ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ
ΕΜ 091
ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ Η/Υ

ΘΕΟΔΩΡΟΣ ΚΑΤΣΑΟΥΝΗΣ

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

1.1 Σύντομη ιστορική αναδρομή

Σκοπός του μαθήματος αυτού είναι μια εισαγωγή στη χρήση τόσο των υπολογιστών όσο και την χρήση του πλέον απαραίτητου λογισμικού.

Σχεδόν δύο δεκαετίες πριν αρχίσαν να κάνουν την εμφανισή τους οι πρώτοι *Προσωπικοί Υπολογιστές (ΠΥ ή PC)* . Τότε κανείς δεν φανταζόταν την επανάσταση που επρόκειτο να ακολουθήσει. Στις αρχές της δεκαετίας του 1980 κυριαρχούσαν τα *Μεγάλα Υπολογιστικά Συστήματα (ΜΥΣ ή Mainframes)*, προσανατολισμένα κυρίως σε αριθμητικούς υπολογισμούς. Άλλωστε αυτός ήταν και ο πρωταρχικός λόγος κατασκεύης των μηχανών αυτών. Στη συνέχεια η εμφάνιση των προσωπικών υπολογιστών δημιούργησε καινούργια δεδομένα. Ο υπολογιστής πλέον καθιερώνεται και γίνεται άμεσα προσβάσιμος από τον καθένα. Σιγά - σιγά γίνεται αναγκαία η εκπαίδευση των νέων επιστημόνων, κωρίως τεχνολογικών και θετικών κατευθύνσεων στους υπολογιστές. Επιπλέον διευρύνεται και η χρήση των υπολογιστών, έτσι από καθαρά υπολογιστικές μηχανές, χρησιμοποιούνται για πληθώρα άλλων εφαρμογών, όπως δακτυλογράφηση κειμένων, αποθήκευση στοιχείων, δημιουργία βάσεων δεδομένων και πολλές άλλες εφαρμογές.

Μια δεύτερη επανάσταση πραγματοποείται στα μέσα της δεκαετίας του 1980 με την εισαγώγη του *Παραθυρικού Περιβάλλοντος* (Windows) στους ΠΥ. Η χρήση τους διευρύνεται ακόμα περισσότερο μιας και γίνονται φιλικότεροι προς τους χρήστες. Φυσικά οι ΠΥ δεν ήταν σε θέση να εκτελέσουν σοβαρούς αριθμητικούς υπολογισμούς. Είναι προφανές λοιπόν ότι δημιουργείται ένα μεγάλο κενό : Άπο την μία μεριά έχουμε τους ΠΥ με μικρή υπολογιστική ισχύ και από την άλλη τα ΜΥΣ με σχετικά μεγάλη για την εποχή, υπολογιστική ισχύ. Το κενό αυτό έρχεται να το καλύψει περί τα τέλη της δεκαετίας του 1980 οι *Σταθμοί Εργασίας (ΣΕ ή Workstations)*. Οι ΣΕ είναι εξελιγμένα υπολογιστικά συστήματα με πολύ περισσότερη υπολογιστική ισχύ από ότι οι ΠΥ αλλά σαφώς μικρότερη αυτών των ΜΥΣ. Οι πρώτοι ΣΕ ήταν σε θέση να λύσουν μικρής σχετικής κλίμακας υπολογιστικά προβλήματα. Με την πάροδο όμως των χρόνων οι ΣΕ εξελίσσονται σε ισχυρά υπολογιστικά συστήματα φτάνοντας έτσι στη σημερινή εποχή να υπάρχουν ΣΕ με δυνατότητα εκτέλεσης 4×10^8 πράξεις το δευτερόλεπτο. Αρκεί να αναφέρουμε εδώ ότι η ισχύς των ΜΥΣ στις αρχές της δεκαετίας του 1980 δεν ξεπερνούσε τις 10^6 πράξεις το δευτερόλεπτο.

Η συνεχής αύξηση των υπολογιστικών αναγκών οδηγεί στη κατασκευή υπολογιστών με περισσότερους από ένα επεξεργαστές. Αυτό έχει σαν άμεση συνέπεια την αύξηση της υπολογιστικής ισχύς αλλά ταυτόχρονα απαιτεί από τον χρήστη περισσότερο χρόνο στη προσαρμογή των υπαρχόντων κωδίκων στα νέα δεδομένα της παράλληλης επεξεργασίας.

1.2 Σημερινά Υπολογιστικά Προβλήματα

Τα σημερινά υπολογιστικά προβλήματα απαιτούν ολοένα και περισσότερη υπολογιστική ισχύ. Τα προβλήματα αυτά προέρχονται από διάφορους κλάδους των επιστημών : Μαθηματικά, Φυσική, Χημεία, Βιολογία, Κατασκεύες, Οικονομία κ.λ.π. Συνήθως η λύση ενός τέτοιου προβλήματος αποτελείται από τρία διαφορετικά στάδια :

1. Διατύπωση του φυσικού προβλήματος.
2. Ανάπτυξη ενός μαθηματικού μοντέλου για την περιγραφή του.
3. Αριθμητική επίλυση του μαθηματικού προβλήματος :
   (α′) Αριθμητική μέθοδος.
   (β′) Κώδικας και υλοποίηση του στον Η/Υ.

Τα βασικά στοιχεία του υπολογισμού της αριθμητικής λύσης του προβλήματος είναι τα εξής :

(i). Η/Υ (Hardware)

(ii). Λογισμικό (Software)

1.3 Δομή Η/Υ

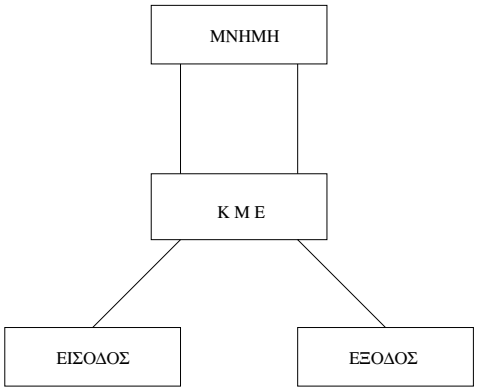
Τα Μηχανικά Μέρη (Hardware)

Ο Η/Υ αποτελείται από τα εξής βασικά μηχανικά μέρη :

(i). Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ ή CPU).

(ii). Μνήμη (ROM/RAM).

(iii). Μονάδες Εισόδου/Εξόδου (Input/Output Devices)



Σχήμα 1.1: Βασική Δομή Η/Υ

Η ΚΜΕ αποτελεί την καρδιά του Η/Υ. Όλες οι εντολές εκτελούνται απο την ΚΜΕ. Η ΕΙΣΟΔΟΣ και η ΕΞΟΔΟΣ είναι τα μέρη του υπολογιστή από τα οποία γίνεται η είσοδος και έξοδος πληροφορίων. Είναι τα μηχανικά εκείνα μέρη με τα οποία γίνεται η επικοινωνία του χρήστη με τον Η/Υ. Την ΕΙΣΟΔΟ του Η/Υ την αποτελούν το πληκτρολόγιο και μια μονάδα δίσκου, σκληρού ή μαλακού. Την ΕΞΟΔΟ του Η/Υ την αποτελεί η οθόνη, ο εκτυπωτής ή οποιοδήποτε άλλο μέσο αποθηκεύσης πληροφορίων (δίσκος, δισκέττα). Η ΜΝΗΜΗ είναι το μηχανικό εκείνο κομμάτι του Η/Υ όπου αποθηκεύονται προσωρινά διάφορες πληροφορίες. Επίσης σαν μνήμη μπορούν να χαρακτηριστούν και άλλα μέσα όπως σκληροί και μαλακοί δίσκοι, οι μαγνητικές ταινίες. Τα δύο βασικά χαρακτηριστικά της μνήμης είναι το μέγεθος της, δηλαδή η χωρητικότητα της σε όγκο πληροφοριών και δεύτερο ο χρόνος πρόσβασης σε αυτήν, δηλαδή ο χρόνος που απαιτείται για να βρεθεί και να γίνει διαθέσιμη μιά πληροφορία στην ΚΜΕ.

1.4 Λογισμικό (Software)

Το λογισμικό σε ένα Η/Υ αποτελείται από τα προγράμματα εκείνα που ελέγχουν και κατευθύνουν τα μηχανικά μέρη. Το λογισμικό απαρτίζεται από δύο βασικές κατηγορίες προγραμμάτων :

(i). Λογισμικό του συστήματος : Είναι τα διάφορα προγράμματα επικοινωνίας και διαχείρησης του Η/Υ από το χρήστη, π.χ. Λειτουργικό σύστημα.

(ii). Εφαρμογές : Διάφορα προγράμματα που εκτελούν συγκεκριμμένες εργασίες.

1.5 Λειτουργικό Σύστημα (Operating System)

Το λειτουργικό σύστημα είναι το σύνολο των προγραμμάτων εκείνων που βοηθούν στην επικοινωνία και διαχείρηση του Η/Υ από το χρήστη. Το λειτουργικό σύστημα φορτώνεται κάθε φορά που ανοίγουμε τον Η/Υ. Μερικές από τις βασικές εργασίες για τις οποίες είναι υπεύθυνο το λειτουργικό σύστημα είναι οι ακόλουθες:

(i). Εκτελεί άμεσα τις εντολές του χρήστη παρέχοντας χρήσιμες πληροφορίες.

(ii). Διαχειρίζεται τα αρχεία όλων των χρηστών.

(iii). Διευθύνει την ροή δεδομένων από/και προς τις διάφορες εισόδους/εξόδους.

(iv). Προσδιορίζει την χρήση του Η/Υ και καταγράφει, ελέγχει τις όποιες ενέργειες γίνονται πάνω σε αυτόν.

Κεφάλαιο 2

ΤΟ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ UNIX

2.1 Εισαγωγή

Το λειτουργικό σύστημα UNIX γράφτηκε από τον Ken Thompson των Bell Laboratories το 1969 σαν αποτέλεσμα μιας προσπάθειας να φτιαχθεί ένα λειτουργικό σύστημα που το περιβάλλον του να είναι εύκολο προς το χρήστη. Ο πυρήνας(kernel) του UNIX, δηλαδή η καρδιά του λειτουργικού συστήματος γράφτηκε σε C από τους Thompson & Ritchie τέσσερα χρόνια αργότερα έτσι ώστε το UNIX να είναι ανεξάρτητο του υπολογιστικού συστήματος. Οι βασικές δομές του UNIX είναι :

(i). Ο πυρήνας (kernel).

(ii). Το κέλυφος (shell).

(iii). Πρόγραμμα Επικοινωνίας (User Interface)

Ο *πυρήνας* ή αλλιώς το κυρίως λειτουργικό σύστημα, έχει ως βασική λειτουργεία τον προγραμματισμό της κεντρικής μονάδας επεξεργασίας(ΚΜΕ) και γενικώς κατευθύνει τα μηχανικά μέρη του Η/Υ.

Το *κέλυφος* χρησιμοποιώντας διάφορα εργαλεία του συστήματος απαντά στις εντολές του χρήστη παρέχοντας διάφορες πληροφορίες.

Το *πρόγραμμα επικοινωνίας* αποτελεί τον τρόπο επαφής του χρήστη με το λειτουργικό σύστημα.

2.2 Προσπέλαση στο Σύστημα (Logging in)

Υπάρχουν διάφοροι τρόποι για να «μπεί» κανείς στον υπολογιστή. Για παράδειγμα από την οθόνη του Η/Υ ή ακόμα από κάποιο τερματικό σταθμό που βρίσκεται γενικά μακριά από τον υπολογιστή. Για να χρησιμοποιήσει κάποιος ένα σύστημα UNIX θα πρέπει να έχει ένα *όνομα-χρήστη(username)* και ένα *κωδικό πρόσβασης(password)*. Έχοντας αυτά τα δύο μπορούμε να αποκτήσουμε πρόσβαση και κατά συνέπεια να χρησιμοποιήσουμε τον υπολογιστή. Η διαδικασία προσβάσης γίνεται ως εξής :

login : username (Πληκτρολογούμε το «όνομα-χρήστη» που μας έχει δοθεί και πατάμε το πλήκτρο RETURN ή ENTER).
password : (Πληκτρολογούμε τον κωδικό πρόσβασης και πατάμε το πλήκτρο RETURN ή ENTER. Παρατηρούμε ότι για προφανείς λόγους ασφαλείας πληκτρολογώντας τον κωδικό οι χαρακτήρες δεν εμφανίζονται στην οθόνη).

Παρατηρήσεις.

1. Αν πληκτρολογήσετε τον κωδικό πρόσβασης λάθος ή καθυστερήσετε αρκετά να τον πληκτρολογήσετε τότε ο υπολογιστής σας γυρίζει το μήνυμα : Login Incorrect(Λαθασμένη Πρόσβαση)!!! Πατάμε 1-2 φορές το RETURN και ξαναρχίζουμε πάλι από την αρχή. Αυτή την φορά πληκτρολογούμε πιό προσεκτικά τόσο το «όνομα-χρήστη» όσο και τον κωδικό πρόσβασης.

Σχήμα 2.1: Τυπική οθόνη πριν την εισαγωγή στοιχείων

2. Το UNIX είναι ευαίσθητο μεταξύ μικρών και κεφαλαίων χαρακτήρων. Έτσι οι χαρακτήρες *a* και *A* είναι τελείως διαφορετικοί, πράγμα που δεν συμβαίνει το ίδιο για άλλα λειτουργικά συστήματα όπως π.χ. το DOS.

3. Είναι χρήσιμο για λόγους ασφαλείας να αλλάζουμε password αμέσως μετά την πρώτη πρόσβαση στο σύστημα. Θα δούμε παρακάτω πως γίνεται αυτό. Φυσικά δεν θα πρέπει να ξεχάσουμε τον καινούργιο κωδικό πρόσβασης.

4. Ο κωδικός πρόσβασης password πρέπει να έχει τουλάχιστον 6 χαρακτήρες. Για λόγους ασφαλείας του συστήματος καλό θα ήταν να μην χρησιμοποιούμε λέξεις που υπάρχουν σε λεξικά. Αντίθετα θα πρέπει να χρησιμοποιούμε ένα συνδυασμό από σύμβολα, γράμματα κεφαλαία/μικρά, αριθμούς, που θα σχηματίζουν μία φράση η οποία μας θυμίζει κάτι έτσι ώστε να το θυμόμαστε σχετικά εύκολα. Βάλτε την φαντασία σας να δουλέψει αλλά με μέτρο.

5. Μετά την επιτυχή πληκτρολόγηση του κωδικού πρόσβασης ο υπολογιστής απαντά συνήθως εμφανίζοντας στη οθόνη το σύμβολο : % ή κάτι άλλο σχετικό. Το σύμβολο αυτό λέγεται *Σήμα Αναμονής* (prompt) και σημαίνει ότι το σύστημα είναι έτοιμο να δεχθεί εντολές. Υπάρχει περίπτωση προτού εμφανιστεί το σήμα αναμονής να σας εμφανιστούν στην οθόνη διάφορα μηνύματα. Ανάμεσα σε αυτά μπορεί να είναι, πότε έγινε η τελευταία επιτυχημένη είσοδο (login) στο σύστημα και πόσες αποτυχημένες προσπάθειες πρόσβασης(login) έχετε κάνει μέχρι εκείνη την στιγμή. Με αυτό τον τρόπο μπορείτε να ελέξετε αν κάποιος άλλος έχει προσπαθήσει να μπεί στον υπολογιστή χρησιμοποιώντας το δικό σας «όνομα-χρήστη» και κωδικό.

6. Για την εκτέλεση κάθε εντολής είναι απαραίτητο, αφότου την πληκτρολογήσουμε να πατήσουμε το πλήκτρο RETURN ή ENTER. Αν κατά την διάρκεια πληκτρολόγησης κάνουμε κάποιο λάθος τότε πατώντας RETURN ή ENTER θα εμφανιστεί στην οθόνη το μήνυμα :

   Command not found (Η εντολή δεν βρέθηκε).

   Πληκτρολογούμε προσεκτικά πάλι την εντολή και συνεχίζουμε. Αν αντιληφθούμε κάποιο λάθος την ώρα της πληκτρολόγησης τότε πατάμε το πλήκτρο DEL για να σβήσει ο τελευταίος χαρακτήρας ή με το πλήκτρο ← μετακινούμαστε αριστερά στο λαθασμένο(ους) χαρακτήρες και κάνουμε την διόρθωση με το πλήκτρο DEL.

Σχήμα 2.2: Οθόνη μετά την εισαγωγή του ονοματος χρήστη (username)

2.3 Αλλαγή κωδικού πρόσβασης

Η αλλαγή του κωδικού πρόσβασης γίνεται με μία από τις ακόλουθες εντολές :

passwd ή *yppasswd*.

Πληκτρολογώντας μία από τις δύο εντολές το σύστημα μας ζητά να εισάγουμε τον νέο κωδικό και κατόπιν ξαναζητά τον νέο κωδικό για να επιβεβαιώσει την αλλαγή. Αν γίνουν όλα σωστά το σύστημα μας δίνει το σήμα αναμονής. Αν κάτι πάει λάθος μας ειδοποιεί ότι η αλλαγή δεν έγινε. Τυπικό λάθος στην παραπάνω διαδικασία είναι η λαθασμένη πληκτρολόγηση του νέου *password* κατά την δεύτερη φορά όταν γίνεται η επιβεβαίωση. Σε αυτή την περίπτωση επαναλαμβάνουμε ξανά την παραπάνω διαδικασία.

2.4 Έξοδος από το σύστημα

Η έξοδος από το σύστημα γίνεται πληκτρολογώντας την εντολή :

%logout

Σε πολλές περιπτώσεις μπορούμε επίσης να βγούμε από το σύστημα δίνοντας την εντολή :

%exit

2.5 Κατάλογοι στο UNIX και το σύστημα αρχείων

2.5.1 Αρχεία

Ενα αρχείο είναι ένα μέρος της μνήμης του υπολογιστή το οποιο περιέχει μια συγκεκριμένη πληροφορία. Η πληροφορία αυτή μπορεί λ.χ. να είναι μια κωδικοποιημένη εικόνα, ένα κείμενο, ένα πρόγραμμα, κωδικοποιημένο ήχο, κ.ο.κ.. Για να μπορούμε να αναφερόμαστε στο μέρος αυτό της μνήμης του υπολογιστή (αρχείο) του δίνουμε ένα όνομα. Συνήθως επιλέγουμε το όνομα του αρχείου με τέτοιο τρόπο, ωστε από το όνομα και μόνο να γνωρίζουμε τι υπάρχει στον αντίστοιχο χώρο μνήμης (περιεχόμενο αρχείου). Για παράδειγμα, ένα κείμενο 0α μπορουσε να ονομαστεί schedule.txt, οπου η κατάληξη .txt υποδηλώνει οτι το αρχείο είναι κείμενο

Σχήμα 2.3: Οθόνη μετά την επιτυχή εισαγωγή του κωδικού πρόσβασης (password)

(text). Οι εικόνες εμφανιζονται συχνά με καταλήξεις όπως .jpg, .jpeg, .gif, .tiff, .bmp. Για παράδειγμα, η λέξη sailing.jpg με παραπέμπει να υποθέσω οτι το αρχείο είναι εικόνα, που έχει κωδικοποιηθει με τη μορφή JPEG και πιθανότατα είναι μία εικόνα από ιστιοπλοΐα.

Με τον καιρό, ο χρήστης του υπολογιστή μπορεί να δημιουργήσει τα δικά του αρχεία με τη βοήθεια διαφόρων προγραμμάτων που προσφέρονται στο λογισμικό του υπολογιστή.

2.5.2 Αρχειοθέτηση - Κατάλογοι

Υποθέστε οτι έχετε ένα σύνολο απο αρχεία τα οποία περιελάμβαναν γράμματα, σημειώσεις, εικόνες και ένα άλλο σύνολο από προγράμματα για τον υπολογιστή. Εάν ο αριθμός των αρχείων ήταν περισσότερα απο 5-6, θα ήταν φυσικό να θέλουμε να καταχωρηθούν τα αρχεία σε διαφορετικά σύνολα, ανάλογα με το περιεχόμενό τους ή την δουλειά για την οποία τα θέλουμε. Θα δημιουργούσαμε ένα συνολο με το όνομα documents όπου θα καταχωρούσαμε γράμματα και σημειώσεις, άλλο σύνολο με το όνομα images και ένα τρίτο με το όνομα programs.

Η αρχειοθέτηση αυτή δέν επηρρεάζει το περιεχόμενο των αρχείων, απλά βοηθά στην ταξινόμησή τους. Τα σύνολα αυτα των αρχείων ονομάζονται κατάλογοι. Όλα τα αρχεία του υπολογιστή ανήκουν σε κάποιο κατάλογο, και αυτός ο κατάλογος με τη σειρά του μπορεί να είναι μέρος ενός μεγαλυτερου συνόλου (καταλόγου) από αρχεία.

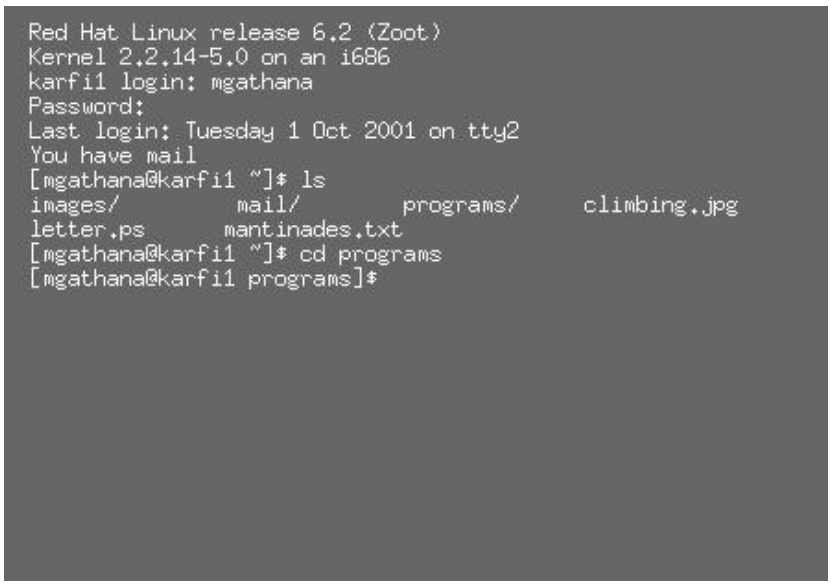
Οι κατάλογοι είναι χώροι μνήμης οι οποίοι λειτουργούν ως χώροι στους οποίους συλλέγονται αρχεία ή άλλοι κατάλογοι. Οι κατάλογοι που περίεχονται σε άλλους καταλόγους ονομάζονται υποκατάλογοι. Τη σχέση καταλόγου και υποκαταλόγου μπορούμε να τη δούμε και σαν σχέση ενός συνόλου με ένα ορισμένο υποσύνολό του.

Για να αναφερόμαστε στους καταλόγους, τους δίνουμε συγκεκριμένα ονόματα (όπως documents, programs, images).

Ενας κατάλογος, όπως είδαμε, μπορεί να περιέχει αρχεία, ή υποκαταλόγους, ή συνδυασμό αρχείων και υποκαταλόγων ή μπορεί να μην περιέχει τίποτε (κενός υποκατάλογος).

Το πώς μπορούμε να δούμε ποιά είναι τα περιεχόμενα ενός καταλόγου θα το δούμε παρακάτω.

Σε έναν υπολογιστή μπορούν να υπάρχουν δύο ή και περισσότερα αρχεία ή κατάλογοι με το ίδιο όνομα, αρκεί τα αρχεία αυτά (ή οι υποκατάλογοι) να μην βρίσκονται στον ίδιο κατάλογο. Ετσι λ.χ. μπορεί να υπάρχει ένα αρχείο με το όνομα letter.ps στον κατάλογο documents και στον καταλογο images ένα διαφορετικό αρχείο αλλά με το ίδιο όνομα letter.ps.



Σχήμα 2.4: Χρήση της εντολής ls για να δούμε τα περιεχόμενα του κυρίως καταλόγου (∼) του χρήστη mgathana και αλλαγή από τον κύριο κατάλογο στον (υπό-) κατάλογο programs (εντολή cd programs). (Σημείωση: Οι χρήστες που για πρώτη φορά προσπελαύνουν στον υπολογιστή ισως να μήν έχουν αρχεία ή υποκαταλόγους στον κύριό τους κατάλογο. Σε μια τέτοια περίπτωση η εντολή ls δεν θα επιστρέψει πληροφορία.)

2.5.3 Σύστημα καταλόγων σε έναν υπολογιστή

Όλοι οι *κατάλογοι(directories)* στο UNIX είναι *υποκατάλογοι (subdirectories)* του *βασικού(root)* καταλόγου. Ολα τα δεδομένα (data) που περιέχονται στο βασικό κατάλογο και τους υποκαταλόγους του αποθηκεύονται στο σκληρό δίσκο του συστήματος. Κάθε κατάλογος μπορεί να περιέχει άλλους υποκαταλόγους ή *αρχεία(file)*. Για κάθε υποκατάλογο υπάρχει ένα μοναδικό *μονοπάτι(path)* που ξεκινά από το βασικό υποκατάλογο και καταλήγει σε αυτό. Ο βασικός κατάλογος συμβολίζεται με το : / . Κάθε χρήστης του συστήματος έχει ένα υποκατάλογο για αποκλειστική του χρήση. Με άλλα λόγια κάθε χρήστης έχει ένα κομμάτι του σκληρού δίσκου στο οποίο μπορεί να αποθηκεύει τα αρχεία του. Ο υποκατάλογος αυτός ονομάζεται *home-directory* και συμβολίζεται με : ∼/. Έτσι κάθε φορά που ένας χρήστης μπαίνει στο σύστημα βρίσκεται αυτόματα στο προσωπικό του υποκατάλογο.

Παραδείγματα, βλέπε Σχήμα 2.6

1. /usr/tem/tem1
   Το home-directory του χρήστη tem1 είναι υποκατάλογος του tem το οποίο είναι υποκατάλογος του usr το οποίο είναι υποκατάλογος του βασικού(root) καταλόγου.

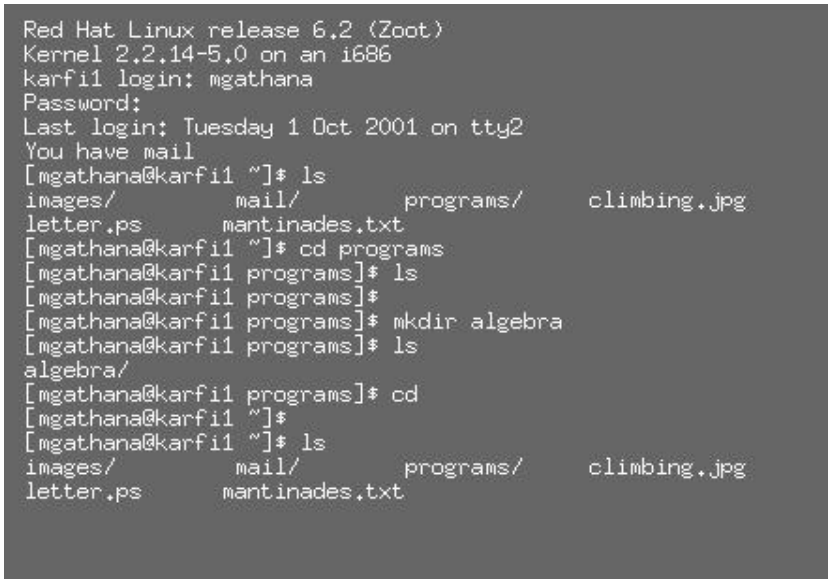
2. $\sim/em091/tests$
   Το tests είναι υποκατάλογος του em091 το οποίο είναι υποκατάλογος του home-directory του χρήστη.

3. $\sim/em091/tests/ask1.f$
   Το αρχείο ask1.f βρίσκεται στο υποκατάλογο tests του υποκαταλόγου em091 του home-directory του χρήστη.

2.6 Διαχείρηση Υποκαταλόγων και Αρχείων

Οι βασικές εντολές διαχείρησης υποκαταλόγων και αρχείων είναι οι ακόλουθες :

mkdir Δημιουργία υποκαταλόγου.



Σχήμα 2.5: Ενώ βρισκόμαστε στον κατάλογο programs, χρησιμοποιούμε την εντολή ls για να δούμε τα περιεχόμενα του (κενού) καταλόγου programs. Στην συγκεκριμένη περίπτωση, επειδή ο κατάλογος είναι κενός, η εντολή ls δεν επιστρέφει πληροφορία. Στη συνέχεια δημιουργούμε τον κατάλογο algebra - υποκατάλογο του καταλόγου programs με την εντολή mkdir algebra. Με την εντολή ls βλέπουμε οτι τώρα ο κατάλογος programs περιέχει τον κατάλογο algebra. Τέλος, με την εντολή cd επιστρέφουμε στον κυρίως κατάλογο (∼) και ξαναβλέπουμε τα περιεχόμενά του με την εντολή ls.

cd Αλλαγή υποκαταλόγου.

pwd Εμφάνιση τρέχοντος υποκαταλόγου.

rmdir Διαγραφή ενός (άδειου) υποκαταλόγου.

ls Περιεχόμενα τρέχοντος υποκαταλόγου.

mv Μετακίνηση ενός υποκαταλόγου ή αρχείου από ένα υποκατάλογο σε άλλο ή αλλαγή ονόματος αρχείου ή υποκαταλόγου.

cp Αντιγραφή αρχείου.

rm Διαγραφή αρχείου.

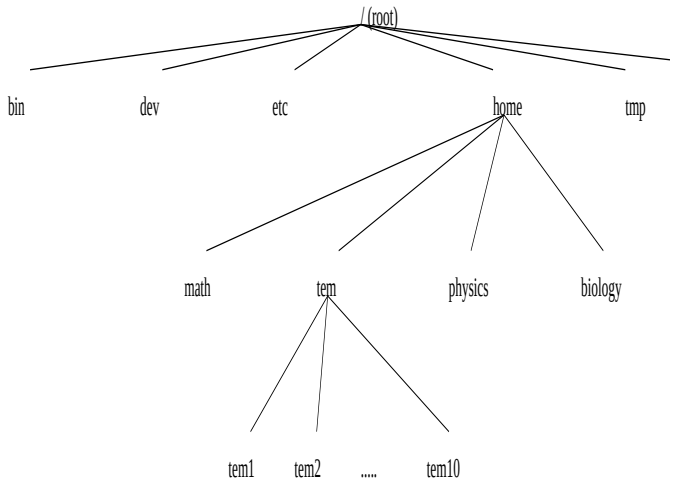
lp Εκτύπωση αρχείου.

2.6.1 Η εντολή *mkdir*

Με την εντολή *mkdir (make directory)* δημιουργούμε έναν υποκατάλογο κάτω από τον τρέχον υποκατάλογο, π.χ.

```
% mkdir tests
```

Δημιουργεί το υποκατάλογο tests στον τρέχον υποκατάλογο.



Σχήμα 2.6: Τυπικό Διάγραμμα Υποκαταλόγων

2.6.2 Η εντολή *cd*

Με την εντολή *cd (change directory)* μετακινούμαστε από έναν υποκατάλογο στον άλλο, π.χ.

%cd

Επιστροφή στο βασικό μας υποκατάλογο(home-directory)

%cd ..

Μετακινούμαστε προς τα «πάνω». Δηλαδή μετακινούμαστε στο υποκατάλογο από τον οποίο κρέμεται ο τρέχον υποκατάλογος.

%cd ∼/tests/em091

Αλλάζουμε υποκατάλογο και μεταφερόμαστε στον υποκατάλογο em091 του υποκαταλόγου tests στο βασικό μας υποκατάλογο.

2.6.3 Η εντολή *pwd*

Η εντολή *pwd (print working directory)* μας εμφανίζει στην οθόνη τον τρέχοντα υποκατάλογο, π.χ.

%pwd
/home/tem/tem1

2.6.4 Η εντολή *rmdir*

Η εντολή *rmdir (remove directory)* διαγράφει έναν υποκατάλογο υπό την προυπόθεση ότι είναι άδειος, π.χ.

%rmdir tests
Directory is not empty

Σε αυτή την περίπτωση δεν μπορούμε να σβήσουμε τον υποκατάλογο tests διότι δεν είναι άδειος, δηλαδή περιέχει αρχεία και άλλους υποκαταλόγους. Αν ο υποκατάλογος tests ήταν άδειος τότε η παραπάνω εντολή θα έσβηνε τον υποκατάλογο tests.

2.6.5 Η εντολή *ls*

Η εντολή *ls (listing)* μας δίνει το περιεχόμενο του τρέχοντος υποκαταλόγου, π.χ.

%ls

mbox tests ex.f a.out

%ls -F

mbox/ tests/ ex.f a.out*

%ls -aF

./ ../ .cshrc .login mbox/ tests/ ex.f a.out*

Η ενολή αυτή μας παρέχει πολλές χρήσιμες πληροφορίες ανάλογα με τις παραμέτρους που βάζουμε. Τον πλήρη κατάλογο των παραμέτρων μπορούμε να τον δούμε δίνοντας την εντολή :
%man ls

2.6.6 Η εντολή *mv*

Η εντολή *mv (move)* μετακινεί ένα αρχείο ή υποκατάλογο από ένα υποκατάλογο σε άλλο ή αλλάζει το όνομα ενός αρχείου, π.χ.

%mv ex1.f tests

Μεταφέρει το αρχείο ex1.f στον υποκατάλογο tests.

%mv ex1.f ask1.f

Αλλάζει το ονόμα του αρχείου ex1.f σε ask1.f

2.6.7 Η εντολή *cp*

Η εντολή *cp (copy)* αντιγράφει ένα αρχείο ή υποκατάλογο π.χ.

%cp ex1.f tests

Αντιγράφει το αρχείο ex1.f στον υποκατάλογο tests.

%cp ex1.f ask1.f

Αντιγράφει το αρχείο ex1.f στο ask1.f

2.6.8 Η εντολή *rm*

Με την εντολή *rm (remove)* διαγράφουμε(σβήνουμε) ένα αρχείο π.χ.

%rm ex1.f

Διαγράφει το αρχείο ex1.f από τον τρέχον υποκατάλογο.

%rm *.f

Διαγράφει ΟΛΑ τα αρχεία με κατάληξη f

%rm -i ex1.f

Διαγράφει το αρχείο ex1.f αφού πρώτα μας ρωτήσει εάν θέλουμε να το σβήσουμε η όχι.

2.6.9 Η εντολή *lop , lpr*

Οι δύο αυτές εντολές εκτυπώνουν ένα αρχείο στον εκτυπωτή του Η/Υ, π.χ.

%lp ex1.f

Τυπώνει το αρχείο ex1.f στον εκτυπωτή.

2.7 Εμφάνιση και Τροποποίηση Αρχείων

more , cat Εμφάνιση αρχείων δίχως την δυνατότητα τροποποίησης τους.

diff , cmp Σύγκριση δύο αρχείων.

vi Άνοιγμα αρχείου με δυνατότητα τροποποίησης.

2.7.1 Οι εντολές *more , cat*

Οι δύο αυτές εντολές μας δίνουν την δυνατότητα να δούμε το περιεχόμενο ενός αρχείου χωρίς όμως να μπορούμε να το τροποποιήσουμε, π.χ.

%more ex1.f

Εμφανίζει το αρχείο ex1.f στον οθόνη. Στη περίπτωση που δεν χωρά ολόκληρο στη οθόνη τότε πατώντας το πλήκτρο RETURN , ENTER συνεχίζουμε γραμμή-γραμμή. Αν πατήσουμε το κενό (SPACEBAR) προχωρούμε ανά οθόνη. Πατώντας το q σταματάει η ροή της εμφάνισης.

%cat ex1.f

Εμφανίζει το περιεχόμενο του αρχείου ex1.f στην οθόνη αλλά με συνεχή ροή, χωρίς να σταματά μόλις γεμίσει η οθόνη.

2.7.2 Οι εντολές *cmp , diff*

Οι εντολές cmp (compare) και diff (difference) συγκρίνουν χαρακτήρα προς χαρακτήρα δύο αρχεία και μας δίνουν την πρώτη διαφορά που θα βρεθεί ανάμεσα στα δύο αρχεία. Αυτό μπορεί να φανεί ιδιαίτερα χρήσιμο όταν συγκρίνουμε αρχεία δεδομένων, π.χ.

%cmp ex1.f ask1.f

Συγκρίνει τα δύο αρχεία ex1.f ask1.f.

2.7.3 Η εντολή *vi*

Η εντολή αυτή ανοίγει ένα αρχείο και μας τοποθετεί στο επίπεδο εντολών του κειμενογράφου (editor) *vi*. Ο εκδότης αυτός είναι ο πιό διαδεδομένος και υπάρχει σε όλα τα συστήματα UNIX. Φυσικά υπάρχουν και πολλοί άλλοι και σίγουρα πιό φιλικοί προς το χρήστη με τους οποίους όμως δεν θα ασχοληθούμε προς το παρόν. Η εντολή,

%vi ex1.f

Ανοίγει το αρχείο ex1.f. Αν το αρχείο δεν υπάρχει ήδη τότε δημιουργείται ένα νέο με το όνομα ex1.f, και μπαίνουμε πλέον στο επίπεδο εντολών του *vi*. Περισσότερες πληροφορίες σχετικά με τον *vi* δίνονται παρακάτω σε ξεχωριστή παράγραφο.

2.8 Παίρνοντας Πληροφορίες

man Δίνει πληροφορίες για κάποιο θέμα ή εντολή.

who Εμφάνιση των χρηστών του συστήματος.

whoami Εμφανίζει το όνομα του χρήστη.

date Δίνει την τρέχουσα ημερομηνία και ώρα.

cal Δίνει το ημερολόγιο του τρέχοντα μήνα.

2.8.1 Η εντολή *man*

Η εντολή αυτή δίνει πληροφορίες για κάποιο θέμα ή εντολή, π.χ.

%man cd

Παίρνουμε όλες τις πληροφορίες σχετικά με την εντολή *cd*.

%man -k print

Μας δίνει όλες τις βοηθητικές σελίδες που περιέχουν την λέξη print.

2.8.2 Οι εντολές *who , whoami*

Η εντολή *who* μας εμφανίζει στη οθόνη τους χρήστες του συστήματος εκείνη τη στιγμή. Από την άλλη μεριά η εντολή *whoami* εμφανίζει στην οθόνη το όνομα του χρήστη.

2.8.3 Οι εντολές *date , cal*

Με την εντολή *date* το σύστημα μας επιστρέφει την τρέχουσα ημερομηνία και ώρα. Η εντολή *cal* μας δίνει το ημερολόγιο του τρέχοντος μηνός.

%date

Μας εμφανίζει : Wed Aug 11 12:52:32 1999

%cal 8 1999

Μας εμφανίζει το ημερολόγιο του μηνός Αυγούστου του έτους 1999.

2.9 Το Ηλεκτρονικό Ταχυδρομείο e-mail

Η εντολή mail μας δίνει πρόσβαση στη ηλεκτρονικό ταχυδρομείο του συστήματος. Με την εντολή αυτή μπορούμε να στείλουμε και να λάβουμε μηνύματα από κάποιο άλλο χρήστη. Περά από το mail υπάρχουν και άλλα εργαλεία διαχείρησης του ηλεκτρονικού ταχυδρομείου.

2.9.1 Αποστολή μηνύματος

Βασική προυπόθεση για την αποστολή μηνύματος σε κάποιο άλλο χρήστη του συστήματος μας ή κάποιου απομακρυσμένου συστήματος είναι η γνώση της ηλεκτρονικής του διευθύνσης. Η ηλεκτρονική διεύθυνση αποτελείται από δύο μέρη : α) Το *όνομα-χρήστη(username)* και β) την *διεύθυνση* του συστήματος του χρήστη, π.χ. η διεύθυνση,

tem091@tem.uch.gr αναφέρεται στο χρήστη tem091 του συστήματος tem.uch.gr.

Η αποστολή κάποιου μηνύματος π.χ. στο χρήστη tem091@tem.uch.gr γίνεται δίδοντας την παρακάτω εντολή :

mail tem091@tem.uch.gr

Μας ζητείται πρώτα ο τίτλος του μηνύματος και κατόπιν δακτυλογραφούμε το περιεχόμενο του μηνύματος και όταν τελειώσουμε πατάμε RETURN ή ENTER τουλάχιστο μία φορά και κατόπιν δίνουμε Ctrl-D, δηλαδή έχοντας πατημένο το πλήκτρο Ctrl πατάμε ταυτόχρονα το πλήκτρο D. Το μήνυμα στάλθηκε !!!

2.9.2 Ανάγνωση μηνυμάτων

Για να διαβάσουμε τα μηνύματα τα οποία μας έχουν στείλει δίνουμε απλά την εντολή :
%mail
Στην οθόνη εμφανίζεται μια λίστα με τα μηνύμητα που έχουμε στο «γραμματοκιβώτιο» μας και μπαινουμε στο επίπεδο εντολών του mail με σήμα αναμονής & ή ?. Η λίστα που εμφανίζεται στη οθόνη μας παρέχει διάφορες πληροφορίες όπως ο αριθμός του μηνύματος, ο αποστολέας, η ημερομηνία άφιξης και ο τίτλος του. Για να διαβάσουμε κάποιο από τα μηνύματα δίνουμε απλά τον αριθμό του. Το επίπεδο εντολών του mail μας παρέχει πληθώρα άλλων εντολών τις οποίες μπορούμε να δούμε δίνοντας help. Μερικές από τις πιό βασικές είναι οι εξής :

?*h* Εμφανίζει τις επικεφαλίδες των μηνυμάτων.

?*r* Απαντούμε στο τρέχον μήνυμα.

?*d* Διαγράφουμε στο τρέχον μήνυμα.

?*s* Σώζουμε το τρέχον μήνυμα σε ένα αρχείο.

?*q* Βγαίνουμε από το mail σώζοντας ότι αλλαγές έχουμε κάνει.

?*x* Βγαίνουμε από το mail χωρίς να σώσουμε τις αλλαγές που έχουμε κάνει.

2.10 Άλλες χρήσιμες εντολές

clear «Καθαρισμός» οθόνης.

history Εμφανίζει τις τελευταίες εντολές που έχουμε δώσει.

! Επανάληψη προηγούμενων εντολών.

2.10.1 Η εντολή *clear*

Η εντολή *clear* καθαρίζει την οθόνη και μεταφέρει το σήμα αναμονής στη κορυφή του παράθυρου εργασίας.

2.10.2 Η εντολή *history*

Η εντολή αυτή μας παρέχει μία λίστα με τις εντολές που έχουμε δώσει πρόσφατα στο παράθυρο εργασίας.

2.10.3 Η εντολή *!*

Με την εντολή αυτή μπορούμε να επαναλάβουμε την τελευταία εντολή που αρχίζει με κάποιο συγκεκριμμένο χαρακτήρα ή αριθμό από την λίστα εντολών που μας παρέχει η εντολή *history*.

%!ma Επαναλαμβάνει την τελευταία εντολή που αρχίζει από ma, π.χ. %man -k f77.
%!12 Επαναλαμβάνει την εντολή με αριθμό 12 από την λίστα εντολών που μας παρέχει η εντολή *history*.
%!! Επαναλαμβάνει την τελευταία εντολή που δώσαμε στο παράθυρο εργασίας.

2.11 Προσπέλαση σε άλλα συστήματα UNIX

rlogin Πρόσβαση σε άλλο σύστημα από μακριά.

telnet Πρόσβαση σε άλλο σύστημα από μακριά.

setenv DISPLAY Επιτρέπει σε άλλους υπολογιστές πρόσβαση σε κάποιον συγκεκριμμένο υπολογιστή.

xhost+ Προσθέτει άλλους υπολογιαστές στη λίστα επιτρεπόμενης πρόσβασης.

ftp Πρωτόκολλο μεταφοράς αρχείων.

2.11.1 Οι εντολές *rlogin , telnet*

Με τις εντολές αυτές αποκτούμε πρόσβαση σε υπολογιστή που βρίσκεται μακριά από εμάς, π.χ.

%rlogin ikaros.edu.uch.gr

Αποκτούμε πρόσβαση στο σύστημα με όνομα ikaros που βρίσκεται στο δύκτιο edu.uch.gr. Αν το σύστημα βρίσκεται στο ίδιο δύκτιο με το δικό μας γράφουμε απλά το όνομα του υπολογιστή. Ισοδύναμα μπορούμε να δώσουμε :

%telnet ikaros.edu.uch.gr

2.11.2 Οι εντολές *setenv DISPLAY & xhost*

Ο συνδυασμός αυτών των δύο εντολών μας επιτρέπει, όταν βρισκόμαστε στο παραθυρικό περιβάλλον, να δουλεύουμε σε κάποιο απομακρυσμένο υπολογιστή και να βλέπουμε όλες τις παραθυρικές εφαρμογές στο τερματικό μας. Η εντολή setenv DISPLAY δίνεται σε παράθυρο του απομακρισμένου υπολογιστή, ενώ η εντολή xhost+ πληκτρολογείται σε παράθυρο του υπολογιστή που βρισκόμαστε, π.χ. από ένα παράθυρο του υπολογιστή με όνομα ikaros πληκτρολογούμε :

%setenv DISPLAY athina.edu.uch.gr:0.0

ενώ σε ένα παράθυρο του ikaros δίνουμε :

%xhost +athina.edu.uch.gr

Οπότε αν είμαστε σε ένα σταθμό εργασίας με όνομα athina και δουλεύουμε στον ikaros τότε όλες οι παραθυρικές εφαρμογές θα εμφανίζονται στην athina.

2.11.3 Η εντολή *ftp (File Transfer Protocol)*

Η εντολή αυτή μας μεταφέρει στο Πρωτόκολλο Μεταφοράς Αρχείων (ΠΜΑ) το οποίο επιτρέπει την μεταφορά αρχείων από και πρός άλλους υπολογιστές. Στο ΠΜΑ μπαίνουμε δίνοντας :

%ftp Όνομα υπολογιστή , π.χ.

%ftp athina.edu.uch.gr

Μπαίνουμε έτσι στο ΠΜΑ. Θα μας ζητηθεί το ονόμα χρήστη καθώς και ο κωδικός πρόσβασης. Από την στιγμή που βρισκόμαστε στο ΠΜΑ μπορούμε να εκτελέσουμε κάποιες απο τις συνήθεις εντολές του UNIX όπως : pwd, ls, cd . Αν θέλουμε να πάρουμε κάποιο αρχείο τότε δίνουμε την εντολή :

%ftp get Όνομα αρχείου

ενώ αν θέλουμε να μεταφέρουμε ένα αρχείο δίνουμε

%ftp put Όνομα αρχείου

Φεύγουμε από το ΠΜΑ με την εντολή

%ftp quit

2.12 Ο κειμενογράφος (editor) vi

Ο κειμενογράφος είναι ένα πρόγραμμα που μας βοηθά στη δημιουργία αρχείων. Στα σύγχρονα παραθυρικά περιβάλλοντα μπορεί να βρεί κανείς κειμενογράφους πολύ ευχρήστους και φιλικούς προς το χρήστη. Ο κειμενογράφος vi είναι από τους πιό διαδεδομένους στα συστήματα UNIX. Θα πρέπει να σημειώσουμε όμως ότι δεν είναι πολύ φιλικός προς το χρήστη, όμως αποτελεί θα λέγαμε αναποσπαστό κομμάτι του λειτουργικού συστήματος και μιά συντομή εισαγωγή στις βασικές του εντολές κρίνεται απαραίτητη.

Ξεκινούμε με μια σημαντική παρατήρηση που θα πρέπει να θυμόμαστε, οι εντολές του vi είναι ευαίσθητες στα μικρά και κεφαλαία γράμματα. Στον vi υπάρχουν δύο διαφορετικά επίπεδα α) το επίπεδο εντολών (ΕΕ) και β) το επίπεδο εισαγωγής κειμένου (ΕΕΚ). Στο ΕΕ χρησιμοποιούμε βασικά εντολές για να μετακινούμαστε μέσα στο κείμενο, να βρούμε λέξεις, να σώσουμε ένα αρχείο κ.λ.π. Το ΕΕΚ χρησιμοποιείται αποκλειστικά και μόνο για την εισαγωγή κειμένου.

Χρησιμοποιώντας : I, i, A, a, O, o, cw, R μπαίνουμε στο ΕΕΚ από το ΕΕ. Οι εντολές αυτές θα περιγραφούν εκτενέστερα παρακάτω. Χρησιμοποιούμε το πλήκτρο **ESC** για να μεταφερθούμε από το ΕΕΚ στο ΕΕ. Αν δεν είμαστε σίγουροι σε ποιό επίπεδο εντολών βρισκόμαστε πατάμε το πλήκτρο **ESC** για να βρέθουμε στο ΕΕ. Αν πατώντας κάποια πλήκτρα δεν πραγματοποιούνται αυτά που θα επρέπε να γίνονται τότε βεβαιωθείτε ότι δεν είναι πατημένο το πληκτρο **CAPS LOCK**. Επίσης δεν θα πρέπει να ξεχνάμε ότι δεν μπορούμε να χρησιμοποιήσουμε τα βελάκια για να μετακινηθούμε μέσα στο αρχείο ενώ βρισκόμαστε στο ΕΕΚ.

2.13 Χρήσιμες εντολές του vi

2.13.1 Μετακίνηση μέσα στο αρχείο στο ΕΕ

$\leftarrow, \rightarrow, \uparrow, \downarrow$	Μετακίνηση μέσα στο αρχείο : αριστέρα, δεξιά, πάνω, κάτω.
h,l,k,j	Μετακίνηση μέσα στο αρχείο : αριστέρα, δεξιά, πάνω, κάτω.
Ctrl-f	Προχωρά μιά σελίδα μπροστά.
Ctrl-b	Προχωρά μιά σελίδα πίσω.
Ctrl-d	Προχωρά μισή σελίδα μπροστά.
Ctrl-u	Προχωρά μισή σελίδα πίσω.
G	Μετακίνηση στο τέλος του αρχείου.
65G	Μετακίνηση στην 65η γραμμή του αρχείου.
$	Μετακίνση στο τέλος της τρέχουσας γραμμής.
0(Μηδέν)	Μετακίνση στη αρχή της τρέχουσας γραμμής.
b	Μετακίνση στη αρχή της τρέχουσας λέξης.
e	Μετακίνση στο τέλος της τρέχουσας λέξης.
/Λέξη	Αναζήτηση προς τα εμπρός της «Λέξης».
?Λέξη	Αναζήτηση προς τα πίσω της «Λέξης».
n	Επόμενη εμφάνιση της «Λέξης» με την τρέχουσα φορά.
N	Επόμενη εμφάνιση της «Λέξης» με την αντίθετη φορά.

2.13.2 Εισαγωγή κειμένου

i	Εισαγωγή κειμένου πριν από την τρέχουσα θέση.
I	Εισαγωγή κειμένου στη αρχή της τρέχουσας γραμμής.
a	Εισαγωγή κειμένου μετά από την τρέχουσα θέση.
A	Εισαγωγή κειμένου μετά το τέλος της τρέχουσας γραμμής.
o	Άνοιγμα καινούργιας γραμμής κάτω από την τρέχουσα και εισαγωγή κειμένου.
O	Άνοιγμα καινούργιας γραμμής πάνω από την τρέχουσα και εισαγωγή κειμένου.

Οι παραπάνω εντολές μας τοποθετούν στο ΕΕΚ. Πατώντας **ESC** επιστρέφουμε στο ΕΕ.

2.13.3 Σβήσιμο κειμένου

x	Σβήσιμο του τρέχοντα χαρακτήρα.
7x	Σβήσιμο επτά χαρακτήρων.
dw	Σβήσιμο μιας λέξης.
10dw	Σβήσιμο 10 λέξεων.
dd	Σβήσιμο μίας γραμμής.
25dd	Σβήσιμο 25 γραμμών.
D	Σβήσιμο από την τρέχουσα θέση ως το τέλος της γραμμής.

Σημείωση. Όλες οι παραπάνω εντολές αποθηκεύουν το κείμενο ή χαρακτήρες που έχουμε σβήσει σε μια προσωρινή μνήμη. Το περιεχόμενο αυτής της μνήμης μπορεί να ανακτηθεί χρησιμοποιώντας τις εντολές P, p (βλ. παρακάτω).

2.13.4 Μετακινώντας κείμενο

yy	Αντιγράφει μία γραμμή σε προσωρινή μνήμη χωρίς να σβήσει την γραμμή αυτή.
3yy	Αντιγράφει 3 γραμμές σε προσωρινή μνήμη χωρίς να σβήσει τις γραμμές αυτές.
p	Μεταφέρει το περιέχομενο της προσωρινής μνήμης μετά την τρέχουσα γραμμή.
P	Μεταφέρει το περιέχομενο της προσωρινής μνήμης πριν την τρέχουσα γραμμή.
J	Ενώνει την γραμμή μέτα την τρέχουσα γραμμή με την τρέχουσα γραμμή.

Συνήθως χρησιμοποιούμε τις εντολές yy και p όταν θέλουμε να αντιγράψουμε κείμενο από ένα μέρος του αρχείου σε άλλο. Όταν θέλουμε να μετακινήσουμε κείμενο τότε χρησιμοποιούμε τις εντολές dd και p.

2.13.5 Αλλάζοντας το κείμενο

~	Αλλάζει από κεφαλαία σε μικρά και αντίθετα ένα χαρακτήρα.
5~	Αλλάζει από κεφαλαία σε μικρά και αντίθετα 5 χαρακτήρες.
r	Αντικαθηστεί ένα χαρακτήρα.
R	Αντικαθηστεί όλους τους χαρακτήρες μέχρι να πατήσουμε ESC.
cw	Αλλάζει μια λέξη με ότι πληκτρολογήσουμε από τη στιγμή που θα δώσουμεcw και ESC.

2.13.6 Σώζοντας το αρχείο

:w	Σώζει το αρχείο αλλά παραμένει στο vi.
:w όνομα	Σώζει το τρέχον αρχείο με καινούργιο όνομα.
:wq	Σώζει το αρχείο και βγαίνει από το vi.
:q	Φεύγει από το vi όταν δεν έχουμε κάνει αλλάγες.
:q!	Φεύγει από το vi δίχως να σώσει καμμία αλλαγή.

2.13.7 Σε περίπτωση λάθους

u	Αναιρεί την αμέσως προηγούμενη εντολή.
U	Επαναφέρει την τρέχουσα γραμμή.
Ctrl-q	Επαναφορά όταν καμμιά εντολή δεν δουλεύει.
Ctrl-l	Ανανεώση της οθόνης.

2.13.8 Αναζήτηση/Αντικατάσταση κειμένου

Με την παρακάτω εντολή έχουμε την δυνατότητα να αντικαταστήσουμε ένα συγκεκριμμένο συνδυασμό χαρακτήρων, όπου αυτός εμφανίζεται στο κείμενο, με κάποιο άλλο.

```
:.,$s/παλιά-λέξη/νέα-λέξη/g
```

Η πολύ σημαντική αυτή εντολή ξεκινά από την τρέχουσα θέση (.) ως το τέλος ($) του κειμένου αντικαθιστά (s) την «παλιά-λέξη» με την «νέα-λέξη» όπου και αν αυτή εμφανίζεται (g). Αν θέλαμε να γίνει η αναζήτηση/αντικατάσταση ανάμεσα σε συγκεκριμμένες γραμμές του κειμένου τότε απλά θα δίναμε, π.χ.

:10,37s/παλιά-λέξη/νέα-λέξη/g

η οποία αναζητά/αντικαθιστά την «παλιά-λέξη» με την «νέα-λέξη» μεταξύ των γραμμών 10 εως 37.

Κλείνοντας την παραγραφό για τον «εκδότη» vi επισημαίνουμε ότι ο καλύτερος τρόπος για να μάθει κανείς το vi είναι η **ΠΡΑΚΤΙΚΗ ΕΞΑΣΚΗΣΗ**.

2.14 Μεταγλώττιση στο UNIX

Ο υπολογιστής «μιλάει» και «καταλαβαίνει» την δική του γλώσσα, αλλά δεν μπορεί να καταλάβει την δική μας γλώσσα. Δεν είναι λίγοι εκείνοι που δεν «μιλούν» την γλώσσα που καταλαβαίνει ο υπολογιστής. Οπότε προγραμματίζουμε σε μια γλώσσα *ανωτέρου επιπέδου*, όπως FORTRAN, C, PASCAL την οποία εμείς καταλαβαίνουμε και μετά με την χρήση του μεταγλωττιστή(compiler) μεταφράζουμε ότι έχουμε γράψει σε μια μορφή την οποία καταλαβαίνει ο υπολογιστής. Η έκδοση αυτή του κώδικα ονομάζεται *αντικειμενική* και έχει την μορφή :*όνομα-αρχείου.o*.

Τυχόν συντακτικά λάθη που μπορούν να υπάρχουν ανιχνεύονται από τον μεταγλωττιστή κατά την φάση της δημιουργίας του *αντικειμενικού* κώδικα. Τα λάθη εμφανίζονται στη οθόνη. Μερικοί μεταγλωττιστές βρίσκουν ακριβώς που είναι τα λάθη, άλλοι απλώς βρίσκουν τα λάθη δίχως όμως να δίνουν πολλές επεξηγηματικές πληροφορίες. Μετά την δημιουργία της αντικειμενικής μορφής του κώδικα ο μεταγλωττιστής συνδέει τον κώδικα με τις απαραίτητες βιβλιοθήκες (π.χ. μαθηματικές) παράγοντας ένα νέο αρχείο το λεγόμενο *εκτελέσιμο* το οποίο μπορεί πλέον να *τρέξει (εκτελεστεί)* στο υπολογιστή.

Με την παρακάτω εντολή καλούμε τον μεταγλωττιστή της FORTRAN για να μεταφράση το πρόγραμμα ask1.f, παράγεται ο αντικειμενικός κώδικας ask1.o και ο εκτελέσιμος ask1,

%f77 -o ask1 ask1.f

Εκτελούμε τον κώδικα δίνοντας απλά

%ask1

Εναλλακτικά θα μπορούσαμε να είχαμε δώσει,

%f77 ask1.f

τότε ο μεταγλωττιστής θα παρήγαγε τον ίδιο αντικειμενικό κώδικα ask1.o αλλά το εκτελέσιμο θα είχε το όνομα a.out. Τότε η εκτέλεση θα γινόταν δίνοντας

%a.out

Κεφάλαιο 3

ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ FORTRAN

3.1 Εισαγωγή

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

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

3.2 Τα βασικά

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

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

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

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

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

```
      program name

      declarations

      statements

      stop
      end
```

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

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

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

Στήλη 6η : Χαρακτήρας συνέχειας της προηγούμενης γραμμής.

Στήλες 7η-72η : Εντολές του προγράμματος.

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

3.2.1 Σχόλια

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

3.2.2 Συνέχιση γραμμής

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

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

3.2.3 Κενά

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

3.3 Μεταβλητές, Τύποι μεταβλητών, Δηλώσεις

3.3.1 Ονόματα μεταβλητών

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

3.3.2 Τύποι, Δηλώσεις

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

integer	Λίστα μεταβλητών
real	Λίστα μεταβλητών
complex	Λίστα μεταβλητών
logical	Λίστα μεταβλητών
character	Λίστα μεταβλητών

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

3.3.3 Ακέραιες μεταβλητές, Μεταβλητές κινητής υποδιαστολής

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

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

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

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

3.3.4 Η εντολή *parameter*

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

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

Το *συντακτικό* της εντολής είναι :

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

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

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

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

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

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

3.4 Εκφράσεις, Αναθέσεις

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

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

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

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

Λογικές Σταθερές: Οι λογικές σταθερές παίρνουν μόνο δύο τιμές :

.TRUE. (Αληθές)
.FALSE. (Ψευδές)

ΠΡΟΣΟΧΗ: Οι τελείες που περικλείουν τις τιμές είναι απαραίτητες.

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

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

3.4.1 Εκφράσεις

Η απλούστερη έκφραση έχει την μορφή :

Όρισμα Τελεστής Όρισμα π.χ. *x+y.*

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

x+2*y.

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

** Ύψωση σε δύναμη.

*,/ Πολλαπλασιασμός, Διαίρεση.

+,- Πρόσθεση, Αφαίρεση.

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

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

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

-x+2*a.

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

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

3.4.2 Ανάθεση

Η ανάθεση έχει την μορφή :

όνομα-μεταβλητής=έκφραση

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

area=pi*r**2.

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

3.4.3 Αλλαγή Τύπου

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

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

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

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

π.χ.

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

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

3.5 Λογικές Εκφράσεις

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

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

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

.AND.	Λογικό «ΚΑΙ».
.OR.	Λογικό «Η».
.NOT.	Λογικό «ΟΧΙ».

3.5.1 Λογικές Μεταβλητές και Αναθέσεις

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

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

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

3.6 Η εντολή IF

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

if (Λογική Έκφραση) Εκτελέσιμη Εντολή

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

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

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

if (Λογική Έκφραση) then
Εκτελέσιμες Εντολές
endif

Η πιο γενική μορφή της εντολής είναι :

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

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

3.6.1 Φωλιασμένες εντολές IF

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

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

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

3.7 Ανακυκλώσεις, (LOOPS)

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

3.7.1 Ανακυκλώσεις do

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

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

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

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

Η γενική μορφή μιας ανακύκλωσης do είναι η παρακάτω :

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

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

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

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

3.7.2 Ανακυκλώσεις *While*

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

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

ή εναλλακτικά

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

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

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

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

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

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

3.8 Διανύσματα, Πίνακες

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

3.8.1 Διανύσματα

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

```
real vec(20)
```

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

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

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

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

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

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

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

3.8.2 Πίνακες

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

real aa(3,5)

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

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

με συνολικό αριθμό στοιχείων

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

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

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

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

3.8.3 Αποθήκευση πινάκων στη FORTRAN 77

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

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

3.8.4 Πίνακες Πολλαπλών Διαστάσεων

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

3.8.5 Η εντολή dimension

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

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

είναι ισοδύναμες με

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

3.9 Υποπρογράμματα

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

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

3.9.1 Συναρτήσεις(FUNCTIONS)

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

```
x=cos(pi/3.0)  .
```

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

abs Απόλυτη τιμή.

min Μικρότερη τιμή.

max Μεγαλύτερη τιμή.

sqrt Τετραγωνική ρίζα.

sin Ημίτονο.

cos Συνημίτονο.

tan Εφαπτομένη

asin Τόξο ημιτόνου.

acos Τόξο συνημιτόνου.

atan Τόξο εφαπτομένης.

exp Εκθετικό (φυσικό).

log Λογάριθμος (φυσικός).

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

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

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

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

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

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

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

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

Η αντίστοιχη συνάρτηση στη FORTRAN είναι :

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

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

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

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

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

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

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

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

3.9.2 Υπορουτίνες(SUBROUTINES)

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

subroutine όνομα(λίστα-ορισμάτων)
δηλώσεις
εντολές
return
end

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

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

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

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

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

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

3.10 Εντολές Εισόδου/Εξόδου

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

3.10.1 Οι εντολές *READ, WRITE, PRINT*

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

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

Τα *αρχείο* και *τρόπος* είναι φυσικοί αριθμοί και καθορίζουν ο μεν την μονάδα αρχείου από/στην οποία θα διαβαστούν/αποθηκευτούν οι μεταβλητές, και ο άλλος την μορφή με την οποία θα διαβαστούν/αποθηκευτούν οι μεταβλητές. Περισσότερες λεπτομέρειες δίνονται στη επόμενη παράγραφο.

Είναι δυνατό να απλοποιήσουμε ακόμα περισσότερο τις εντολές αυτές χρησιμοποιώντας αστερισκούς (*) τόσο για το *αρχείο* όσο και για τον *τρόπο*,

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

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

read*, λίστα-μεταβλητών

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

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

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

3.11 Η εντολή FORMAT

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

```
        read(* , label1) λίστα-μεταβλητών
label1 format( κώδικας )
        write(* , label2) λίστα-μεταβλητών
label2 format( κώδικας )
```

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

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

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

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

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

3.11.1 Τρόποι Μορφοποίησης

Οι πιό συνήθεις τρόποι μορφοποίησης είναι :

A Για ορμαθούς χαρακτήρων.

D Για πραγματικούς αριθμούς διπλής ακρίβειας, εκθετική αναπαράσταση.

E Για πραγματικούς αριθμούς, εκθετική αναπαράσταση.

F Για πραγματικούς αριθμούς, μορφή σταθερής υποδιαστολής.

I Για ακεραίους.

X Ένα κενό.

/ Αλλαγή γραμμής.

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

Παραδείγματα Οι παρακάτω εντολές

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

θα δώσουν τα εξής αποτελέσματα

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

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

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

θα δώσει

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

3.11.2 Έμμεσες Ανακυκλώσεις

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

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

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

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

το οποίο μπορεί να συντομευτεί και να γραφτεί ως

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

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

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

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

3.12 Είσοδος/Έξοδος Αρχείων

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

3.12.1 Ανοίγοντας/κλείνοντας ένα αρχείο

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

open(Λίστα - παραμέτρων)

όπου οι συνήθεις παράμετροι είναι :

UNIT= Αριθμός Μονάδος.

FILE= Όνομα Αρχείου.

STATUS= Κατάσταση Αρχείου.

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

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

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

close(Αριθμός - Μονάδας)

π.χ.

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

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

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

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

3.13 Πίνακες σε υποπρογράμματα

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

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

3.13.1 Διανύσματα/Πίνακες Μεταβλητής Διάστασης

Η βασικότερη ίσως πράξη μεταξύ διανυσμάτων είναι :

y = a * x + y

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

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

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

real x(n), y(n)

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

real x(1), y(1)

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

3.13.2 Υποπίνακες σε υπορουτίνες

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

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

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

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

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

3.14 Η εντολή COMMON

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

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

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

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

Το γενικό *συντακτικό* της εντολής είναι το ακόλουθο,

common/όνομα/λίστα-μεταβλητών

Θα πρέπει να γνωρίζουμε ότι

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

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

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

Σαν συνέχεια του προηγούμενου παραδείγματος έχουμε :

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

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

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

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

3.14.1 Πίνακες στη εντολή common

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

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

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

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

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

3.15 Η εντολή DATA

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

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

π.χ.

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

το οποίο μπορεί να γραφτεί και ως εξής

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

Η παραπάνω εντολή είναι ισοδύναμη με τις παρακάτω καταχωρήσεις:

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

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

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

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

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

η ακόμα όταν η διάσταση του πίνακα είναι μικρή

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

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

Κεφάλαιο 4

Ο στοιχειοθέτης LaTeX

4.1 Γενικά

4.1.1 Τι είναι το TeX και το LaTeX

Το TeX είναι μία γλώσσα προγραμματισμού η οποία δημιουργήθηκε από τον Donald E. Knuth το 1977 με βασικό σκοπό την παραγωγή κειμένων υψηλής τυπογραφικής ποιότητας. Η έκδοση του TeX που χρησιμοποιείτε σήμερα χρονολογείται από το 1982. Από τότε έχουν γίνει σχετικά λίγες και μικρές βελτιώσεις. Το TeX είναι πολύ ευσταθές και όπως ισχυρίζεται και ο ίδιος ο D. E. Knuth δεν έχει καθόλου λάθη. Η τρέχουσα έκδοση είναι η 3.14159 και συγκλίνει στο π. Το όνομα TeX, όπως γράφει ο D. E. Knuth, προέρχεται από τα τρία πρώτα γράμματα της κοινής ρίζας των λέξεων *τεχνολογία* και *τέχνη*. Έτσι το TeX προφέρεται *τεχ*.

Το LaTeX είναι ένα μακροπακέτο το οποίο δίνει την δυνατότητα στο χρήστη να στοιχειοθετήσει κείμενα υψηλής ποιότητας χρησιμοποιώντας προκαθορισμένα περιβάλλοντα. Το LaTeX γράφτηκε από τον L. Lamport, και χρησιμοποιεί το TeX σαν βασικό του εργαλείο στοιχειοθεσίας. Η σημερινή έκδοση τουLaTeX ονομάζεταιLaTeX 2_ε και αυτή θα περιγράψουμε εν συντομία σε αυτές τις πρόχειρες σημειώσεις.

4.1.2 Πως λειτουργεί το LaTeX

Η δημιουργία ενος κειμένου με την βοήθεια του LaTeX αποτελείται από δύο στάδια. Το πρώτο στάδιο περιλαμβάνει την σύνταξη του αντίστοιχου αρχείου LaTeX, ενω στο δεύτερο στάδιο 'τροφοδοτούμε'το LaTeX με το αρχείο αυτό. Το αρχείο LaTeX δεν είναι τίποτε άλλο παρά ένα αρχείο με κείμενο και εντολές του LaTeX που παράγουν το επιθυμητό αποτέλεσμα. Η σύνταξη του αρχείου αυτού μπορεί να γίνει π.χ. με το vi ή κάποιο άλλο προσφιλή μας κειμενογράφο. Κατόπιν σώζοντας το αρχείο αυτό το τροφοδοτούμε στο LaTeX με αποτέλεσμα να παραχθούν τα αρχεία με καταλήξεις : .aux, .log, .dvi. Στο αρχείο με κατάληξη .dvi μπορουμε να δούμε το στοιχειθετημένο πλεον κείμενο το οποίο στη συνέχεια μπορούμε και να εκτυπώσουμε σε χαρτί.

Ένα απλό παράδειγμα. Χρησιμοποιώντας το vi φτιάχνουμε το αρχείο prwto.tex το οποίο περιέχει τα εξής :

```
\documentclass{article}
\begin{document}
This is my first \LaTeX{} document.
\end{document}
```

Κατόπιν σώζουμε το αρχείο και στο σήμα αναμονής του UNIX δίνουμε:

```
% latex prwto.tex
```

Με την εκτέλεση αυτής της εντολής στη οθόνη μας εμφανίζονται διάφορες πληροφορίες σχετικά με το LaTeX και ότι το LaTeX δημιούργησε 3 αρχεία : prwto.aux, prwto.log, prwto.dvi. Τα δύο πρώτα παρέχουν διάφορες πληροφορίες ενώ το τρίτο prwto.dvi είναι το αρχείο είναι αυτό που περιέχει το τελικό προιόν της στοιχειοθεσίας και μπορούμε να το δούμε δίνοντας την εντολή

```
% xdvi prwto.dvi
```

Το τελικό αποτέλεσμα για το παραπάνω παράδειγμα θα είναι

This is my first LaTeX document.

Είναι επίσης δυνατό να εκτυπώσουμε το τελικό κείμενο στο χαρτί αφού πρώτα μετατρέψουμε το αρχείο prwto.dvi σε μορφή *postscript(prwto.ps)* δίνοντας τη εντολή

```
% dvips -o prwto.ps prwto.dvi
```

και κατόπιν εκτυπώνοντας το αρχείο prwto.ps με την εντολή lp.

4.2 Το αρχείο εισαγωγής του LaTeX

Όπως είδαμε το αρχείο που τροφοδοτούμε στο LaTeX είναι ένα απλό αρχείο που δημιουργούμε με το προσφιλή μας κειμενογράφο και το οποίο περιέχει τόσο το κείμενο όσο και τις εντολές του LaTeX.

4.2.1 Κενά

Τα κενά ένα η περισσότερα μεταχειρίζονται από το LaTeX σαν *ένα* κενό. Οποιοδήποτε κενό ή κενά στη αρχή της γραμμής αγνοούνται. Μία κενή γραμμή ανάμεσα σε δύο γραμμές κειμένου καθορίζει το τέλος μιας παραγράφου. Παραπάνω από μια κενές γραμμές είναι σαν μία κενή γραμμή.

4.2.2 Ειδικοί Χαρακτήρες

Τα παρακάτω σύμβολα είναι δεσμευμένοι χαρακτήρες οι οποίοι έχουν ειδική χρήση στο LaTeX. Αν εισαχθούν στο κείμενο υπο κανονικές συνθήκες δεν θα τυπωθούν και είναι πολύ πιθανό το LaTeX να κάνει ανεπιθύμητα πράγματα. Οι χαρακτήρες αυτοί είναι :

```
$ & % # _ { } ~ ^ \
```

Τους χαρακτήρες αυτούς μπορούμε να τους εισάγουμε στο κειμενό μας προσθέτοντας μπροστά από αυτά την *αντιπλαγία (backslash)*:

```
\$ \& \% \# \_ \{ \}
```

$ & % # _ { }

Τα άλλα σύμβολα όπως και πολλοί άλλοι χαρακτήρες μπορούν να τυπώθουν με ειδικές εντολές. Την αντιπλαγία \ δεν μπορούμε να την εισάγουμε προσθέτοντας μια επιπλέον αντιπλαγία \\ μιας και αυτή η ακολουθία χαρακτήρων χρησιμοποιείται για την διακοπή γραμμής. (Δοκιμάστε την εντολή `$\backslash$` , θα παράγει \).

4.2.3 Εντολές LaTeX

Οι εντολές LaTeX είναι ευαίσθητες στα μικρά και κεφαλαία γράμματα και μπορούν να έχουν μία από τις δύο ακόλουθες μορφές:

1. Ξεκινούν με την αντιπλαγία \ και όνομα που απαρτίζεται μόνο από γράμματα. Τα ονόματα εντολών τερματίζονται είτε με ένα κενό είτε με αριθμό ή κάποιο άλλο χαρακτήρα (μη-γράμμα).

2. Ξεκινούν με την αντιπλαγία \ και ακολουθεί ένας μόνο ειδικός χαρακτήρας.

Το LaTeX αγνοεί τα κενά μετά τις εντολές. Αν θέλουμε να προσθέσουμε κενό χώρο μετά από μία εντολή τότε είτε βάζουμε `{}` και ένα κενό ή μία ειδική εντολή που προσθέτει κενό χώρο, π.χ.

```
\LaTeX{} is a for typesetting documents. \\
The \LaTeX input file is a plain ASCII file.
```

LaTeX is a for typesetting documents.
The LaTeXinput file is a plain ASCII file.

Μερικές εντολές χρειάζονται παράμετρο ή οποία μπαίνει ανάμεσα σε δύο `{}`. Μερικές εντολές υποστηρίζουν προεραιτικές παραμέτρους οι οποίες προστίθενται μετά την εντολή ανάμεσα σε `[ ]`, π.χ.

4.2.4 Σχόλια

Το σύμβολο % μπορεί να χρησιμοποιηθεί για να εισάγουμε σχόλια μέσα στο αρχείο LaTeX σε οποιοδήποτε σημείο θέλουμε. Όταν το LaTeX κατά την διάρκεια επεξεργασίας του κειμένου συναντά το χαρακτήρα % αγνοεί τα υπόλοιπο της τρέχουσας γραμμής, τον τερματισμό γραμμής, και όλα τα κενά, π.χ.

```
This is an % not serious example
example for using the comment symbol \%
```

This is an example for using the comment symbol %

4.3 Δομή του κειμένου LaTeX

Η βασική δομή του κειμένου LaTeX είναι προκαθορισμένη. Κάθε πρέπει να ξεκινά με την εντολή

```
\documentclass{...}
```

Η εντολή αυτή προκαθορίζει τον τύπο του εγγράφου που θέλουμε να δημιουργήσουμε. Μετά ακολουθούν εντολές που προσδιορίζουν την μορφή του εγγράφου. Επίσης σε αυτό το σημείο μπορούμε να φορτώσουμε διάφορα πακέτα που προσφέρουν επιπλέον δυνατότητες στο LaTeX. Η σχετική εντολή είναι

```
\usepackage{...}
```

Κατόπιν ακολουθεί το κυρίως μέρος του εγγράφου με την εντολή,

```
\begin{document}
```

Τώρα είμαστε σε θέση να γράψουμε το κείμενο μας και τις εντολές του LaTeX που θέλουμε να χρησιμοποιήσουμε. Στο τέλος του κειμένου δίνουμε την εντολή

```
\end{document}
```

η οποία σηματοδοτεί το τέλος του κειμένου. Οτιδήποτε ακολουθεί αγνοείται. Το απλούστερο κείμενο LaTeX το είδαμε στη παράγραφο 4.2.

4.4 Η μορφή του κειμένου

4.4.1 Κατηγορίες κειμένων

Το πρώτο πράγμα που χρειάζεται το LaTeX να ξέρει για το αρχείο είναι η κατηγορία στην οποία θέλουμε να ανήκει. Αυτό καθορίζεται με την εντολή `\documentclass`. Το πλήρες συντακτικό της εντολής είναι:

```
\documentclass[options]{class}
```

Η παράμετρος *class* προσδιορίζει την κατηγορία του εγγράφου που θέλουμε να δημιουργήσουμε. Οι κατηγορίες των εγγράφων με τις οποίες θα ασχοληθούμε είναι οι ακόλουθες :

article Για άρθρα σε επιστημονικά περιοδικά, παρουσιάσεις, μικρές αναφορές κ.λ.π.

report Για μεγάλες αναφορές που μπορεί να περιέχουν κεφάλαια, εγχειρίδια, επιστημονικές διατριβές κ.λ.π.

book Για πραγματικά βιβλία.

slides Για διαφάνειες παρουσιάσεων.

Για τις επιλογές *options* μερικές από τις δυνατότητες είναι οι εξής :

10pt,11pt, 12pt Καθορίζει το μέγεθος της γραμματοσειράς. Το στάνταρτ είναι **10pt**

a4paper,letterpaper, ... Καθορίζει το μέγεθος του χαρτίου. Το στάνταρτ μέγεθος είναι το **letterpaper**. Η επιλογή **a4paper** είναι λοιπόν απαραίτητη για μας.

fleqn Τυπώνει τους μαθηματικούς τύπους και εξισώσεις με αριστερή στοίχηση αντί της στάνταρτ που είναι η κεντρική.

leqno Τοποθετεί την αρίθμηση των μαθηματικών τύπων και εξισώσεων στα αριστερά αντί στα δεξιά που είναι το στάνταρτ.

titlepage, notitlepage Καθορίζει αν μετά τον τίτλο του κειμένου ακολουθεί νέα σελίδα ή όχι. Στη κατηγορία article δεν ακολουθεί νέα σελίδα, το αντίθετο από ότι στις κατηγορίες report,book.

twocolumn Τυπώνει το κείμενο σε δύο στήλες ανά σελίδα.

π.χ.

```
\documentclass[a4paper,11pt]{aricle}
```

Η παραπάνω εντολή λέει στο LaTeX να στοιχειοθετήσει ένα κείμενο σαν άρθρο (article) με μέγεθος γραμματοσειράς 11pt[1] και το μέγεθος του χαρτιού να είναι A4.

4.4.2 Πακέτα

Δεν είναι λίγες οι φορές εκείνες που οι βασικές δυνατότητες του LaTeX δεν είναι επαρκής για να καλύψει κάποιες ιδιαίτερες απαιτήσεις, π.χ. θα θέλαμε να εισάγουμε γράφημα ή εικόνα στο κειμενό μας. Είναι λοιπόν αναγκαίο να βελτιώσουμε τις δυνατότητες του LaTeX. Οι βελτιώσεις αυτές γίνονται με την βοήθεια των διαφόρων *πακέτων*. Η χρήση των πακέτων γίνεται με την βοήθεια της εντολής

```
\usepackage[options]{package}
```

όπου πακέτο(package) είναι το όνομα του πακέτου και επιλογές(options) είναι ένα σύνολο επιλογών που ενεργοποιούν συγκεκριμμένες δυνατότητες του πακέτου. Πακέτα υπάρχουν πολλά, άλλωστε ο καθένας θα μπορούσε να φτιάξει το δικό του πακέτο. Στη βασική έκδοση του LaTeX περιλαμβάνονται τα εξής πακέτα :

fontenc Καθορίζει την κωδικοποίηση των γραμματοσειρών που θα χρησιμοποιήση το LaTeX.

latexsym Δυνατότητα πρόσβασης στο εκτεταμένο πίνακα συμβόλων του LaTeX.

makeidx Δυνατότητα δημιουργίας ευρετηρίου.

inputenc Καθορίζει την κωδικοποίηση του τρόπου εισαγωγής του κειμένου.

[1] 1in=72,27pt, 1in=2,54cm

4.4.3 Τύποι σελίδας

Το LaTeX υποστηρίζει τρείς προκαθορισμένους τύπους σελίδας. Η ενεργοποίηση κάποιου τύπου γίνεται με την εντολή

`\pagestyle{style}`

όπου *style* είναι ο τύπος της σελίδας. Οι τρείς προκαθορισμένοι τύποι είναι :

plain Τυπώνει τους αριθμούς των σελίδων στη μέση του κάτω μέρους της σελίδας.

headings Τυπώνει την επικεφαλίδα του τρέχοντος κεφαλαίου και τον αριθμό της σελίδας στο πάνω μέρος της σελίδας.

empty Τοσό το πάνω μέρος όσο και το κάτω μέρος της σελίδας παραμένουν κενά.

Είναι επίσης δυνατό να αλλάξουμε μόνο το τύπο της τρέχουσας σελίδας με την εντολή :

`\thispagestyle{style}`

Τέλος υπάρχει η δυνατότητα να δημιουργήσει κανείς τον δικό του τύπο σελίδας.