,T), ο υπολογιστής αναζητά το πρόγραμμα THETA για να εκτελέσει.

14.4 Η ΕΝΤΟΛΗ INTRINSIC

Η εντολή INTRINSIC λειτουργεί όπως και η EXTERNAL, αναφέρεται όμως στις εσωτερικές συναρτήσεις.
Παράδειγμα

```
.
.
INTRINSIC SIN,COS
.
.
.
CALL TESUB(0.5,SIN,X)
CALL TESUB(2.2,COS,Y)
.
.
STOP
```

```
      END
      SUBROUTINE TESUB(A,B,C)
      C=B(A)
      RETURN
      END
```

Με την πρώτη κλήση υπολογίζεται το ημίτονο της τιμής 0.5 και τη δεύτερη το συνημίτονο τις τιμής 2.2.

14.5 Η ΕΝΤΟΛΗ PARAMETER

Η εντολή PARAMETER χρησιμοποιείται για να δώσει τιμές σε μεταβλητές που επιθυμεί ο προγραμματιστής. Οι μεταβλητές που παίρνουν τιμές μέσω της PARAMETER δεν μπορούν να αλλάξουν κατά την εκτέλεση του προγράμματος.

14.6 Η ΕΝΤΟΛΗ IMPLICIT

Η εντολή IMPLICIT χρησιμοποιείται για να οριστεί ο χαρακτήρας για όλες εκείνες τις μεταβλητές που το όνομά τους αρχίζει από συγκεκριμμένο γράμμα ή από μια περιοχή γραμμάτων. Η γενική μορφή σύνταξης είναι

IMPLICIT TYPE(A1,A2,...),TYPE(B1,B2,...)

όπου TYPE μία από τις INTEGER, REAL, COMPLEX LOGICAL, CHARACTER, DOUBLE PRECISION και A1,A2,..... ένα γράμμα ή μια περιοχή γραμμάτων

Οι εντολές

IMPLICIT COMPLEX(A,B,C)

IMPLICIT REAL (A-J)

σημαίνουν ότι: Όλες οι μεταβλητές των οποίων το όνομα αρχίζει από A,B, ή C θα είναι μιγαδικές. Στη δεύτερη πείπτωση όλες οι μεταβλητές των οποίων το όνομα αρχίζει από A έως και του J περιλαμβανομένου θα είναι πραγματικές.

14.7 Η ΕΝΤΟΛΗ COMPLEX

Όπως είναι γνωστό κάθε μιγαδικός αριθμός αποτελείται από το πραγματικό και το φανταστικό μέρος. Η δήλωση των μιγαδικών μεταβλητών γίνεται στην αρχή του προγράμματος με την εντολή

COMPLEX X,Y

η οποία πληροφορεί τον υπολογιστή ότι οι μεταβλητές X και Y παριστάνουν μιγαδικές ποσότητες. Για να δημιουργηθεί ένας μιγαδικός αριθμός, από δύο πραγματικούς, χρησιμοποιείται η συνάρτηση CMPLX

Παράδειγμα

```
COMPLEX X,Y
X=CMPLX(4.3,1.2)
Y=X+CMPLX(7.3,2.3)
```

η μεταβλητή X θα έχει τιμή 4.3+1.2i και η Y θα έχει την τιμή 11.6 +3.5i.

Για να γίνει διαχωρισμός ενός μιγαδικού σε δύο μέρη (πραγματικό και φανταστικό) χρησιμοποιούνται οι συναρτήσεις REAL και AIMAG που δίνουν αντίστοιχα το πραγματικό και φανταστικό μέρος ενός αριθμού. Έτσι στο παράδειγμα

```
COMPLEX X
X=CMPLX(1.5,2.3)
A=REAL(X)
B=AIMAG(X)
```

η A θα έχει την τιμή 1.5 και η B την 2.3.

15. ΜΕΤΑΒΛΗΤΕΣ ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑΣ

Σημαντικά ψηφία ενός αριθμού ονομάζονται όλα τα ψηφία που βρίσκονται μεταξύ του πρώτου και του τελευταίου μή μηδενικού ψηφίου συμπεριλαμβανομένων. Π.χ. ο αριθμός 0000123.40000 έχει 4 σημαντικά ψηφία, ο 00000010000.000020000 έχει 10 και ο 0.0000001000000200 έχει 7 σημαντικά ψηφία.

Κάθε υπολογιστής μπορεί να κρατά και επεξεργάζεται ένα ορισμένο πλήθος σημαντικών ψηφίων κάθε αριθμού. Το πλήθος των σημαντικών ψηφίων εξαρτάται από την κατασκευή κάθε υπολογιστή και στην πραγματικότητα εξαρτάται από το βάθος της μνήμης των. Οι περισσότεροι υπολογιστές αναπαριστούν έναν πραγματικό αριθμό με 7 σημαντικά ψηφία. Στις περιπτώσεις που ο προγραμματιστής απαιτεί μεγαλύτερη ακρίβεια (περισσότερα σημαντικά ψηφία) δηλώνει ποιές μεταβλητές επιθυμεί να έχουν διπλή ακρίβεια με εντολή στην αρχή του προγράμματος

```
DOUBLE PRECISION A,B
```

Έτσι οι μεταβλητές A και B θα έχουν διπλή ακρίβεια.

Για τη δήλωση ενός αριθμού, σε μορφή διπλής ακρίβειας, χρησιμοποιείται η συνάρτηση DBLE.

Παράδειγμα

```
DOUBLE PRECISION A,B
C=2./7.
A=DBLE(2.)/DBLE(7.)
```

B=DBLE(2./7.)

Σε έναν υπολογιστή με ακρίβεια 7 σημαντικών ψηφίων, η τιμή του C θα είναι 0.2857143, η τιμή του A θα είναι 0.28571428..... και η τιμή του B θα είναι 0.2857143. Η τιμή του A προφανώς πλησιάζει περισσότερο το πραγματικό αποτέλεσμα της διαίρεσης. Τονίζεται ιδιαίτερα το αποτέλεσμα της τιμής του B, που δεν είναι ίδιο με αυτό της A, διότι γίνεται πρώτα η διαίρεση δύο αριθμών απλής ακρίβειας και μετά μετατρέπεται σε διπλής ακρίβειας. Το μόνο που γίνεται δηλαδή στην περίπτωση αυτή είναι να προστεθούν μηδενικά στο αποτέλεσμα της διαίρεσης.

Στην περίπτωση που επιθυμείται όλες οι μεταβλητές να γίνουν διπλής ακρίβειας, χωρίς να ξαναγραφεί το πρόγραμμα, είναι δυνατόν να χρησιμοποιηθεί η εντολή

IMPLICIT DOUBLE PRECISION A-Z

Με τον τρόπο αυτό όλες οι μεταβλητές, ακέραιες και πραγματικές, μετατρέπονται σε μεταβλητές διπλής ακρίβειας.

Η μετατροπή ενός αριθμού διπλής ακρίβειας σε απλή, γίνεται με τη χρήση της συνάρτησης SNGL.

16. ΛΟΓΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ

Όλα όσα αναφέρθηκαν μέχρι τώρα, απευθύνονται σε μεταβλητές που αφορούν σε αριθμητικές ποσότητες. Υπάρχουν όμως μερικές περιπτώσεις όπου διευκολύνει η χρησιμοποιήση λογικών ποσοτήτων. Η FORTRAN έχει τη δυνατότητα απεικόνισης λογικών σταθερών και λογικών μεταβλητών.

Οι λογικές σταθερές είναι δύο. Μία φράση είναι "αληθής" ή "ψευδής". Η αναπαράστασή τους στη FORTRAN είναι αντίστοιχα .TRUE. και .FALSE.

Οι λογικές μεταβλητές παριστάνονται με ονόματα όπως και οι αριθμητικές μεταβλητές και υπακούουν στους ίδιους κανόνες όπως και οι αριθμητικές μεταβλητές. Η δήλωση ότι μια ποσότητα είναι λογική γίνεται με την εντολή LOGICAL, στην αρχή του προγράμματος.

Π.χ. η εντολή LOGICAL A,B

δηλώνει ότι οι μεταβλητές A και B είναι λογικές μεταβλητές.

```
LOGICAL A,B
A=.TRUE.
B=.FALSE.
```

Με τις παραπάνω εντολές, η A παίρνει την τιμή "αληθής" και η B την τιμή "ψευδής".

Παράδειγμα

```
LOGICAL A
X=2.
Y=3.
A=(X.LT.5.).OR.(X*Y.GT.2.)
```

Η A θα πάρει την τιμή .TRUE. (αληθής) αν X<5 ή X*Y>2. Διαφορετικά θα πάρει την τιμή .FALSE. .

ΠΑΡΑΔΕΙΓΜΑΤΑ

1. Δίνονται N το πλήθος ζεύγη τιμών X_i, Y_i που προέκυψαν από κάποιο πείραμα. Αν είναι γνωστό ότι τα X_i, και Y_i υπακούουν σε μια γραμμική σχέση, να βρεθεί η καλλίτερη ευθεία που αναπαριστά τα πειραματικά δεδομένα.

Έστω ότι γραμμική σχέση που συνδέει τα X και Y είναι η

$Y=AX+B$

και αυτό που πρέπει να γίνει είναι να προσδιοριστούν οι τιμές των συντελεστών A(κλίση) και B (σταθερά) για τα ζεύγη τιμών X_i, Y_i . Τα A και B δίνονται από τις σχέσεις

$$A=\frac{N\Sigma XY-\Sigma X\Sigma Y}{N\Sigma X^2-(\Sigma X)^2} \quad \text{και} \quad B=\frac{\Sigma X^2\Sigma Y-\Sigma X\Sigma XY}{N\Sigma X^2-(\Sigma X)^2}$$

Οι τιμές των X_i, και Y_i μπορούν να δοθούν από το τερματικό. Αυτό όμως δεν είναι πάντοτε εύκολο να γίνει, καθώς υπάρχει μεγάλη πιθανότητα λάθους, χωρίς δυνατότητα διόρθωσης του. Έστω ότι τα ζεύγη τιμών των X_i, Y_i βρίσκονται γραμμένα ανά δύο σε κάθε γραμμή στο αρχείο με το όνομα INPUT και ότι το πλήθος τους δεν είναι γνωστό αλλά είναι μικρότερο έστω του 100.

```
      PROGRAM LSQFIT
      DIMENSION X(100),Y(100)
      CHARACTER*8 FILEN
      DO 10 I=1,100
      X(I)=0.
100   Y(1)=0.
      WRITE(*,*)'ENTER FILENAME CONTAINING DATA PAIRS'
      READ(*,'(A8)')FILEN
      OPEN(10,FILE=FILEN)
      DO 20 I=1,100
      READ(20,*,ERR=25)X(I),Y(I)
20    CONTINUE
25    N=I-1
      SXY=0.
      SY=0.
      SX2=0.
      DO 30 I=1,N
      SXY=SXY+X(I)*Y(I)
```

```
      SX=SX+X(I)
      SY=SY+Y(I)
      SX2=SX2+X(I)*X(I)
30    CONTINUE
      S2X=SX*SX
      DNOM1=N*SXY-SX*SY
      DNOM2=SX2*SY-SX*SXY
      DEN=S*SX2-S2X
      A=DNOM1/DEN
      B=SNOM2/DEN
      WRITE(*,40)A,B
40    FORMAT('SLOPE A=',F25.5,/,
     >' CONSTANT B=',F15.5)
      STOP
      END
```

2. Να γραφεί ένα πρόγραμμα σε γλώσσα FORTRAN, το οποίο να υπολογίζει πόσοι είναι όλοι οι τετραψήφιοι αριθμοί, του δεκαδικού συστήματος, που δεν περιέχουν το ψηφίο 5.

Ο μικρότερος τετραψήφιος είναι ο 1000 και ο μεγαλύτερος ο 9999. Ένας τετραψήφιος μπορεί να σχηματιστεί με 4 DO LOOPS, όπου το πρώτο θα αρχίζει από I=1 έως 9 με βήμα 1, το δεύτερο, τρίτο και τέταρτο από 0 έως 9. Αν I, J, K, L οι δείκτες των DO LOOPS, τότε ένας τετραψήφιος σχηματίζεται με την πράξη I*1000+J*100+L*10+K. Ορίζεται μια μεταβλητή N με αρχική τιμή 0 η οποία θα αυξάνει κατά 1 κάθε φορά που συναντάται το ψηφίο 5.

```
      PROGRAM FIVES
      N=0
      DO 10 I=1,9
      IF(I.WQ.5)GOTO 10
      DO 20 J=0,9
      IF(J.EQ.5)GO TO 20
      DO 30 K=0,9
      IF(K.EQ.5)GOTO 30
      DO 40 L=0,9
      IF(L.EQ.5)GOTO 40
      N=N+1
```

```
40    CONTINUE
30    CONTINUE
20    CONTINUE
10    CONTINUE
      WRITE(*,50)N
50    FORMAT('THERE ARE',I4,'FOUR DIGIT NUMBERS THAT'/,
     >' THEY DO NOT CONTAIN DIGIT 5').
      STOP
      END
```

3. Ενα πρόβλημα που συναντάται συχνά είναι μία γεννήτρια τυχαίων αριθμών. Στην πραγματικότητα βέβαια η ακολουθία των αριθμών που παράγονται δεν είναι τυχαία, με την έννοια ότι άν ξανατρέξει το ίδιο πρόγραμμα, θα παραχθεί η ίδια τυχαία ακολουθία αριθμών. Ανεξάρτητα απο αυτά το παρακάτω υποπρόγραμμα είναι κατάλληλο για την δημιουργία "τυχαίων αριθμών" στο διάστημα 0 έως και 1 συμπεριλαμβανομένων.

```
      FUNCTION RANDOM()
C
C  RANTOM IS A PORTABLE RANDOM NUMBER GENERATOR IN (0,1)
C
      DIMENSION R(97)
      PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=3.8580247E-6)
      PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=7.4373773E-6)
      PARAMETER (M3=243000,IA3=4561,IC3=51349)
       SAVE
      DATA IFF /0/
       DATA IDUM/-1/
C
      IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
       IFF=1
       IX1=MOD(IC1-IDUM,M1)
       IX1=MOD(IA1*IX1+IC1,M1)
       IX2=MOD(IX1,M2)
       IX1=MOD(IA1*IX1+IC1,M1)
       IX3=MOD(IX1,M3)
       DO 11 J=1,97
        IX1=MOD(IA1*IX1+IC1,M1)
        IX2=MOD(IA2*IX2+IC2,M2)
        R(J)=(IX1+IX2*RM2)*RM1
11     CONTINUE
       IDUM=1
      ENDIF
      IX1=MOD(IA1*IX1+IC1,M1)
      IX2=MOD(IA2*IX2+IC2,M2)
      IX3=MOD(IA3*IX3+IC3,M3)
      J=1+(97*IX3)/M3
      IF(J.GT.97.OR.J.LT.1)PAUSE 'ERROR IN RAN1'
```

```
      RANDOM=R(J)
      R(J)=(IX1+IX2*RM2)*RM1
      END
```

ΠΑΡΑΡΤΗΜΑ

ΕΙΣΑΓΩΓΗ ΣΤΟ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ ΕΡ/ΙΧ (UNIX)

Το λειτουργικό σύστημα θα μπορούσε να θεωρηθεί σαν ο διευθυντής μιας επιχείρησης, ο οποίος αναθέτει εργασίες στο προσωπικό, προγραμματίζει την πορεία της επιχείρησης και γενικά καθορίζει τις δραστηριότητες της επιχείρησης. Το λειτουργικό σύστημα αποτελείται απο ένα σύνολο προγραμμάτων που ελέγχει και ρυθμίζει την λειτουργία ενός ηλεκτρονικού υπολογιστή (Η/Υ), κατανέμει την εργασία που θα αναλάβει να φέρει σε πέρας κάποιο τμήμα, αναλαμβάνει και διεκπεραιώνει την επικοινωνία μεταξύ χρήστη και Η/Υ και παρέχει κάποια εργαλεία για την διευκόλυνση της εργασίας του χρήστη. Για τους μικροϋπολογιστές το πιό διαδεδομένο λειτουργικό σύστημα είναι το DOS. Στους μεγάλους υπολογιστές αντίστοιχα είναι το σύστημα UNIX, του οποίου η αρχική μορφή είναι δημιουργία της εταιρίας AT&T. Όλες οι μεγάλες εταιρίες κατασκευής Η/Υ χρησιμοποιούν σήμερα κατα κύριο λόγο το σύστημα αυτό, με κάποιες μικρές ή μεγάλες προσθήκες, που αποσκοπούν στην διευκόλυνση των χρηστών. Για να είναι δυνατός ο διαχωρισμός των διαφόρων μορφών του λειτουργικού συστήματος UNIX οι διάφορες εταιρείες του δίνουν ένα διαφορετικό όνομα.

Η δομή του λειτουργικού συστήματος UNIX στηρίζεται σε τέσσερα κύρια σημεία:

1. **Kernel** : Αποτελεί τον πυρήνα του λειτουργικού συστήματος. Είναι ένα σύνολο προγραμμάτων, το οποίο σκοπό έχει να συντονίζει την εσωτερική λειτουργία του Η/Υ.
2. **Σύστημα Αρχείων** : Είναι ένα σύνολο προγραμμάτων και εντολών που συντονίζει την αποθήκευση, προσπέλαση και διαχείριση των πληροφοριών.
3. **Shell** : Είναι ένα σύνολο προγραμμάτων που αποτελεί τον σύνδεσμο μεταξύ χρήστη και πυρήνα του Η/Υ και με άμεσο τρόπο ελέγχει και εκτελεί τις εντολές.
4. **Commands** : Είναι στην πραγματικότητα τα ονόματα των προγραμμάτων τα οποία ζητά ο χρήστης να εκτελεστούν απο τον Η/Υ. Ενα σύνολο εντολών ονομάζεται εργαλεία (tools).

ΣΥΝΔΕΣΗ ΜΕ ΤΟ UNIX (ΕΡ/ΙΧ)

Για να συνδεθεί ο χρήστης με τον υπολογιστή πρέπει να έχει διατεθεί στον χρήστη ένα κωδικό όνομα, το οποίο είναι μοναδικό για κάθε χρήστη, και το οποίο πρέπει να έχει δηλωθεί και στον Η/Υ. Απαραίτητο είναι επίσης να έχει διατεθεί ένα password, το οποίο είναι επίσης κωδικό, και το γνωρίζει ο χρήστης και μόνο. Ο κάθε

χρήστης μπορεί οποιαδήποτε στιγμή επιθυμεί να αλλάξει το password και είναι δική του ευθύνη να μην το κοινοποιεί σε άλλους. Δεν είναι δυνατή η σύνδεση ενός χρήστη με εναν υπολογιστή αν δεν δοθούν το κωδικό όνομα και το password που αντιστοιχεί σε αυτόν.

Για να συνδεθεί ο κάθε χρήστης στον Η/Υ πρέπει να έχει στη διάθεσή του ένα τερματικό. Ο Η/Υ μπορεί να συνδεθεί με διάφορους τύπους τερματικών, το καθένα όμως απο αυτά, για να δουλεύει σωστά, πρέπει να έχει δηλωθεί με τον σωστό τρόπο και με τις σωστές παραμέτρους στο ίδιο το τερματικό και συγχρόνως και στον Η/Υ. Για τον λόγο αυτό πρέπει να αποφεύγεται η αλλαγή των παραμέτρων δήλωσης των τερματικών. Η σύνδεση του χρήστη, μέσω του τερματικού, με τον Η/Υ γίνεται ως εξής:

1. Ανοίγετε το τερματικό, και πατάτε το πλήκτρο Return (ή Carriage Return ή Enter). Απο εδώ και στο εξής η εντολή αυτή θα δηλώνετε ως **<CR> ή <cr>**.
2. Στην οθόνη του τερματικού εμφανίζεται το **login**: Στο σημείο αυτό ο χρήστης γράφει το κωδικό του όνομα, χρησιμοποιώντας μικρούς χαρακτήρες (πεζά) και στο τέλος πατά το **<CR>**.
3. Ο Η/Υ σας απαντά με **password** : Ο χρήστης πρέπει να πληκτρολογήσει το προσωπικό του password. Ας σημειωθεί οτι καθώς πληκτρολογούνται οι χαρακτήρες που αποτελούν το password, δεν εμφανίζεται τίποτε στην οθόνη και τούτο για λόγους πρόσθετης ασφάλειας. Όταν πατάτε κάποιο πλήκτρο, φροντίστε να μην το κρατάτε πατημένο για πολύ, διότι θα τυπωθούν πολλοί ίδιοι χαρακτήρες. Μετά την πληκτρολόγηση του password πατήστε το **<CR>**.

Μετά απο αυτά, και εφόσον όλα είναι σωστά, βρίσκεστε μέσα στον κωδικό σας και η απάντηση του υπολογιστή είναι το σήμα του δολλαρίου **$**.

ΑΠΟΣΥΝΔΕΣΗ (logout)

Για να αποσυνδεθείτε απο τον Η/Υ πρέπει να γράψετε **logout <CR>**, ή να πατήσετε το **ctrl-d** (control-d : κρατάτε πατημένο το πλήκτρο ctrl και συγχρόνως πατάτε το d).

ΑΛΛΑΓΗ PASSWORD

Για να αλλάξετε το password πρέπει να γράψετε **passwd** και να ακολουθήσετε τις οδηγίες που εμφανίζονται στην οθόνη. Θα πρέπει όμως να ακολουθήσετε τους εξής κανόνες:

1. Το καινούργιο password πρέπει να περιέχει τουλάχιστον 6 χαρακτήρες, και μέχρι 8 χαρακτήρες μέγιστο. Αν δοθούν παραπάνω απο 8 χαρακτήρες, τότε θα διατηρηθούν μόνον οι 8 πρώτοι.

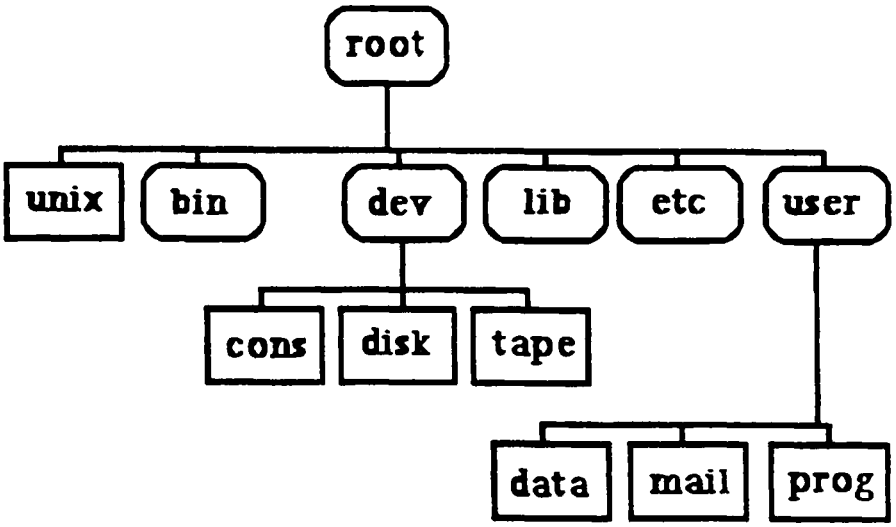
2. Το καινούργιο password πρέπει να περιέχει τουλάχιστον 2 γράμματα του λατινικού αλφαβήτου και έναν τουλάχιστον αριθμητικό ή ειδικό χαρακτήρα. Π.χ. δεκτά password θα ήταν τα ακόλουθα: bj127dsa, bbb22fd, d12345, afs12*%, αλλά όχι τα παρακάτω : 12345678, absafred για προφανείς λόγους.
3. Πρέπει απαραίτητα να διαφέρει απο το login-name (κωδικός).
4. Πρέπει να διαφέρει απο το παλιό σας κατα τουλάχιστον 4 χαρακτήρες.

ΠΡΟΣΟΧΗ:

Τονίζεται οτι στο λειτουργικό σύστημα UNIX τα πεζά και τα κεφαλαία έχουν διαφορετική δράση. Το ίδιο γράμμα όταν είναι πεζό εκτελεί διαφορετική διαδικασία απο όταν είναι κεφαλαίο. Γι'αυτό χρειάζεται ιδιαίτερη προσοχή στη χρήση πεζών και κεφαλαίων χαρακτήρων.

ΣΥΣΤΗΜΑ ΑΡΧΕΙΟΘΕΤΗΣΗΣ

Το λειτουργικό σύστημα UNIX (EP/IX) μοιάζει πολύ με το DOS, τόσο στη δομή του όσο και σε ορισμένες εντολές. Οι ομοιότητες όμως σταματούν εκεί. Το σύστημα αρχειοθέτησης του UNIX έχει τη δομή ανάστροφου δένδρου (tree structure). Στη ρίζα του δένδρου συνδέονται όλοι οι χρήστες που αποτελούν τους κλάδους, οι δε κόμβοι είναι οι φάκελλοι (directories) ή τα αρχεία (files). Μια τυπική σχηματική παράσταση των παραπάνω φαίνεται στο ακόλουθο σχήμα.



Κάθε χρήστης έχει τη δική του σύνδεση με τη ρίζα (root) και τα δικά του directories και files. Δεν μπορεί να επέμβει με κανένα τρόπο στα αρχεία άλλων χρηστών, εκτός αν έχει την άδεια (permission) απο τον χρήστη, στον κωδικό του οποίου βρίσκεται το αρχείο.

Με τον όρο αρχείο (file) εννοείται ένα σύνολο χαρακτήρων, αναγνώσιμων ή μή, που αντιμετωπίζονται όμως σαν σύνολο. Με τον όρο φάκελλος (directory)

εννοείται ένα σύνολο αρχείων. Κάθε χρήστης μπορεί θεωρητικά να έχει μεγάλο αριθμό απο φακέλλους και αρχεία στον κωδικό του. Στην πράξη όμως αυτό ειναι ανεφάρμοστο. Συνιστάται όμως, ιδιαίτερα στους νέους χρήστες, κάθε φάκελλος να περιέχει αρχεία με σχετικό αντικείμενο και με κατάλληλο όνομα που να περιγράφει το περιεχόμενο και να δηλώνεται κάνοντας χρήση κεφαλαίων χαρακτήρων. Τα ονόματα των αρχείων είναι προτιμότερο να είναι με πεζά γράμματα για να ξεχωρίζουν απο τα ονόματα των φακέλλων. Το ίδιο όνομα με διαφορετικό τύπο γραμμάτων (πεζά-κεφαλαία) δηλώνουν διαφορετικά πράγματα (π.χ. source, SOURCE, SOUrce κλπ. δηλώνουν διαφορετικά ονόματα). Γενικά μπορείτε να δώσετε ονόματα σε αρχεία και φακέλλους που να έχουν απο 1 μέχρι 255 χαρακτήρες απο τους οποίους πρέπει να αποφεύγονται οι εξής: space, tab, backspace, ?, @, #, $, ., *, (,), [,], {, }, |, /, <, >. Για την διευκόλυνση των χρηστών όμως καλό θα είναι να αποφεύγονται ονόματα με πολλούς χαρακτήρες.

ΔΗΜΙΟΥΡΓΙΑ ΦΑΚΕΛΛΩΝ ΚΑΙ ΧΡΗΣΗ

Απο τη στιγμή που κάποιος χρήστης δηλώνεται στον Η/Υ αυτόματα δημιουργείται και ο αρχικός του κατάλογος (home directory). Απο εκεί μπορεί να δημιουργήσει καινούρια directories με την εντολή :

$ mkdir DIRNAME <cr>

Το **DIRNAME** αποτελεί το όνομα ενός directory που είναι subdirectory (υποφάκελλος) του τρέχοντος directory.

Μπορεί να γίνει μετακίνηση απο ένα directory σε ένα άλλο που βρίσκεται όμως στον ίδιο δρόμο με το τρέχον και προς τα εμπρός, με την εντολή :

$ cd DIRNAME <cr>

Για να διαπιστωθεί σε πιό directory βρισκόμαστε γράφουμε :

$ pwd <cr> (print working directory)

Για την μετακίνηση κατα ένα directory προς τα πίσω γράφουμε :

$ cd <cr>

Για να δούμε το περιεχόμενο ενός directory χρησιμοποιείται η εντολή ls :

$ ls <cr> η οποία μας δίνει ένα κατάλογο των αρχείων του τρέχοντος directory. Ενώ η εντολή :

$ ls -l <cr> μας δίνει περισσότερες πληροφορίες σχετικά με τα αρχεία του τρέχοντος directory.

Κάθε αρχείο χαρακτηρίζεται απο 10 συγκεκριμμένους χαρακτήρες π.χ.

drw-r-xr-x.

Ο πρώτος δηλώνει τον τύπο του αρχείου και είναι ένας απο τους: **d** (=directory), **-** (=file συνηθισμένο αρχείο), **b** (=block special file), **c** (=character special file)

Οι τρείς επόμενοι χαρακτήρες δηλώνουν τις επιτρεπτές διαδικασίες που έχει ο χρήστης στο αρχείο αυτό. Οι χαρακτήρες μπορούν να είναι : r (=read permission), w (=write permission), x (=execute permission).
Οι επόμενοι τρείς χαρακτήρες δηλώνουν τις επιτρεπτές διαδικασίες που έχουν τα μέλη της ομάδας που ανήκει ο κωδικός, στο αρχείο αυτό.
Οι επόμενοι τρείς χαρακτήρες δηλώνουν τις επιτρεπτές διαδικασίες που έχουν άλλοι χρήστες στο αρχείο αυτό.
Οι διαδικασίες μπορούν να αλλάξουν απο τον δημιουργό του αρχείου ως εξής :
$ chmod w+permission filename <cr>
Όπου w(ho)= u (=user), g (=group), o (=other)
+ ==> παραχώρηση άδειας (permission)
- ==> αναίρεση άδειας (denied permission)
permission = r (=read), w (=write), x (=execute)
Παράδειγμα: Η εντολή chmod g-r filen δηλώνει οτι οι χρήστες της ομάδας που ανήκει ο κωδικός δέν έχουν δυνατότητα να διαβάσουν το περιεχόμενο του αρχείου filen. Αντίστοιχα η εντολή chmod o-r filen δηλώνει οτι οι άλλοι χρήστες δεν έχουν δυνατότητα να διαβάσουν το περιεχόμενο του αρχείου filen.

$ ls -CF <cr>
Δίνει πληροφορίες για το τρέχον directory βάζοντας ενα * στα εκτελέσιμα αρχεία και ένα / στα directories, αν βέβαια υπάρχουν τέτοια.

$ rm filename <cr>
Διαγράφεται το αρχείο με το όνομα filename.

$ rmdir DIRNAME <cr>
Διαγράφεται το directory με το όνομα DIRNAME, εφ'όσον βέβαια είναι κενό απο αρχεία.

ΔΙΑΧΕΙΡΗΣΗ ΑΡΧΕΙΩΝ

Μερικές βασικές εντολές που χρησιμοποιούνται στη διαχείρηση αρχείων είναι :

$ cat filename <cr>
Εμφανίζει το περιεχόμενο του αρχείου στην οθόνη. Η ροή των χαρακτήρων σταματά με την χρήση των **ctrl-s** και συνεχίζει με **crtl-q**.

$ pg filename <cr>

Χρησιμοποιείται για αρχεία μεγάλου μήκους και εμφανίζει στην οθόνη μία σελίδα, το μήκος της οποίας εξαρτάται απο τον τρόπο με τον οποίο έχει δηλωθεί το τερματικό στον Η/Υ.

$ more filename <cr>

Ακριβώς όπως η προηγούμενη εντολή. Πατώντας το πλήκτρο space εμφανίζεται η επόμενη σελίδα ενώ με το **<cr>** εμφανίζεται η επόμενη γραμμή.

$ lpr filename <cr>

Στέλνει το αρχείο για εκτύπωση στον κεντρικό εκτυπωτή.

$ cp filen1 filen2 <cr>

Αντιγράφει το περιεχόμενο του αρχείου filen1 στο αρχείο με όνομα filen2. Αν βρισκεστε σε διαφορετικό directory απο όπου βρίσκεται το αρχείο filen1 τότε πρέπει να δηλώσετε και τον δρόμο που πρέπει να ακολουθηθεί απο τον Η/Υ για να βρεί το αρχείο π.χ.

$ cp /user1/DIR1/SDIR2/filen1 /user1/DIRX/SDIRY/SDIRZ/filen2

Αντιγράφει το περιεχόμενο του αρχείου filen1, το οποίο βρίσκεται στο directory SDIR2, το οποίο είναι subrirectory του DIR1 του user1, στο αρχείο με το όνομα filen2, το οποίο βρίσκεται στο directory SDIRZ, το οποίο είναι subdirecrory του SDIRY και του DIRX του user1.

$ mv filen1 filen2 <cr>

Αλλάζει το όνομα του αρχείου filen1 σε filen2. Το περιεχόμενο του αρχείου filen1 έχει μεταφερθεί στο αρχείο με όνομα filen2. Αν υπάρχει ήδη αρχείο με το όνομα filen2 τότε το παλιό διαγράφεται χωρίς προειδοποίηση και το περιεχόμενο αντικαθίσταται. Το αρχείο filen1 δέν υπάρχει πλέον.

$ rm filename <cr>

Διαγράφει το αρχείο. Η εντολή μπορεί να συνεχιστεί και να γραφούν πολλά ονόματα αρχείων στην ίδια γραμμή. Δεν υπάρχει δυνατότητα ανάκτησης των πληροφοριών που διεγράφησαν.

$ diff filen1 filen2 <cr>

Εμφανίζει στην οθόνη τις διαφορές που υπάρχουν μεταξύ των δύο αρχείων **filen1** και **filen2**.

$ grep string filename <cr>

Εμφανίζει στην οθόνη όλες τις γραμμές του αρχείου filename που περιέχουν την συγκεκριμένη ακολουθία χαρακτήρων που εμφανίζονται στο string. Αν υπάρχουν ειδικοί χαρακτήρες στο string, τότε η ακολουθία των χαρακτήρων του string πρέπει να βρίσκεται μέσα σε εισαγωγικά (').

$ sort filename <cr>

Ταξινομεί αλφαβητικά το αρχείο. Η ταξινόμηση γίνεται με βάση το Αγγλικό αλφάβητο και με το δεκαδικό σύστημα. Δεν είναι δυνατόν να γίνει ταξινόμηση με βάση το Ελληνικό αλφάβητο.

ΕΠΕΞΕΡΓΑΣΤΗΣ ΚΕΙΜΕΝΟΥ vi

Ο συντάκτης (ή επεξεργαστής) κειμένου vi έχει σχεδιαστεί για να χρησιμοποιείται σε τερματικό. Για την σωστή λειτουργία του **vi** πρέπει να έχει δηλωθεί προηγούμενα και με σωστό τρόπο ο τύπος του τερματικού που χρησιμοποιείται.

Ο συντάκτης κειμένου **vi** καλείται με την εντολή **vi filename** και στην οθόνη του τερματικού δημιουργείται ένα "παράθυρο", το οποίο εμφανίζει περίπου 20 γραμμές του αρχείου. Μπορείτε να μετακινηθείτε σε οποιοδήποτε σημείο του παραθύρου για να κάνετε προσθήκες ή άλλες μεταβολλές.

Για να εισαχθούν στοιχεία στο κείμενο, πρέπει να πατηθεί πρώτα το γράμμα **i** (insert). Στο κάτω δεξιά μέρος της οθόνης εμφανίζεται η πληροφορία (**INSERT CHARACTER**) που δηλώνει οτι βρίσκεστε σε κατάσταση εισόδου. Στο σημείο αυτό οποιοδήποτε πλήκτρο και αν πατηθεί, ο αντίστοιχος χαρακτήρας εμφανίζεται στην οθόνη. Η πληροφορία για το τέλος εισαγωγής στοιχείων δηλώνεται με το πλήκτρο **esc(escape)**. Μπορείτε να μετακινηθείτε σε οποιοδήποτε σημείο του κειμένου χρησιμοποιώντας τα τέσσερα βέλη με τα σύμβολα ←, ↑, ↓, → (άν διαθέτει τέτοια σύμβολα το τερματικό), ή με τα πλήκτρα h, j, k, l αντίστοιχα. Υπάρχει πιθανότητα παρ' όλο που το πληκτρολόγιο σας να έχει τα βέλη, να μήν εκτελλούνται οι εντολές μετακίνησης που αναφέρθηκαν. Αυτό μπορεί να οφείλεται στο οτι τα βέλη εκτελούν διαφορετική λειτουργία στο τερματικό σας. Αν συναντήσετε τέτοιου είδους προβλήματα, επικοινωνήστε με το προσωπικό του υπολογιστικού κέντρου το οποίο θα φροντίσει για την επίλυση του προβλήματος. Θα πρέπει να έχετε πάντοτε κατα νού οτι τα πεζά και τα κεφαλαία επιτελούν διαφορετική λειτουργία. Συνοπτικά οι εντολές που χρησιμοποιούνται στον συντάκτη κειμένου vi είναι :

vi filename	Άνοιγμα υπάρχοντος ή δημιουργία νέου αρχείου
i	Δήλωση εισαγωγής χαρακτήρων
esc	Τέλος εισαγωγής χαρακτήρων ή άλλης εντολής
h ή ←	Κίνηση του cursor αριστερά
l ή →	Κίνηση του cursor δεξιά
j ή ↑	Κίνηση του cursor πάνω
k ή ↓	Κίνηση του cursor κάτω
nh, nl, nj, nk	Κίνηση κατα n θέσεις αντίστοιχα
w	Ο cursor τοποθετείται στον πρώτο χαρακτήρα της επόμενης λέξης.
e	Ο cursor τοποθετείται στον τελευταίο χαρακτήρα της επόμενης λέξης.

b	Ο cursor τοποθετείται στον πρώτο χαρακτήρα της προηγούμενης λέξης.
nw, ne, nb	Κατα n χαρακτήρες αντίστοιχα με τα παραπάνω.
$	Ο cursor τοποθετείται στον τελευταίο χαρακτήρα της γραμμής.
0(μηδέν)	Ο cursor τοποθετείται στον πρώτο χαρακτήρα της γραμμής.
^	Ο cursor τοποθετείται στον πρώτο μή κενό χαρακτήρα της γραμμής.
n+	Ο cursor μετακινείται κατα n γραμμές πάνω.
n-	Ο cursor μετακινείται κατα n γραμμές κάτω.
H	Ο cursor τοποθετείται στην πρώτη γραμμή της οθόνης.
M	Ο cursor τοποθετείται στο μέσον της οθόνης.
L	Ο cursor τοποθετείται στην τελευταία γραμμή της οθόνης.
G	Ο cursor τοποθετείται στην τελευταία γραμμή του κειμένου.
ctrl-f	Εμφανίζεται η επόμενη οθόνη.
ctrl-b	Εμφανίζεται η προηγούμενη οθόνη.
ctrl-d	Εμφανίζεται η επόμενη μισή οθόνη.
ctrl-u	Εμφανίζεται η προηγούμενη μισή οθόνη.
fx	Αναζητεί τον πρώτο χαρακτήρα x προς τα δεξιά.
Fx	Αναζητεί τον πρώτο χαρακτήρα x προς τα αριστερά.
tx	Αναζητεί τον πρώτο χαρακτήρα x προς τα δεξιά, και τοποθετείται στον ακριβώς προηγούμενο χαρακτήρα.
Tx	Αναζητεί τον πρώτο χαρακτήρα x προς τα αριστερά, και τοποθετείται στον ακριβώς προηγούμενο χαρακτήρα.
;	Συνεχίζει την αναζήτηση όπως στην τελευταία εντολή.
,	Συνεχίζει την αναζήτηση όπως στην τελευταία εντολή αλλά προς την αντίθετη κατεύθυνση.
/string	Εντοπίζει την πρώτη εμφάνιση της ακολουθίας χαρακτήρων που εμφανίζεται στο string απο τη θέση που βρίσκεται ο cursor και προς τα κάτω.
?string	Εντοπίζει την τελευταία εμφάνιση του string.
n, /	Επαναλαμβάνει την προηγούμενη αναζήτηση.
N	Επαναλαμβάνει την προηγούμενη αναζήτηση στην αντίστροφη πορεία.
i	Εισάγει κείμενο πρίν απο τον θέση του cursor.
I	Εισάγει κείμενο στην αρχή της γραμμής.
a	Εισάγει κείμενο μετά απο την θέση του cursor.
A	Εισάγει κείμενο στο τέλος της γραμμής.

o	Εισάγει μία κενή γραμμή κάτω απο τον cursor.
O	Εισάγει μία κενή γραμμή πάνω απο τον cursor.
x	Σβήνει τον χαρακτήρα πάνω στον οποίο βρίσκεται ο cursor.
nx	Σβήνει n χαρακτήρες απο την θέση του cursor.
dw	Σβήνει μία λέξη.
ndw	Σβήνει n λέξεις.
dd	Σβήνει ολόκληρη την γραμμή όπου βρίσκεται ο cursor.
D	Σβήνει την τρέχουσα γραμμή απο τη θέση του cursor και μετά.
r	Επιτρέπει την αντικατάσταση του χαρακτήρα που βρίσκεται ο cursor.
R	Αντικαθίστανται οι χαρακτήρες που πληκτρολογούνται μέχρι να πατηθεί το πλήκτρο esc.
cw	Αντικατάσταση μιάς λέξης. Στον τελευταίο χαρακτήρα που μπορεί να αντικατασταθεί εμφανίζεται το $.
ncw	Όπως και παραπάνω, αλλά για n λέξεις.
cc	Επιτρέπει την αλλαγή χαρακτήρων στην τρέχουσα γραμμή.
ncc	Επιτρέπει την αλλαγή χαρακτήρων σε n γραμμές, συμπεριλαμβανομένης και της τρέχουσας.
C	Επιτρέπει την αλλαγή χαρακτήρων στην τρέχουσα γραμμή απο τη θέση που βρίσκεται ο cursor και μετά.
nC	Όπως και παραπάνω, αλλά για n γραμμές.
: g/string/	Τυπώνει στην οθόνη όλες τις γραμμές που περιέχουν το string.
: s/string1/string2/p <cr>	Αντικαθίσταται το string1 από το string2 την πρώτη φορά που θα συναντηθεί στη γραμμή.
: s/string1/string2/gp <cr>	Ομοίως αλλά όσες φορές απαντηθεί στη γραμμή.
: g/string1/s//string2/g <cr>	Αντικαθίσταται το string1 από το string 2 σε όλο το αρχείο.
nyw	αντιγραφή των επόμενων n λέξεων
nyy	αντιγραφή n γραμμών από την τρέχουσα και κάτω.
"c5dd	Διαγράφει 5 γραμμές και τις τοποθετεί στον προσωρινό καταχωρητή c.
"cp	Τοποθετεί το περιεχόμενο του καταχωρητή c μετά τον cursor.
J	ενώνει τη γραμμή που βρίσκεται ο cursor με την επόμενη.
u	Ακυρώνει την τελευταία εντολή που εδόθει.
U	Ακυρώνει όλες τις μεταβολές που έγιναν στην τρέχουσα γραμμή.

ctrl l	Επανεμφανίζει την οθόνη των στοιχείων στο τερματικό.
:.=	Απαντά με τον αριθμό της γραμμής
: n	Ο cursor πηγαίνει στη γραμμή n
: r filename<cr>	Το περιεχόμενο του αρχείου filename τοποθετείται μετά τη γραμμή που βρίσκεται ο cursor.
: nr filename<cr>	Τοποθετείται το περιεχόμενο του filename κάτω από την n-οστή γραμμή
ZZ ή :wq<cr>	Αποθηκεύει το περιεχόμενο του αρχείου και εγκαταλείπει τον vi
: q<cr>	Εγκαταλείπει τον vi χωρίς να καταχωρηθούν οι αλλαγές. Ο vi προειδοποιεί για το χάσιμο των αλλαγών που τυχόν έγιναν.
: q!<cr>	Όπως και ανωτέρω χωρίς προειδοποίηση.
: w nfile<cr>	Αποθηκεύεται το περιεχόμενο του αρχείου στο αρχείο nfile και παραμένει στο vi.
: w! filen<cr>	Αποθηκεύει το περιεχόμενο πάνω σε ήδη υπάρχον αρχείο με όνομα filen.

ΑΣΚΗΣΕΙΣ

1. Δίνεται η ακολουθία, A_n, που οι όροι της ικανοποιούν την αναδρομική σχέση:

$$A_{n+1} = 2A_n + \frac{3}{A_{n-1}} \qquad \text{με } A_0=A_1=1$$

Κατασκευάστε ένα πρόγραμμα FORTRAN που θα υπολογίζει και εκτυπώνει τις τιμές των όρων: A_{10}, A_{20}, A_{30}, A_{40} και A_{50}.

2. Δίνονται οι εξισώσεις των ελλείψεων:

$$\frac{x^2}{10^2} + \frac{y^2}{5^2} = 1 \qquad \frac{x^2}{4^2} + \frac{y^2}{10^2} = 1$$

Κατασκευάστε ένα πρόγραμμα που θα υπολογίζει τα σημεία με ακέραιες τιμές των συντεταγμένων (x,y) που ευρίσκονται εντός (ή επί) και των δύο ελλείψεων, και να τοποθετεί τις συντεταγμένες αυτές στους πίνακες CX(I),CY(I).
Στον πίνακα CX(I) θα τοποθετούνται οι τιμές των X.
Στον πίνακα CY(I) θα τοποθετούνται οι τιμές των Y.
Να εκτυπωθούν: το πλήθος των σημείων και οι πίνακες CX,XY.

3) Δίνεται η αναδρομική σχέση:

$$a_{n+3} = 2a_{n+2} + a_{n+1} - a_n \qquad \text{με } a_0 = 1,\ a_1 = 2,\ a_2 = 0$$

Κατασκευάστε το υποπρόγραμμα.

FUNCTION AKOL (N,A0,A1,A2)

που θα υπολογίζει τον N- όρο της ακολουθίας (a_N) δεδομένων των τριών πρώτων a_0, a_1, a_2.
Χρησιμοποιήστε το υποπρόγραμμα αυτό σε ένα κύριο πρόγραμμα για να υπολογίσετε και να εκτυπώσετε τις τιμές των: a_{10}, a_{30}, a_{50}, a_{100}.

4) Γράψτε ένα πρόγραμμα Fortran το οποίο θα "διαβάζει" έναν ακέραιο $N<1x10^{12}$ και θα υπολογίζει το πλήθος των ψηφίων του.

5) Δίνεται η αναδρομική σχέση:

$$a_{n+1} = a_n + n! \qquad \text{με } a_0 = 1$$

Γράψτε ένα πρόγραμμα Fortran που να υπολογίζει και να εκτυπώνει τους όρους a_{10}, a_{15} και a_{20}.

6) Γράψτε ένα υποπρόγραμμα FUNCTION: FUNCTION COMB (N,K)
που θα υπολογίζει τους συνδυασμούς

$$\binom{N}{K} = \frac{N!}{K!(N-K)!}$$

και ελέγξατε την ορθότητά της χρησιμοποιώντας την σε ένα κυρίως πρόγραμμα.

7) Δίνεται η αναδρομική σχέση

$$a_n = 2. + \frac{1}{a_{n-1}} \quad \text{με } a_0=1.$$

υπολογίστε τον όρο a_{20}.

8) Δίνεται η αναδρομική σχέση

$$a_{n+1} = \sqrt{a_{n+3}} \quad \text{με } a_0=3$$

Υπολογίστε τους όρους a_{10},a_{11},a_{12}.

9) Κατασκευάστε ένα πρόγραμμα που θα υπολογίζει τον ελάχιστο N>1 έτσι ώστε το άθροισμα:
$1^2+2^2+3^2+....+N^2$ να είναι τέλειο τετράγωνο ακέραιου αριθμού.

10) Κατασκευάστε ενα πρόγραμμα που θα υπολογίζει πόσοι είναι όλοι οι ακέραιοι τετραψήφιοι αριθμοί, που δεν περιέχουν το ψηφίο 8.

11) Γράψτε ένα πρόγραμμα το οποίο να μετατρέπει βαθμούς Κελσίου σε βαθμούς Φαρενάιτ και Κέλβιν.

12) Γράψτε ένα πρόγραμμα το οποίο να μετατρέπει μοίρες, σε βαθμούς και ακτίνια.

13) Δίνεται η συνάρτηση $Y=\sqrt[n]{bx}$. Να υπολογιστεί η τιμή του Y για x=1., 10., 1000., 10000. όταν β=2.5 και n=2,3,4.

14) Δίνεται η συνάρτηση $Y=A\sin(\frac{X}{Z})$. Υπολογίστε την τιμή του Y όταν A=4.2 και X=10., 20., 30., 40., 50., 60. και Z=1., 2., 3., 4..

15) Το ίδιο για τη συνάρτηση $Y=Ae^{-(\frac{X}{Z})}$

16) Υπολογίστε τις τιμές της συνάρτησης $Y=\sqrt{A+B\sin x}$ σε 100 ίσα διαστήματα του x, από 0 έως π/2, όταν A=1. και B=3.2.

17) Γράψτε ένα πρόγραμμα FORTRAN το οποίο να υπολογίζει τις τιμές της συνάρτησης

$$Y=\frac{(2x^2+3x+5)}{(x-3)}$$

για x=4., 5., 6.,....., 10..

18) Γράψτε πρόγραμμα FORTRAN το οποίο να υπολογίζει τις τιμές της συνάρτησης

$Y=2x^2 + \sin(\omega)$

για όλους τους συνδυασμούς των τιμών

x=1., 2., 3., 4. και ω=15., 30., 45., 60., 75., 90. (σε μοίρες).

19) Γράψτε πρόγραμμα FORTRAN το οποίο να υπολογίζει τις τιμές της συνάρτησης

$Y=x^2 + \ln x$

για x=2., 3., 4.,.....10..

20) Βρείτε για ποιές τιμές των X και Y ικανοποιείται η σχέση

$$\frac{X^2}{3^2} + \frac{Y^2}{3^2} = 1$$

21) Δίνεται η αριθμητική πρόοδος

$$1+2+3+.....n = \frac{1}{2} n (n+1)$$

(i) Γράψτε πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη της αριθμητικής προόδου για να επαληθεύσετε τις σχέσεις.

Κάντε το ίδιο για τις παρακάτω:

(ii) $1+3+5+7+...+(2n-1) = n^2$

(iii) $1^2+2^2+3^2+....+n^2 = \frac{n(n+1)(2n+1)}{6}$

(iv) $1^3+2^3+3^3+...+n^3= \frac{n^2(n+1)^2}{4} = (1+2+3+...n)^2$

(v) $1^4+2^4+3^4+....n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

(vi) $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-.... = \ln 2$

(vii) $\frac{1}{1*3}+\frac{1}{3*5}+\frac{1}{5*7}+\frac{1}{7*9}+.....=\frac{1}{2}$

(viii) $\frac{1}{1*3} + \frac{1}{2*4} + \frac{1}{3*5} + \frac{1}{4*6} + \ldots = \frac{3}{4}$

(ix) $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

22) Δίνονται δύο μονοδιάστατοι πίνακες Α και Β τα στοιχεία των οποίων είναι 10 τυχαίοι αριθμοί. Να βρεθεί ο πίνακας C τα στοιχεία του οποίου είναι η διαφορά των αντιστοίχων των Α και Β. Να βρεθεί ο μέσος όρος των στοιχείων του πίνακα C.

α) Γράψτε ένα πρόγραμμα FORTRAN για το παραπάνω πρόβλημα.

β) Μετατρέψτε το πρόγραμμα σε υποπρόγραμμα.

23) Γράψτε ένα πρόγραμμα το οποίο αφού διαβάζει έναν ακέραιο πενταψήφιο, θα σχηματίζει έναν άλλο πενταψήφιο με ανεστραμμένη τη σειρά των ψηφίων του πρώτου και θα τον τυπώνει.

24) Γράψτε ένα πρόγραμμα το οποίο να βρίσκει πόσοι και ποιοί απο τους ακέραιους μεταξύ 1 και 1000, είναι πρώτοι αριθμοί. (Πρώτος καλείται ένας αριθμός ο οποίος διαιρείται ακριβώς μόνον με τον εαυτόν του και τη μονάδα)

25) Δίδεται η συνάρτηση

$$Z(x,y) = \frac{x^2+3y^2}{x-2} - \frac{x^3+3y^3}{y-3},$$

όπου x=1., 2., 3., 4 και y=1., 2., 3., 4., 5..

Γράψτε πρόγραμμα που να υπολογίζει την τιμή του Z για όλους τους δυνατούς συνδυασμούς των τιμών των x και y. Μετατρέψτε το πρόγραμμα ώστε να περιέχει την εντολή DIMENSION.

26) Για κάθε δυνατό συνδυασμό των τιμών των x και y υπολογίστε την τιμή της παράστασης

$$Z(x,y) = \frac{x^2+y^2-2y}{(x-3)(y-4)}$$

όπου x=1., 2., 3., 4 και y=1., 2., 3., 4., 5..

27) Να γραφεί πρόγραμμα FORTRAN το οποίο θα διαβάζει ακέραιους αριθμούς από ένα αρχείο. Οι αριθμοί είναι τετραψήφιοι και είναι γραμμένοι ανά 8 σε κάθε γραμμή.

Να γραφούν 2 υποπρογράμματα τα οποία θα βρίσκουν
(i) τους αριθμούς που είναι δυνάμεις του 4 καθώς και το άθροισμά τους.
(ii) Τους αριθμούς που είναι τέλεια τετράγωνα κάποιων άλλων αριθμών και να καταγράφονται τα ζεύγη αυτά.

Η εκτύπωση των αποτελεσμάτων να γίνεται από το κυρίως πρόγραμμα.

28) Δίνεται ένας πίνακας A(120) με τυχαίους αριθμούς στις 120 θέσεις. Να βρεθούν και τυπωθούν όλα τα στοιχεία που είναι αριθμοί μεγαλύτεροι από το μηδέν, καθώς επίσης και η θέση τους στον πίνακα. Να βρεθεί επίσης και τυπωθεί το πλήθος τους και το ποσοστό τους επί του συνόλου των στοιχείων.

29) Δίδονται δύο πίνακες (A5,5) και B(5,5). Γράψτε πρόγραμμα FORTRAN το οποίο:
i) Να δημιουργεί πίνακα C(5,5) στον οποίο τα στοιχεία της άνω διαγωνίου και της διαγωνίου, είναι το αποτέλεσμα της άθροισης των στοιχείων των πινάκων A(5,5) και B(5,5). Τα στοιχεία της κάτω διαγωνίου να είναι ίσα με μηδέν. Τέλος να εκτυπώνει τους πίνακας A, B και C.

30) Κατασκευάστε ένα πρόγραμμα που θα διαβάζει ένα αρχείο στο οποίο θα περιέχεται ένα πρόγραμμα FORTRAN και θα παράγει σαν OUTPUT ένα άλλο αρχείο που θα είναι ίδιο με το προηγούμενο αλλά επιπρόσθετα θα έχει αύξοντες αριθμούς γραμμής τοποθετημένους δεξιότερα της 74ης στήλης.

31) Τα μαγικά τετράγωνα είναι διδιάστατοι πίνακες (nxn) που περιέχουν τους αριθμούς 1,2,3,4.....,(n^2) σαν στοιχεία τους, τοποθετημένους κατά τέτοιον τρόπο ώστε το άθροισμα οποιασδήποτε στήλης, γραμμής ή διαγωνίου να είναι σταθερό (και ίσο προς $n(n^2+1)/2$). Μερικά παραδείγματα φαίνονται στο παρακάτω σχήμα.

8	1	6
3	5	7
4	9	2

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Υπάρχουν πολλοί τρόποι κατασκευής των μαγικών τετραγώνων. Θα περιγράψουμε την μέθοδο του de la Loubere του 17ου αιώνα, που ισχύει όμως μόνον για τετράγωνα (nxn), όπου το n είναι περιττός.

Πρώτα τοποθετούμε τον αριθμό 1 στο μεσαίο τετράγωνο της πρώτης γραμμής. Οι επόμενοι αριθμοί τοποθετούνται με την φυσική τους σειρά ο ένας μετά τον άλλον διαγωνίως προς τα πάνω και δεξιά με τις εξής διαφοροποιήσεις:

α) 'Οταν ξεπεράσουμε την πρώτη γραμμή (και βγαίνουμε έξω από το τετράγωνο προς τα πάνω) ο αριθμός τοποθετείται στην τελευταία γραμμή (σαν να ήταν η τελευταία γραμμή ακριβώς πάνω από την πρώτη).

β) 'Οταν ξεπεράσουμε και την πιο δεξιά στήλη (και βγαίνουμε δεξιά έξω από το τετράγωνο) ο αριθμός τοποθετείται στην πρώτη στήλη εξ' αριστερών (σαν να ήταν αυτή αμέσως δεξιότερα της πιο δεξιάς στήλης).

γ) 'Οταν φθάσουμε σε τετράγωνο που είναι ήδη γεμάτο ο αριθμός τοποθετείται στο τετράγωνο που βρίσκεται ακριβώς από κάτω του τελευταίου τετραγώνου που γεμίσαμε.

Κατασκευάστε ένα πρόγραμμα FORTRAN που θα κατασκευάζει μαγικά τετράγωνα τάξεως n, με n<30.

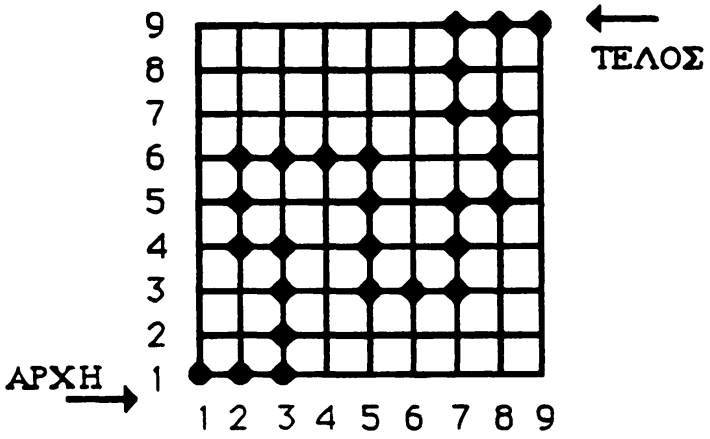
32) Μια μέθοδος παραδειγματισμού που χρησιμοποιούσαν πολλές φορές οι Ρωμαίοι για να αποθαρρύνουν τυχόν στάσεις των στρατευμάτων τους ήταν η παρακάτω:

Από τους στασιαστές διάλεγαν 1000 άνδρες που τους τοποθετούσαν σε μια ευθεία γραμμή. Κατόπιν άρχιζαν να τους εκτελούν ανά δυο. Δηλαδή την πρώτη φορά εκτελούσαν τον δεύτερο, τον τέταρτο κ.λ.π. Εάν ο τελευταίος της σειράς

επιζούσε, ετοποθετείτο πρώτος στην ευθεία για την επόμενη εκτέλεση. Με τον τρόπο αυτό, τελικά επιζούσε ένας, ο οποίος αφηνόνταν ελεύθερος ώστε να κάνει γνωστή την ιστορία στο υπόλοιπο στράτευμα.

Να κατασκευασθεί ένα πρόγραμμα που θα υπολογίζει την θέση στην αρχική ευθεία των 1000 ατόμων, που εξασφαλίζει την σωτηρία.

33) Η Μάτα Χάρι είναι εγκλωβισμένη σε έναν παγιδευμένο λαβύρυνθο μεταφέροντας μαζί της απόρρητα έγγραφα θεμελιώδους σημασίας. Ξεκινώντας από την αρχή θα πρέπει να φθάσει στο τέλος του λαβύρινθου. (Βλέπε σχήμα).



Για να επιτύχει το σκοπό της θα πρέπει να προσέχει το κάθε της βήμα. Μπορεί να κινείται κάθετα ή οριζόντια με μοναδιαία βήματα. (Το βήμα της δηλαδή είναι ίσο με την πλευρά ενός τετραγώνου). Οι μαυρισμένοι κόμβοι είναι ασφαλείς, ενώ οι μη μαυρισμένοι είναι θανατηφόρες παγίδες. Οι συνεργάτες της που βρίσκονται έξω από τον λαβύρινθο γνωρίζουν τις συντεταγμένες των ασφαλών κόμβων και στέλνουν οδηγίες στην Μάτα Χάρι για το πως να κινηθεί. Οι οδηγίες είναι κωδικοποιημένες πολύ απλά ως εξής: U, D, L, R και αντιστοιχούν σε βήματα προς τα επάνω (U), κάτω (D), αριστερά (L) και δεξιά (R).

Κατασκευάστε ένα πρόγραμμα σε FORTRAN που έχοντας σαν INPUT τις παραπάνω συντεταγμένες (x_i,y_i) και δεδομένου ότι ο δρόμος είναι μοναδικός και δεν υπάρχουν παρακλάδια που οδηγούν σε αδιέξοδα, θα παράγει σαν OUTPUT τις οδηγίες για την Μάτα Χάρι.

34) Στον ιερό ναό του Βραχμάνων στο Μπεναρές κάτω από τον θόλο που συμβολίζει το κέντρο του κόσμου υπάρχουν τρείς διαμαντένιες βελόνες.

Σε μια από αυτές πιστεύεται ότι ο Βράχμα τοποθέτησε 64 δίσκους από καθαρό χρυσάφι έτσι ώστε ο δίσκος με την μεγαλύτερη διάμετρο να είναι στη βάση και οι άλλοι να είναι κατά σειρά διαμέτρου ολοένα και μικρότερη. Αυτός είναι ο πύργος του Βράχμα. Μέρα και νύχτα οι ιερείς μεταφέρουν τους δίσκους από τη μια βελόνα στην άλλη, σύμφωνα με τους νόμους του Βράχμα, που απαιτούν ο ιερέας της βάρδιας να μετακινεί ένα μόνο δίσκο έτσι ώστε πάντοτε, να μην υπάρχει δίσκος μικρότερης διαμέτρου κάτω απ' αυτόν. 'Οταν όλοι οι 64 δίσκοι μεταφερθούν από την αρχική βελόνα σε μία από τις άλλες βελόνες τότε σύμφωνα με τους βραχμάνους είναι η συντέλεια του κόσμου.

Εάν η κίνηση ενός δίσκου απαιτεί ένα sec να υπολογισθεί ο χρόνος συντελείας. Να χρησιμοποιηθεί η αναδρομική σχέση $u_n = 2u_{n-1}+1$ όπου u_n ο μικρότερος δυνατός αριθμός κινήσεων μεταφοράς ενός πύργου από n δίσκους.

35) Η μετάδοση κωδικοποιημένων μηνυμάτων (διαβαθμισμένων με απαιτήσεις ασφαλείας) βασίζεται στις ακέραιες λύσεις της εξίσωσης

$$x^n + y^n = z^n \qquad (1)$$

Να μελετηθεί για την περίπτωση n=2 ποιοί ακέραιοι αριθμοί x,y,z μικρότεροι του 50 $(x,y,z) \leq 50$ επαληθεύουν την εξίσωση (1).
π.χ. η τριάδα (x,y,z) = (3,4,5) επαληθεύει την εξίσωση (1).

α) Να γραφεί ένα πρόγραμμα που να σαρώνει όλους τους ακέραιους (x,y,z) μικρότερους του 50 που θα τυπώνει τους αριθμούς που έχουν την ιδιότητα (1) για n=2.
β) Να επαναληφθεί η διαδικασία για n=3.
(Υπόδειξη: Χρησιμοποιείστε DOUBLE PRECISION για όλους τους υπολογισμούς. Μην επιτρέψετε στα x,y,z να πάρουν την τιμή 0 διότι αποτελεί προφανή λύση.)

36) Πέντε ναυαγοί προσεγγίζουν ένα νησί του Ειρηνικού, όπου ζει ένας πίθηκος που τρέφεται με καρύδες κοκοφοίνικα. Οι ναυαγοί μαζεύουν όλες τις καρύδες με σκοπό να τις μοιραστούν μεταξύ τους την άλλη μέρα και πέφτουν για ύπνο. 'Ενας ναυαγός όμως ξυπνάει και πάει να πάρει το 1/5 που του αναλογεί. Μετράει τις καρύδες αλλά δεν είναι διαιρετές ακριβώς δια 5, αλλά περισσεύει μια. Δίνει λοιπόν την μία στον πίθηκο, παίρνει το 1/5 και το κρύβει κάπου. Η

ίδια διαδικασία επαναλαμβάνεται και με τους υπόλοιπους 4 ναυαγούς, για τις καρύδες που απομένουν. Οι καρύδες ποτέ δεν διαιρούνται ακριβώς με το 5 αλλά πάντα περισσεύει μια την οποία καρπούται ο πίθηκος. Την άλλη μέρα υπάρχουν ακόμα αρκετές καρύδες οι οποίες και πάλι δεν διαιρούνται ακριβώς με το 5, μια περισσεύει. Την δίνουν στον πίθηκο και ο καθένας παίρνει το 1/5 από τις καρύδες που απέμειναν τελικά. Ποιός είναι ο ελάχιστος αριθμός καρύδων ώστε η περιγραφείσα διαδικασία να είναι δυνατή;

Το πρόβλημα είναι δυνατόν να λυθεί αναλυτικά πολύ εύκολα. Παρόλα αυτά δεν είναι αυτός ο σκοπός. Θέλουμε να αναπτύξουμε ένα κώδικα FORTRAN που να έχει την παρακάτω φιλοσοφία: Έστω a_0 ο αρχικός αριθμός των καρύδων και a_n ο αριθμός των καρύδων που απέμειναν μετά που και ο n-οστός ναυαγός πήρε το μερίδιό του (n=1,2,....5). Γνωρίζουμε ότι:

$$a_n = 4/5\ (a_{n-1}-1) \qquad (1)$$

και ότι

$$a_5 = 5K + 1 \qquad (2)$$

Ζητάμε τη λύση των (1) και (2) για τον ελάχιστο ακέραιο K και όπου οι a_0, a_1, a_2, a_3, a_4, a_5 είναι επίσης ακέραιοι.

37) Μία από τις μοντέρνες μεθόδους για τη μελέτη μοντέλων με την βοήθεια ενός Η/Υ είναι η μέθοδος της προσομοίωσης (Computer simulation). Θεωρείστε το παρακάτω πρόβλημα:

Ένα δάπεδο απείρων διαστάσεων είναι χαραγμένο με παράλληλες ισαπέχουσες ευθείες. Έστω d η σταθερή απόσταση δύο γειτονικών παράλληλων. Αποδεικνύεται αναλυτικά, βάσει της θεωρίας πιθανοτήτων ότι, εάν μία ράβδος μήκους d, ριφθεί τυχαία στο δάπεδο, η πιθανότητα να τμήσει μία από τις παράλληλες ευθείες είναι ίση προς 2/π (π=3.14159...). Η προσέγγιση του προβλήματος αυτού με την τεχνική της προσομοιωσης είναι δυνατόν να γίνει ως εξής:

Η τυχαία ρίψη της ράβδου προσομοιώνεται με δύο τυχαίους αριθμούς.

1) Η απόσταση, X, του πλησιέστερου άκρου της ράβδου ως προς την πρώτη από αριστερά μη τεμνόμενη παράλληλο. (ο αριθμός αυτός είναι τυχαίος στο διάστημα (0,d).

2) Η γωνία θ, της διεύθυνσης της ράβδου με την διεύθυνση των παραλλήλων ευθειών. (τυχαίος στο διάστημα (0,π)).
Εάν x+dsinθ > d, η ράβδος τέμνει μία παράλληλο, ειδ' άλλως η ράβδος δεν τέμνει καμμία παράλληλο. Εάν κατασκευάσουμε n ζεύγη (x,θ) τυχαίων αριθμών στα αντίστοιχα διαστήματα (0,d) και (0,π) (που ισοδυναμεί με n τυχαίες ρίψεις της ράβδου) και βρούμε ότι στις k περιπτώσεις η ράβδος τέμνει μία παράλληλο, τότε η πιθανότητα που ζητάμε προσεγγίζεται από το λόγο k/n.

Γράψτε ένα πρόγραμμα FORTRAN που θα διαλέγει τυχαίους αριθμούς σε ζεύγη και για κάθε ζεύγος να υπολογίζει αν υπάρχει ή όχι τετμημένη παράλληλος. Να υπολογίζει πόσα ζεύγη οδηγούν σε τμήση, και εν συνεχεία να διαιρεί με τον αριθμό των συνολικών ζευγών ώστε να ευρεθεί με αυτόν τον τρόπο προσεγγιστικά η πιθανότητα τμήσης.

38) Δίνεται ένα αρχείο με το όνομα RDATA στο οποίο είναι γραμμένοι αριθμοί, ανά τρείς σε κάθε γραμμή και δεν είναι γνωστό το πλήθος τους, αλλά είναι μικρότερο του 999. Ο πρώτος αριθμός κάθε γραμμής είναι ο αύξων αριθμός , i , ο δεύτερος περιέχει την ποσότητα X_i και ο τρίτος την Y_i. Το τέλος του αρχείου χαρακτηρίζεται από τον αριθμό 1999 στην τελευταία γραμμή .
Γράψτε ένα πρόγραμμα FORTRAN το οποίο:

(i) Θα ανοίγει το αρχείο RDATA και θα διαβάζει τους αριθμούς καταχωρώντας τους σε πίνακες.

(ii) Θα καλεί μία υπορουτίνα η οποία θα τακτοποιεί τα στοιχεία του πίνακα X κατά αύξουσα σειρά, με αντίστοιχη μετακίνηση των στοιχείων του πίνακα Y όπου χρειάζεται.

(iii) Θα γράφει τα στοιχεία σε ένα νέο αρχείο DDATA, το οποίο θα πρέπει να περιλαμβάνει τα εξής:
- Τη μέγιστη και ελάχιστη τιμή του πίνακα X στην αρχή του αρχείου.
- Τη μέγιστη και ελάχιστη τιμή του πίνακα Y στην επόμενη γραμμή.
- Ακολουθούν σε τρείς στήλες, ο νέος αύξων αριθμός, οι τιμές του X και του Y με καθορισμένο FORMAT (F και όχι E).

39) Δίνονται οι συντεταγμένες των κορυφών ενός πενταγώνου (x_i,y_i).
Να υπολογιστεί η περίμετρος του πενταγώνου.
(Η απόσταση μεταξύ δύο σημείων (x_1,y_1) και (x_2,y_2) δίνεται από τη σχέση $d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$)

Τα στοιχεία διαβάζονται από αρχείο. Κάντε το πρόγραμμά σας γενικώτερο ώστε να εφαρμόζεται σε πολύγωνο.

40) Μια φοιτήτρια έχει το όνομα POZA I AZOP και της αρέσουν οι γρίφοι. Ζητά λοιπόν από την παρέα της να λύσει το εξής πρόβλημα: Στο όνομα της POZA κάθε γράμμα παριστά έναν μονοψήφιο ακέραιο αριθμό και ο τετραψήφιος που σχηματίζεται αν πολλαπλασιαστεί με έναν άλλο ακέραιο, δίνει τον τετραψήφιο αναγραμματισμένο έτσι ώστε να ισχύει η σχέση POZA*I = AZOP.
Γράψτε ένα πρόγραμμα για να λύσετε το παραπάνω πρόβλημα.

41) Γράψτε ένα πρόγραμμα το οποίο θα τυπώνει τα

X	X**2	x**3	SQRT(X)	LOG(X)	FACTORIAL(X)
1.0	1.0	1.0	1.0	0.0	0.10000000E01
2.0.....					

όταν το X παίρνει τιμές από 1 έως 50. Το παραγοντικό (FACTORIAL) να υπολογιστεί για τιμές μικρότερες του 20 ενώ οι υπόλοιπες θέσεις θα παραμείνουν κενές στην εκτύπωση.

42) Δίνεται η συνάρτηση $Z=\sin(X+B)+A\log\frac{B+X}{2}$.

Γράψτε πρόγραμμα FORTRAN που να υπολογίζει την τιμή της συνάρτησης για όλους τους συνδυασμούς των μεταβλητών

x = 1.0, 1.1, 1.2, 1.3, ...,2.0

A = 0.10, 0.15, 0.20, ...,1.0

B = 1.0, 2.0, 3.0,.....10.0

Χρησιμοποιώντας υπορουτίνα ή υπορουτίνες.
Τα αποτελέσματα να τυπώνονται στο κυρίως πρόγραμμα.

43) Ένας ενδιαφερόμενος να πάρει δάνειο, θέλει να γνωρίζει ποιό ποσό θα πληρώνει κάθε μήνα μέχρι την εξόφληση του δανείου, αν ζητήσει δάνειο 1000000.. 2000000., 3000000., 4000000. ή 5000000. με χρόνο εξόφλησης 24, 36,

48, ή 60 μήνες. Υπολογίστε τα στοιχεία για επιτόκιο 15%, 18%, 20%, και 25%. Για τον προσδιορισμό του ποσού της δόσης χρησιμοποιείστε τη σχέση

$$S = \frac{P}{R}\left(1-\frac{1}{(1+R)^N}\right)$$

όπου S το ποσόν της δόσης, P το κεφάλαιο δανεισμού, N ο χρόνος εξόφλησης σε μήνες και R το επιτόκιο σε εκατοστιαίες μονάδες.

44) Δίνονται δέκα τυχαίοι θετικοί ακέραιοι αριθμοί που αποτελούν τα στοιχεία ενός πίνακα A(10). Να βρεθεί το ελάχιστο κοινό πολλαπλάσιο των αριθμών αυτών.
(Υπόδειξη: Βρίσκεται ο μεγαλύτερος των αριθμών και ελέγχεται αν διαιρείται ακριβώς από όλους του υπόλοιπους. Αν ναι, αυτός είναι ο ζητούμενος. Αν όχι, τότε ο μεγαλύτερος διπλασιάζεται και ελέγεται αν ο νέος αριθμός διαιρείται ακριβώς από τους υπόλοιπους. Η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί κάποιο πολλαπλάσιο του μεγαλύτερου αριθμού, το οποίο διαιρείται ακριβώς από όλους τους άλλους).

45) Ο επαναληπτικός αλγόριθμος που βρίσκει την τετραγωνική ρίζα ενός αριθμού A δίνεται από τη σχέση

$$X_{n-1} = \frac{1}{2}\left(X_n + \frac{A}{X_n}\right)$$ (τύπος του Newton)

Η πρώτη τιμή X_1 δίνεται κατ' εκτίμηση ($X_1 = 1.$ ή $X_1 = \frac{A}{2.}$)

Το πλήθος των επαναληπτικών προσεγγίσεων καθορίζεται από το εξής κριτήριο. Η απόλυτη τιμή της διαφοράς δυο διαδοχικών προσεγγίσεων πρέπει να είναι μικρότερη ενός ορίου π.χ. 10^{-5}.
Γράψτε πρόγραμμα το οποίο να βρίσκει την τετραγωνική ρίζα ενός αριθμού, χρησιμοποιώντας υπορουτίνα για τον προσδιορισμό. Η ανάγνωση και η εκτύπωση του αριθμού και της τετραγωνικής του ρίζας να γίνεται από το κυρίως πρόγραμμα.

46) Γράψτε πρόγραμμα FORTRAN, το οποίο να υπολογίζει το εμβαδόν ακανόνιστου σχήματος πολυγώνου, αν είναι γνωστές οι συντεταγμένες των κορυφών του.
(Υπόδειξη: Το πρόβλημα ανάγεται στον υπολογισμό εμβαδών τριγώνων. Αρχίζοντας από μία κορυφή, το πολύγωνο χωρίζεται σε τρίγωνα με διαδοχικές πλευρές, που έχουν κοινή τη μία κορυφή).

47) Γράψτε πρόγραμμα FORTRAN το οποίο να βρίσκει το Μέγιστο Κοινό Διαιρέτη ενός συνόλου ακεραίων αριθμών.
(Υπόδειξη: Βρίσκεται ο μικρότερος των αριθμών και ελέγχεται αν διαιρεί ακριβώς όλους τους άλλους. Αν ναι, αυτός είναι ο ζητούμενος. Αν όχι, τότε όλα τα υπόλοιπα των προηγούμενων των διαιρέσεων, που είναι διάφορα του μηδενός, αποτελούν ένα νέο σύνολο αριθμών, όπου επαναλαμβάνεται η προηγούμενη διαδικασία).

48) Έστω το πολυώνυμο

$$A_0x^n + A_1x^{n-1}+....+A_{n-1}X+A_n$$

Γράψτε πρόγραμμα FORTRAN το οποίο αφού διαβάζει το βαθμό του πολυωνύμου και τους συντελεστές του, να υπολογίζει τους συντελεστές του πολυωνύμου.

$$B_0x^{n-1} + B_1x^{n-2}+....+B_{n-1}$$

που είναι η παράγωγος αυτού που δόθηκε αρχικά.
(Υπόδειξη: $B_i = A_i(N-1)$)

49) Έστω το πολυώνυμο

$$A_0x^n + A_1x^{n-1}+....+A_{n-1}X+A_n$$

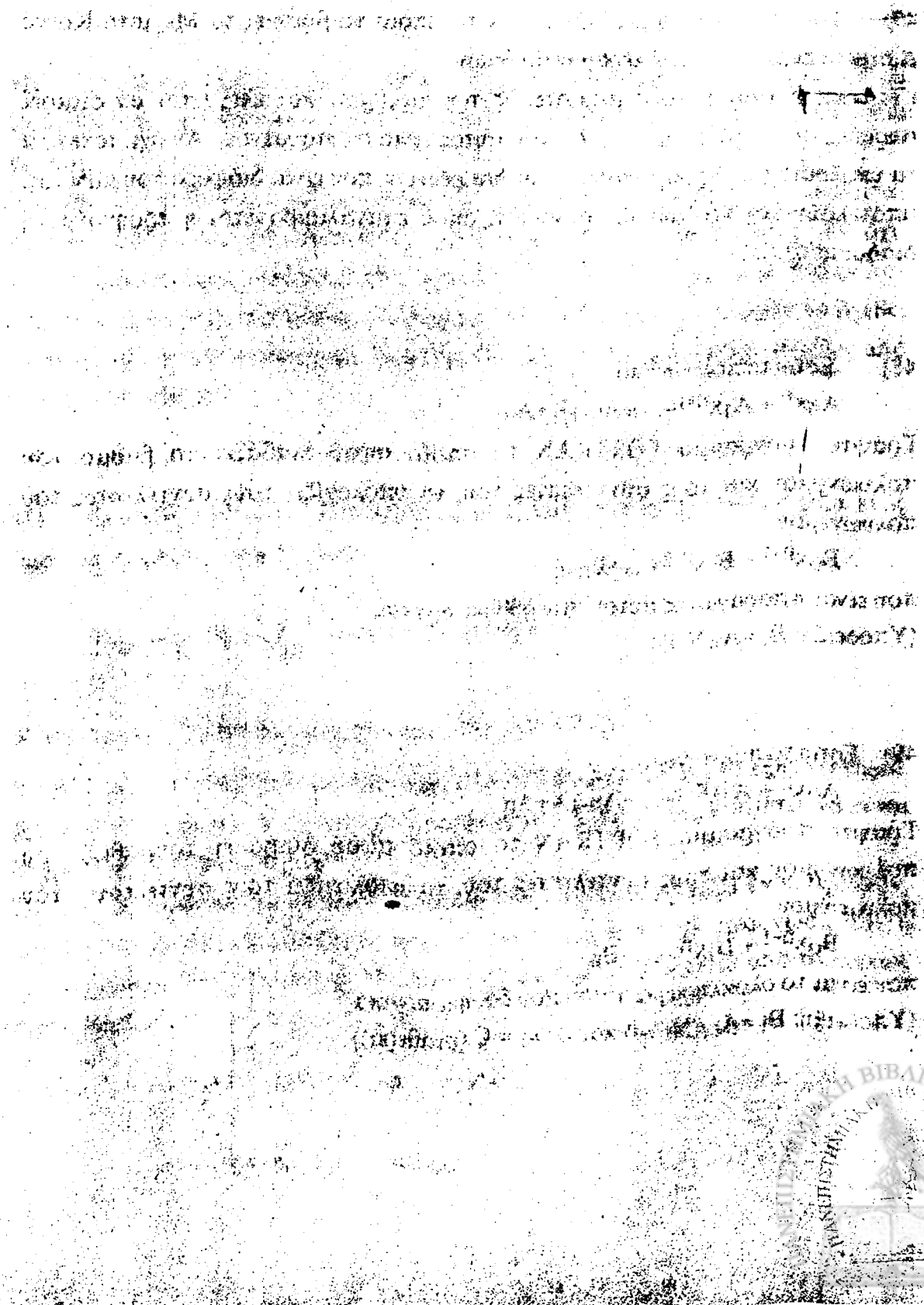
Γράψτε πρόγραμμα FORTRAN το οποίο αφού διαβάσει το βαθμό του πολυωνύμου και τους συντελεστές του, να υπολογίζει τους συντελεστές του πολυωνύμου

$$B_0x^{n-1} + B_1x^n +....+B_{n+1}$$

που είναι το ολοκλήρωμα αυτού που δόθηκε αρχικά
(Υπόδειξη: $B_i = A_i /(N+1-i)$ και $B_{n+1} = C$ (σταθερά))

50) Γράψτε πρόγραμμα FORTRAN το οποίο να υπολογίζει το εμβαδόν ενός τετραέδρου χρησιμοποιώντας υπορουτίνα. Μετατρέψτε το πρόγραμμα ώστε να χρησιμοποιεί FUNCTION.

51) Γράψτε πρόγραμμα FORTRAN το οποίο αφού διαβάσει έναν ακέραιο αριθμό, θα τον αναλύει στα ψηφία του καταχωρώντας τα σε μια μεταβλητή με δείκτες.



Τυπώθηκε στο Πανεπιστημιακό Τυπογραφείο με δαπάνη του Πανεπιστημίου Ιωαννίνων

Κ.Α. Πανεπιστημιακού Τυπογραφείου.....

ΔΙΑΝΕΜΕΤΑΙ ΔΩΡΕΑΝ στους φοιτητές.

785/98 Συγγραφέας: ΜΠΑΚΑΣ, Θ.....

Τίτλος: Εισαγωγή στη γλώσσα προγραμ/μού FORTRAN ΑΡ. ΕΙΣ.:

ΔΑΝΕΙΖΟΜΕΝΟΣ	ΗΜΕΡΟΜΗΝΙΑ ΔΑΝΕΙΣΜΟΥ	ΗΜΕΡΟΜΗΝΙΑ ΕΠΙΣΤΡΟΦΗΣ	
Τσάμαν	11/6	18/6	R
Μπακιζάμη	15/3	22/3	R
Δούκα	15/5	22/5	R
Καλύτζης	5/4	23/4	45423
-''-	4/4	18/5	
Ζήτσιος		5/07/02	R