Κεφάλαιο IV:

Δείκτες και πίνακες.

4.1 Δείκτες.

Η C, όπως έχουμε αναφέρει, είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία αναπτύχθηκε για πρώτη φορά το 1972 από τον Dennis Ritchie στα AT&T Bell Labs. Στόχος ήταν η δημιουργία μιας γλώσσας μέσω της οποίας θα υπήρχε η δυνατότητα ανάπτυξης λειτουργικών συστημάτων (όπως το λειτουργικό σύστημα Unix). Μία τέτοια γλώσσα θα έπρεπε να δίνει την δυνατότητα στον προγραμματιστή με εύκολο τρόπο να επικοινωνεί κατ΄ευθείαν με την μνήμη του υπολογιστή και έμμεσα με το υλισμικό (hardware) του υπολογιστή. Για τον λόγο αυτό δημιουργήθηκαν οι **δείκτες**.

Οι δείκτες θα μπορούσαμε να πούμε πως αποτελούν την μεγάλη "δύναμη" της γλώσσας στον προγραμματισμό. Ως έννοιες για κάποιον αρχάριο είναι μάλλον δυσνόητες. Ένας καλός όμως προγραμματιστής στη C σήμερα οφείλει να ξέρει να χρησιμοποιεί τους δείκτες με ευχέρεια. Στα πλαίσια του εισαγωγικού μαθήματος μας, σε αυτή την παράγραφο θα κάνουμε μια απλή εισαγωγή στις έννοιες των δεικτών. Ο κάθε φοιτητής για περισσότερες λεπτομέρειες πρέπει να ανατρέξει στο διδακτικό σύγγραμμα.

Ας ξεκινήσουμε με τις ακόλουθες δύο γραμμές προγράμματος όπου ορίζουμε την μεταβλητή x που είναι τύπου float, την οποία και αρχικοποιούμε στην τιμή 3.1415.

```
float x;        /* Δήλωση μεταβλητής x τύπου float */
x=3.1415;       /* Αρχικοποίηση της μεταβλητής x */
```

Μέχρι τώρα την παραπάνω διαδικασία την έχουμε κάνει αυτόματα πολλές φορές. Τι όμως ακριβώς συμβαίνει στον υπολογιστή μας κατά την εκτέλεση των δύο αυτών γραμμών;

Τα πάντα εξελίσσονται στην μνήμη (RAM) του υπολογιστή μας. Μπορούμε να φανταστούμε την μνήμη σαν μια τεράστια ταινία γεμάτη κυψελίδες όπως φαίνεται στην παρακάτω εικόνα. Σε κάθε κυψελίδα (μεγέθους στο συγκεκριμένο παράδειγμα τεσσάρων bytes) μπορούν να αποθηκευτούν τα δεδομένα των μεταβλητών που χρησιμοποιούμε στα προγράμματά μας. Κάθε κυψελίδα όμως έχει και μία συγκεκριμένη διεύθυνση, έτσι ώστε ο υπολογιστής να γνωρίζει που να ανατρέξει κάθε φορά που το πρόγραμμα ζητά τα συγκεκριμένα δεδομένα. Η διεύθυνση κάθε κυψελίδας είναι συνήθως ένας μεγάλος αριθμός τον οποίο, στην γλώσσα των υπολογιστών, τον αναγράφουμε σε δεκαεξαδικό σύστημα. Στην παρακάτω εικόνα εικονίζονται μια σειρά κυψελίδων με τις αντίστοιχες διευθύνσεις τους.

x ⟶

Διευθύνσεις Μνήμης	Περιεχόμενο Μνήμης RAM
bfffdfdc	
bfffdfe0	3.1415
bfffdfe4	
bfffdfe8	
……..	

Όταν λοιπόν εκτελείται η γραμμή

```
float x;          /* Δήλωση μεταβλητής x τύπου float */
```

ο υπολογιστής ανατρέχει στην πρώτη κυψελίδα της μνήμης του η οποία είναι διαθέσιμη και δεσμεύει τον κατάλληλο χώρο για την μεταβλητή x. Στο συγκεκριμένο παράδειγμα δεσμεύει την κυψελίδα με διεύθυνση bfffdfe0. Όταν εκτελείται η δεύτερη γραμμή

```
x=3.1415;         /* Αρχικοποίηση της μεταβλητής x */
```

ο υπολογιστής πηγαίνει στην κυψελίδα με την διεύθυνση bfffdfe0 και αποθηκεύει τον αριθμό 3.1415. Με αυτόν τον τρόπο ο υπολογιστής σε κάθε μελλοντική αναφορά στην μεταβλητή x γνωρίζει που να ανατρέξει για να βρει τα δεδομένα.

Την διεύθυνση στη μνήμη που καταλαμβάνει κάθε μεταβλητή στο πρόγραμμά μας μπορούμε πολύ εύκολα να βρούμε κάνοντας χρήση του ***Τελεστή Διεύθυνσης &***. Για την διεύθυνση παραδείγματος χάριν της μεταβλητής x αρκεί να γράψουμε &x. Στις επόμενες δύο γραμμές χρησιμοποιώντας την συνάρτηση printf() εκτυπώνουμε τόσο το περιεχόμενο της μεταβλητής x όσο και την διεύθυνσή της στην μνήμη του υπολογιστή μας, για το παραπάνω παράδειγμα.

```
                                  Αποτέλεσμα εκτύπωσης
printf("%f \n",x);       =>       3.1415

printf("%p \n",&x);      =>       bfffdfe0
```

Παρατηρείστε πως για την εκτύπωση διευθύνσεων στην printf() χρησιμοποιούμε τον χαρακτήρα μετατροπής **%p**.

Ας δούμε τώρα την έννοια του ***δείκτη***. Θεωρείστε τις ακόλουθες γραμμές προγράμματος :

```
float x;              /* Δήλωση μεταβλητής x τύπου float */
float *point_x;       /* Δήλωση δείκτη *point_x τύπου float */
x=3.1415;             /* Αρχικοποίηση της μεταβλητής x */
point_x=&x;           /* Η μεταβλητή point_x λαμβάνει τιμή αυτήν της διεύθυνσης του x */
```

Στην πρώτη γραμμή δηλώνουμε μια μεταβλητή x τύπου float. Στη δεύτερη γραμμή ορίζουμε έναν δείκτη με όνομα point_x χρησιμοποιώντας τον ***Τελεστή Έμμεσης Αναφοράς ****. Μην συγχέετε το σύμβολο του πολλαπλασιασμού, το οποίο μπαίνει ανάμεσα από δύο μεταβλητές (πχ. x*y), με τον τελεστή έμμεσης αναφοράς, ο οποίος μπαίνει μπροστά από μία μεταβλητή όταν αυτή ορίζεται για να δηλώσει πως είναι δείκτης. Ο δείκτης *point_x οφείλει να είναι του ιδίου τύπου με την μεταβλητή x. Στην τρίτη γραμμή αρχικοποιούμε την μεταβλητή x στην τιμή 3.1415. Τέλος στην τέταρτη γραμμή αντιστοιχούμε στην μεταβλητή point_x την τιμή της διεύθυνσης του x.

Στο επόμενο σχήμα εικονίζεται το διάγραμμα στην μνήμη του υπολογιστή μας μετά την εκτέλεση των παραπάνω τεσσάρων γραμμών.

Στις επόμενες γραμμές χρησιμοποιώντας την συνάρτηση printf() εκτυπώνουμε το περιεχόμενο της μεταβλητής x, την διεύθυνσή του x στην μνήμη του υπολογιστή μας, την τιμή της μεταβλητής point_x, την διεύθυνσή της point_x στην μνήμη του υπολογιστή μας και τέλος την τιμή του δείκτη *point_x.

		Αποτέλεσμα εκτύπωσης
printf(“%f \n”,x);	=>	3.1415
printf(“%p \n”,&x);	=>	bfffdfe0
printf(“%p \n”,point_x);	=>	bfffdfe0
printf(“%p \n”,&point_x);	=>	bfffdfe4
printf(“%f \n”,*point_x);	=>	3.1415

Ενδιαφέρον μεγάλο παρουσιάζει η τελευταία γραμμή από την οποία συμπεραίνουμε πως ο δείκτης *point_x δείχνει το περιεχόμενο της κυψελίδας της μεταβλητής x. Με αυτόν τον τρόπο είτε αναφερόμαστε στην μεταβλητή x είτε στον δείκτη *point_x είναι το ίδιο πράγμα. Για παράδειγμα γράφοντας:

*point_x=6.2;

αλλάζουμε στην ουσία την τιμή της μεταβλητής x και την κάνουμε 6.2.

Τους δείκτες μπορούμε να τους χρησιμοποιούμε σε πολλές περιπτώσεις όπως να περνάμε με αναφορά (by reference) μεταβλητές σε συναρτήσεις, να επικοινωνούμε έμμεσα με το υλισμικό (hardware) του υπολογιστή κτλ. Αυτά τα πράγματα όμως ξεφεύγουν από ένα εισαγωγικό μάθημα προγραμματισμού και οι φοιτητές που ενδιαφέρονται πρέπει να ανατρέξουν στις αντίστοιχες παραγράφους του διδακτικού βιβλίου και στο internet.

4.2 Μονοδιάστατοι πίνακες.

Η ομαδοποίηση της πληροφορίας στον προγραμματισμό είναι πολύ σημαντική και επιβάλλεται έτσι ώστε να είναι εύκολη η επεξεργασία της. Για παράδειγμα εάν έχουμε δέκα διαφορετικά αριθμητικά δεδομένα του ιδίου τύπου, μπορούμε να ορίσουμε δέκα διαφορετικές μεταβλητές και να τα επεξεργαστούμε. Ο τρόπος αυτός όμως δεν είναι ο ενδεδειγμένος. Σε τέτοιες περιπτώσεις χρησιμοποιούμε πίνακες δεδομένων. Στην παράγραφο αυτή θα ασχοληθούμε με μονοδιάστατους πίνακες.

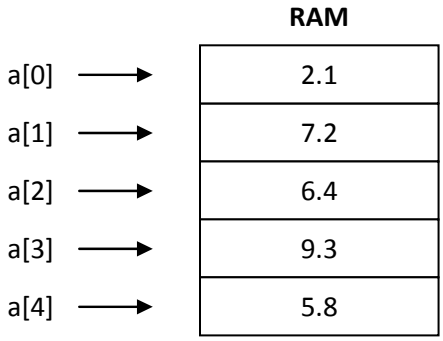
Για παράδειγμα ας υποθέσουμε ότι έχουμε πέντε αριθμούς τύπου float τους 2.1, 7.2, 6.4, 9.3, και 5.8. Ας ορίσουμε τώρα έναν πίνακα ο οποίος να αποτελείται από πέντε στοιχεία και ας τον αρχικοποιήσουμε στις παραπάνω τιμές. Αυτό γίνεται ως ακολούθως:

```
float a[5];      /* Δήλωση του πίνακα a με πέντε στοιχεία */
a[0]=2.1;        /* Αρχικοποίηση των πέντε στοιχείων */
a[1]=7.2;
a[2]=6.4;
a[3]=9.3;
a[4]=5.8;
```

Εδώ πρέπει να σημειώσουμε και να προσέξουμε τα ακόλουθα:

- Κάθε πίνακας έχει ένα όνομα όπως ακριβώς και μία απλή μεταβλητή. Ισχύουν οι ίδιοι κανόνες ονοματολογίας με τις απλές μεταβλητές.
- Όλα τα στοιχεία του πίνακα είναι του ιδίου τύπου.
- Το όνομα του πίνακα ακολουθεί ο αριθμός των στοιχείων του μέσα σε [].
- Η αρίθμηση των στοιχείων του πίνακα ξεκινά από το μηδέν και σταματά στο n-1, όπου n η διάσταση του πίνακα. Στο παραπάνω παράδειγμα το πρώτο στοιχείο του είναι το a[0] και το τελευταίο το a[4].
- Προσοχή! Δεν υπάρχει το στοιχείο a[5] και κάθε αναφορά σε αυτό προκαλεί σφάλμα στην εκτέλεση του προγράμματος με μήνυμα *segmentation fault*. Το ίδιο σφάλμα παίρνουμε εάν ξεφύγουμε από τα όρια του πίνακα, πχ να καλέσουμε το στοιχείο a[7] το οποίο δεν υπάρχει για τον παραπάνω πίνακα.

- Τα στοιχεία κάθε πίνακα κατατάσσονται στην μνήμη του υπολογιστή στην σειρά. Για παράδειγμα για τον παραπάνω πίνακα η κατάταξη των στοιχείων του στην μνήμη του υπολογιστή φαίνεται στο παρακάτω σχήμα:



Σημειώνουμε πως στο παραπάνω σχήμα κάθε κυψελίδα μνήμης αποτελείται από τέσσερα bytes.

Η αρχικοποίηση ενός πίνακα μπορεί να γίνει και με τους ακόλουθους δύο τρόπους:

```
float a[5]={ 2.1, 7.2, 6.4, 9.3, 5.8.};
float a[]={ 2.1, 7.2, 6.4, 9.3, 5.8.};
```

Προσέξτε πως στον τελευταίο τρόπο παραλείψαμε την διάσταση του πίνακα. Όταν αρχικοποιούμε έναν μονοδιάστατο πίνακα με τον παραπάνω τρόπο έχουμε το δικαίωμα να μην δηλώσουμε την διάσταση του. Ο υπολογιστής προσμετρά τα δεδομένα μέσα στις αγκύλες και την καθορίζει αυτόματα.

Στις παρακάτω γραμμές κώδικα παραθέτουμε ένα απλό παράδειγμα δημιουργίας ενός πίνακα τύπου double με δέκα στοιχεία. Ο πίνακας γεμίζει με τους αριθμούς 0., 1., 2., ….9. Με αυτό το παράδειγμα δείχνουμε πως μπορούμε με την εντολή for να διαχειριστούμε εύκολα έναν πίνακα.

```
int i;                      /* Δήλωση μεταβλητής i */
double x[10];               /* Δήλωση πίνακα x δέκα στοιχείων */

for(i=0; i<10; i++){
    x[i]=(double)i;         /* Αρχικοποίηση των στοιχείων με τους αριθμούς 0., 1., ….9. */
}
```

Η πλήρης εκτύπωση ενός πίνακα μπορεί να γίνει με την χρήση μιας εντολής for και της συνάρτησης printf(). Ακολουθεί ένα παράδειγμα εκτύπωσης ενός πίνακα:

```
int i;                          /* Δήλωση μεταβλητής i */
double x[7]={1.3, 5.6, 8.1, 9.5, 6.3, 6.5, 5.2 };       /* Δήλωση-αρχικοποίηση πίνακα x */

for(i=0; i<7; i++){
    printf("%f \n",x[i]);           /* Εκτύπωση των 7 στοιχείων του πίνακα */
}
```

Ας δούμε τώρα, με ένα παράδειγμα, τον τρόπο με τον οποίο μπορούμε να εισάγουμε από το πληκτρολόγιο ένα-ένα τα στοιχεία ενός πίνακα. Για παράδειγμα θα θεωρήσουμε ένα πίνακα double a[10]. Ακολουθούν οι γραμμές προγράμματος για την εισαγωγή των δεδομένων από το πληκτρολόγιο

```
int i;                /* Δήλωση μεταβλητής i */
double a[10];     /* Δήλωση πίνακα a δέκα στοιχείων */

for(i=0; i<10; i++){
    printf("Eisagete to stoixeio a[%d]: ",i);        /* Μήνυμα για τον χρήστη */
    scanf("%lf",&a[i]);                              /* Εισαγωγή των δεδομένων */
}
```

Όπως παρατηρούμε για να διαχειριστούμε έναν μονοδιάστατο πίνακα χρειαζόμαστε μία εντολή for. Κάθε φορά εισάγουμε τα δεδομένα σε ένα στοιχείο. Πριν την εισαγωγή των δεδομένων, με την συνάρτηση scanf(), στο κατάλληλο στοιχείο του πίνακα καλό είναι να τυπώνουμε ένα μήνυμα στον χρήστη του προγράμματος για το σε ποιο ακριβώς στοιχείο βρίσκεται κάθε φορά.

4.3 Πίνακες χαρακτήρων και συμβολοσειρές.

Ένας πίνακας των οποίων τα στοιχεία είναι χαρακτήρες ονομάζεται ***πίνακας χαρακτήρων***. Για παράδειγμα ο πίνακας

```
char a[5]={'H', 'e', 'l', 'l', 'o'};
```

είναι ένας πίνακας με πέντε στοιχεία τα οποία σχηματίζουν την αγγλική λέξη Hello.

Γενικά η εισαγωγή κειμένου και η επεξεργασία του στα προγράμματα είναι πολύ σημαντική γιαυτό τον λόγο στη C υπάρχουν οι ***συμβολοσειρές*** (strings). Ένα παράδειγμα συμβολοσειράς είναι το ακόλουθο:

```
char b[6]={'H', 'e', 'l', 'l', 'o', '\0'};
```

Η παραπάνω συμβολοσειρά b είναι ένας πίνακας χαρακτήρων με 6 στοιχεία και περιέχει την λέξη Hello. Παρατηρήστε πως έχει ένα στοιχείο περισσότερο από όσα χρειάζονται, το τελευταίο, το οποίο περιέχει τον κενό χαρακτήρα \0. Ο κενός χαρακτήρας είναι απαραίτητος και δηλώνει το τέλος της συμβολοσειράς. Η παραπάνω συμβολοσειρά μπορεί να αρχικοποιηθεί και με τους ακόλουθους δύο τρόπους:

```
char b[6]="Hello";
char b[]="Hello";
```

Στην τελευταία περίπτωση η συμβολοσειρά αποκτά αυτόματα την διάστασή της. Άλλα παραδείγματα συμβολοσειρών αποτελούν τα ακόλουθα:

```
char s[]="University of Ioannina";
char name[]="George Adams";
```

Για να εκτυπώσουμε συμβολοσειρές κάνουμε χρήση της συνάρτησης printf() και του χαρακτήρα μετατροπής s. Για παράδειγμα:

```
char a[]="University of Ioannina";
printf("%s \n", a);                /* Εκτυπώνεται η φράση University of Ioannina */
```

Για να εισάγουμε δεδομένα από το πληκτρολόγιο σε μία συμβολοσειρά χρησιμοποιούμε τη συνάρτηση scanf(), όπως στο επόμενο παράδειγμα:

```
char b[20];            /*Δήλωση συμβολοσειράς */
scanf("%s",&b);        /*Εισαγωγή δεδομένων στην συμβολοσειρά b */
```

Η C είναι εφοδιασμένη με έτοιμες συναρτήσεις οι οποίες σχετίζονται με τις συμβολοσειρές. Για να έχει την δυνατότητα ο προγραμματιστής να τις χρησιμοποιήσει πρέπει να συμπεριλάβει στο πρόγραμμά του το αρχείο επικεφαλίδας string.h (δηλαδή #include<string.h>). Έστω οι a και b κατάλληλες συμβολοσειρές. Μερικές βασικές συναρτήσεις που σχετίζονται με συμβολοσειρές είναι οι ακόλουθες:

- strlen(a) : Επιστρέφει το μήκος της συμβολοσειράς a.
- strcmp(a,b) : Συγκρίνει τα περιεχόμενα της συμβολοσειράς a με αυτά της b. Εάν αυτά είναι ίδια επιστρέφει μηδέν. Επιστρέφει τιμή >0 ή <0 ανάλογα με την αλφαβητική τους κατάταξη. Η συνάρτηση αυτή είναι πολύ χρήσιμη όταν ψάχνουμε μια λέξη μέσα σε ένα κείμενο ή μέσα σε μια βάση δεδομένων.
- strcpy(a,"hello") : Κοπιάρει την λέξη πχ hello μέσα στην συμβολοσειρά a.
- strcat(a," there") : Προσθέτει στο τέλος της συμβολοσειράς την λέξη πχ there. Οι δύο τελευταίες συναρτήσεις έχουν ως αποτέλεσμα η συμβολοσειρά a να γίνει η φράση hello there.

Για περισσότερες συναρτήσεις σχετικές με συμβολοσειρές ο αναγνώστης πρέπει να ανατρέξει στο διδακτικό βιβλίο ή το internet.

4.4 Χρήση πίνακα ως όρισμα σε συνάρτηση.

Ας δούμε με ένα απλό παράδειγμα πως μπορούμε να περάσουμε έναν ολόκληρο πίνακα ως όρισμα σε μία συνάρτηση. Υποθέτουμε πως έχουμε ένα πίνακα double x[5] και θέλουμε να γράψουμε μια συνάρτηση η οποία να υπολογίζει το άθροισμα των στοιχείων της. Το όλο πρόγραμμα έχει ως ακολούθως:

```
#include<stdio.h>

double sum(double a[]);          /* Δήλωση πρωτότυπο της συνάρτησης */

int main(void){
    double x[5]={5.2, 7.1, 6.3, 9.2, 4.7};   /* Δήλωση-αρχικοποίηση του πίνακα x[5] */
    double y;                                /* Δήλωση μιας μεταβλητής y */
    y=sum(x);                                /* Κλήση της συνάρτησης */
    printf("Athroisma stoixeion = %f \n",y);         /* Εκτύπωση Αποτελέσματος */
    return 0;                                /* Τερματισμός του προγράμματος */
}
```

```
double sum(double a[]){                    /* Αρχή της συνάρτησης sum() */
    double s;              /* Δήλωση μιας μεταβλητής s για τον υπολογισμό του αθροίσματος */
    s=0.;                  /* Αρχικοποίηση της s στο μηδέν */
    for(i=0; i<5; i++){
        s=s+a[i];          /* Υπολογισμός αθροίσματος των στοιχείων του πίνακα */
    }
    return s;              /* Επιστροφή του αποτελέσματος */
}
```

Τα βασικά σημεία του παραπάνω προγράμματος είναι τα ακόλουθα:

- Κατά την κλήση της συνάρτησης (y=sum(**x**)) ο πίνακας x περνά ως μια απλή μεταβλητή χωρίς την διάστασή του (δηλαδή χωρίς τα σύμβολα []).
- Στην συνάρτηση ως όρισμα βάζουμε ένα πίνακα χωρίς διαστάσεις δηλαδή
  double sum(double **a[]**).
  ο πίνακας αποκτά αυτόματα διάσταση ίση με αυτή του πίνακα της συνάρτησης main().

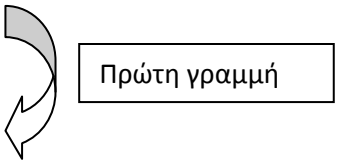
4.5 Πίνακες πολλών διαστάσεων.

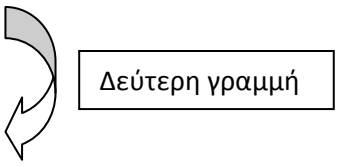
Η έννοια του πίνακα μπορεί να επεκταθεί και σε περισσότερες από μία διαστάσεις. Με αυτόν τον τρόπο μπορούμε να ορίσουμε πίνακες δύο, τριών, τεσσάρων κτλ. διαστάσεων. Ας πάρουμε ένα παράδειγμα ενός πίνακα δύο διαστάσεων. Ο παρακάτω πίνακας Α είναι ένας πίνακας δύο διαστάσεων με τρεις γραμμές, τέσσερις στήλες και συνολικά δώδεκα στοιχεία

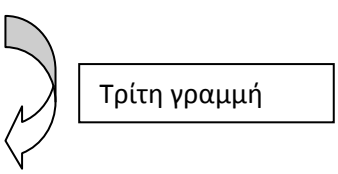
$$A(3\times 4)=\begin{bmatrix}1.2 & 4.3 & 7.8 & 9.3\\ 6.3 & 8.9 & 6.7 & 3.4\\ 1.9 & 9.8 & 4.1 & 6.6\end{bmatrix}$$

Ακολουθεί ο ορισμός του πίνακα στη C και η αρχικοποίηση των στοιχείων του:

```
double a[3][4];

a[0][0]=1.2;
a[0][1]=4.3;
a[0][2]=7.8;
a[0][3]=9.3;

a[1][0]=6.3;
a[1][1]=8.9;
a[1][2]=6.7;
a[1][3]=3.4;

a[2][0]=1.9;
a[2][1]=9.8;
a[2][2]=4.1;
a[2][3]=6.6;
```







Εκτός από την παραπάνω δήλωση και αναλυτική αρχικοποίηση του πίνακα μπορούμε να χρησιμοποιήσουμε και τους παρακάτω δύο τρόπους:

```
double a[3][4]={1.2, 4.3, 7.8, 9.3, 6.3, 8.9, 6.7, 3.4, 1.9, 9.8, 4.1, 6.6};

double a[][4]={1.2, 4.3, 7.8, 9.3, 6.3, 8.9, 6.7, 3.4, 1.9, 9.8, 4.1, 6.6};
```

Προσέξτε πως στον τελευταίο τρόπο παραλείψαμε την πρώτη διάσταση. Όπως και στους μονοδιάστατους πίνακες έχουμε το δικαίωμα να μην δηλώσουμε την <u>πρώτη μόνο</u> διάσταση ενός πολυδιάστατου πίνακα. Ο υπολογιστής προσμετρά τα δεδομένα μέσα στις αγκύλες και την καθορίζει αυτόματα.

Με την ίδια λογική όπως παραπάνω μπορούμε να ορίσουμε πίνακες τριών ή περισσοτέρων διαστάσεων. Για παράδειγμα ο πίνακας b[3][4][5] έχει συνολικά 60 στοιχεία τα οποία και πρέπει προσεκτικά να τα αρχικοποιήσουμε ακολουθώντας τους παραπάνω κανόνες.

Ας δούμε τώρα, με ένα παράδειγμα, τον τρόπο με τον οποίο μπορούμε να εισάγουμε ένα-ένα τα στοιχεία ενός πίνακα. Για παράδειγμα θα θεωρήσουμε ένα πίνακα double a[3][4]. Ακολουθούν οι γραμμές προγράμματος για την εισαγωγή των δεδομένων από το πληκτρολόγιο.

```
double a[3][4];          /* Δήλωση του πίνακα */
int i,j;                 /* Δήλωση των ακεραίων i και j για τη διαχείριση του πίνακα */

for(i=0; i<3; i++){
    for(j=0; j<4; j++){
        printf("Dose to stoixeio a[%d][%d]:",i,j);      /* Μήνυμα για τον χρήστη */
        scanf("%lf",&a[i][j]);                          /* Εισαγωγή των δεδομένων */
    }
}
```

Όπως παρατηρούμε για να διαχειριστούμε έναν πίνακα δύο διαστάσεων χρειαζόμαστε δύο εντολές for την μία μέσα στην άλλη. Η πρώτη "τρέχει" πάνω στις γραμμές και η δεύτερη στις στήλες του πίνακα. Κάθε φορά εισάγουμε τα δεδομένα σε ένα στοιχείο. Πριν την εισαγωγή των δεδομένων, με την συνάρτηση scanf(), στο κατάλληλο στοιχείο του πίνακα καλό είναι να τυπώνουμε ένα μήνυμα στον χρήστη του προγράμματος για το σε ποιο ακριβώς στοιχείο βρίσκεται κάθε φορά.

Ας δούμε τώρα πως μπορούμε να εκτυπώσουμε έναν πίνακα δύο διαστάσεων σε μορφή γραμμών-στηλών πλήρως στοιχισμένων. Παίρνουμε για παράδειγμα τον παραπάνω πίνακα double a[3][4]. Ο κώδικας για την εκτύπωσή του θα έχει ως ακολούθως:

```
int i,j;                 /* Δήλωση των ακεραίων i και j για τη διαχείριση του πίνακα */

for(i=0; i<3; i++){
    for(j=0; j<4; j++){
        printf("%10.2f",a[i][j]);          /*Εκτύπωση του στοιχείου a[i][j]  */
    }
```

```
        printf("\n");           /* Αλλαγή γραμμής */
    }
```

Όπως παρατηρούμε για να εκτυπώσουμε έναν πίνακα δύο διαστάσεων χρειαζόμαστε δύο εντολές for την μία μέσα στην άλλη. Η πρώτη "τρέχει" πάνω στις γραμμές και η δεύτερη στις στήλες του πίνακα. Κάθε φορά εκτυπώνουμε ένα στοιχείο. Για την εκτύπωση χρησιμοποιούμε το κατάλληλο εύρος πεδίου και αριθμό δεκαδικών ψηφίων. Με αυτόν τον τρόπο επιτυγχάνουμε την πλήρη στοίχιση των στοιχείων κατά την εκτύπωση του. Τέλος χρησιμοποιούμε μία εκτύπωση με τον χαρακτήρα της νέας γραμμής "\n" (μεταξύ των δύο for) έτσι ώστε να αλλάζουμε γραμμή.