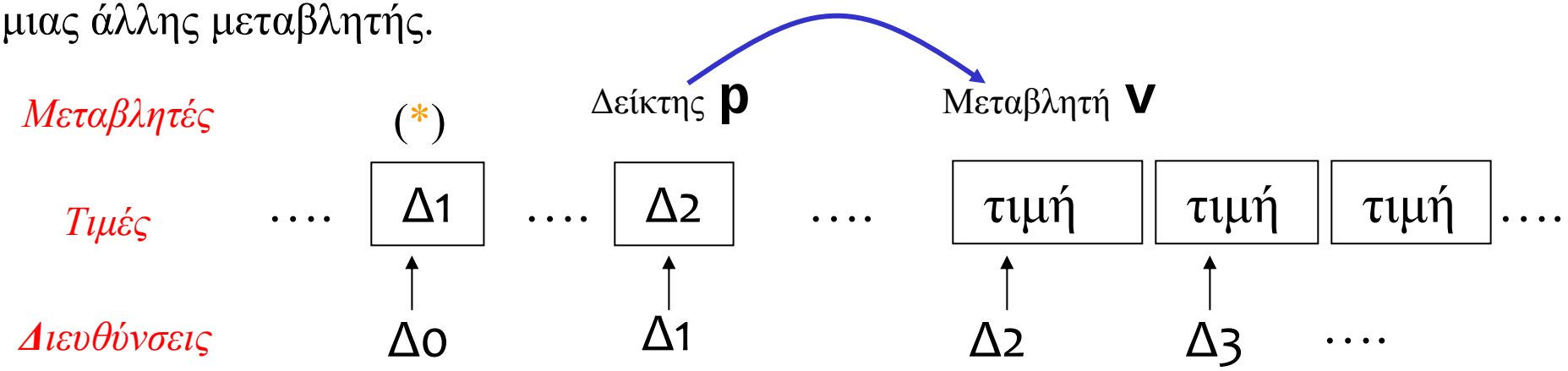
Δείκτες & Πίνακες

Δείκτες, Πίνακες

Δείκτες

Δείκτης είναι μια μεταβλητή που ως δεδομένο περιέχει τη θέση μνήμης (διεύθυνση) μιας άλλης μεταβλητής.



Ο δείκτης **p** δείχνει την μεταβλητή **v**

```
p=&v
```

- **Τελεστής &** : εφαρμόζεται μπροστά από μεταβλητές και δείχνει την διεύθυνσή τους
- **Τελεστής *** : εφαρμόζεται μπροστά από έναν δείκτη και δείχνει την τιμή που είναι αποθηκευμένη στη διεύθυνση που δείχνει ο δείκτης

```
v=*p
```

- Κάθε δείκτης δηλώνεται με έναν τύπο (char, int, float κλπ) που οφείλει να εκφράζει τον τύπο της μεταβλητής στην οποία δείχνει.

(*) Θέση μνήμης στην οποία είναι αποθηκευμένη η διεύθυνση του δείκτη p. Η διαχείριση της γίνεται από το σύστημα

Δηλώσεις και χειρισμός Δεικτών

```c
/* prog41pointers1.c */

#include <stdio.h>

main()
{
        int* p;
        double* q;
        int m=100;
        printf("the size of pointer p is %d bytes\n",sizeof(p));
        printf("the size of pointer q is %d bytes\n",sizeof(q));
        printf("The memory address of variable m is %p\n",&m);
        p=&m;
        printf("p points at the memory address %p\n",p);
        printf("The stored value where p points = %d\n",*p);
        printf("Pointer p is stored by the system at %p\n",&p);
        getchar();
}
```

Δήλωση δεικτών (τύπος*)

Όλοι οι δείκτες καταλαμβάνουν στη μνήμη το ίδιο μέγεθος (όσο ένας integer)

Ο δείκτης p δείχνει στη μεταβλητή m

Η τιμή *p είναι η τιμή του m

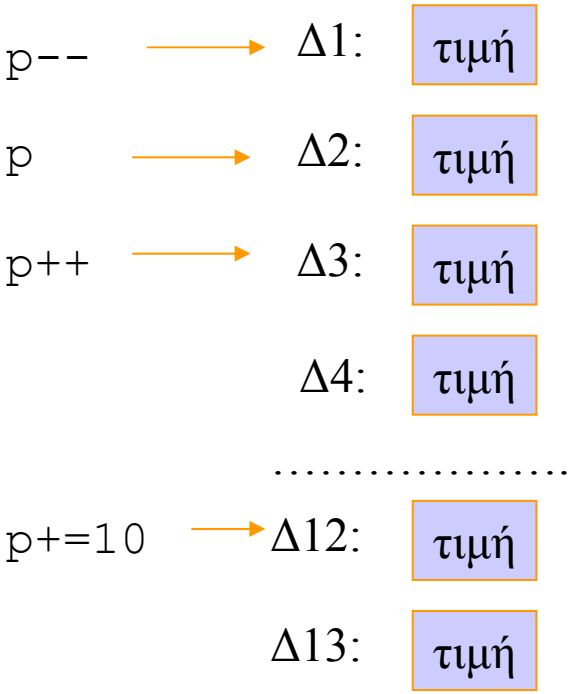
```
C:\RunCourses\Cprog\Mathima4_parrays\prog41pointers1.exe
the size of pointer p is 4 bytes
the size of pointer q is 4 bytes
The memory address of variable m is 0022FF6C
p points at the memory address 0022FF6C
The stored value where p points = 100
Pointer p is stored by the system at 0022FF74
```

Δες επίσης ***prog41pointers2.c***

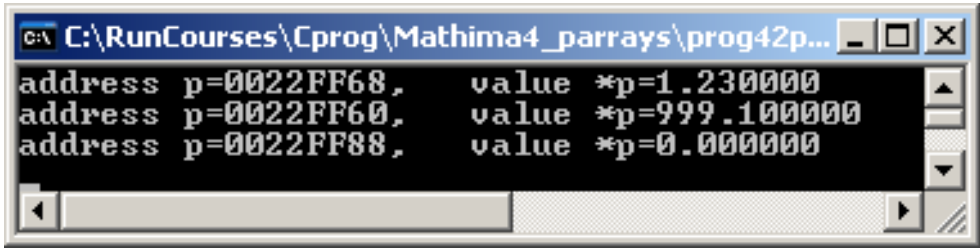
Αριθμητική Δεικτών

(τελεστές +, - / τελεστές σύγκρισης)

Η αύξηση ή μείωση της τιμής του δείκτη `p` κατά `n`, αλλάζει την τιμή του δείκτη (διεύθυνση μνήμης) κατά `n` θέσεις (κάτω ή πάνω, αντίστοιχα) ή κατά `n*s` bytes, όπου `s` το μέγεθος του τύπου του δείκτη.




```
double* p;
double x=1.23, y=999.1;
p=&x;
printf("address p=%p,    value *p=%f\n",p,*p);
p=p-1; /* or p--; */
printf("address p=%p,    value *p=%f\n",p,*p);
p+=5; /*or p=p+5 */
printf("address p=%p,    value *p=%f\n",p,*p);
```



Οι τελεστές σύγκρισης λειτουργούν στους δείκτες με τον ίδιο τρόπο όπως και στους αριθμούς

Οι τελεστές δεικτών *, & έχουν πιο ισχυρή σύνδεση από τους αριθμητικούς. Όμως ισχύει η προσεταιριστικότητα από δεξιά προς τα αριστερά

Δες επίσης ***prog42pnumerics.c***

Πίνακες

Ο πίνακας είναι ένα σύνολο δεδομένων (του ίδιου τύπου) στα οποία αναφερόμαστε με ένα σύμβολο μιας μεταβλητής και σε έναν ακέραιο (δείκτη) με αρχή το μηδέν.

Δήλωση π.χ.

`double X[Nmax]` : πίνακας με Nmax δεδομένα τύπου double (Nmax : σταθερά)

`int a[10]` : ένας πίνακας με 10 δεδομένα τα τύπου int που αντιστοιχούν στις μεταβλητές `a[0], a[1], a[2], …., a[9].`

Εισαγωγή στοιχείων

```
int i;
int a[]={1,2,3,4,5,6,7,8};
double b[5], c[100];

b[0]=2.1; b[1]=3.5; b[2]=0.7; b[3]=5.1; b[4]=1.9;

scanf("%d %d  %d",&b[0],&b[1],&b[2]);
scanf("%d %d",&b[3],&b[4]);

for(i=0;i<100;i++) c[i]=sqrt(i);
```

Τιμές κατά τη δήλωση και καθορισμός του μεγέθους του πίνακα

Τιμές σε κάθε στοιχείο ξεχωριστά

Εισαγωγή τιμών από μονάδα εισόδου

Τιμές μέσω ενός βρόχου επανάληψης.

Δες κώδικα ***prog43array.c***

Ασκήσεις – Πίνακες με τυχαίους αριθμούς

Τυχαίοι αριθμοί (βιβλιοθήκη **stdlib**)

`rand( )` : συνάρτηση που δίνει ένα τυχαίο ακέραιο αριθμό από 0 έως RAND_MAX-1

(η τιμή RAND_MAX εξαρτάται από το σύστημα, για την ANCI C είναι RAND_MAX=32768)

`srand(` *seed* `)` : Αρχικοποιεί τη γεννήτρια τυχαίων αριθμών με βάση τον ακέραιο *seed*.

* Ως *seed* μπορεί να δοθεί ο ακέραιος που επιστρέφει η συνάρτηση `time(NULL)` ώστε κάθε φορά που εκτελείτε το πρόγραμμα να τίθεται διαφορετικό *seed*.

Άσκηση 4.20. Γεμίστε ένα πίνακα μεγέθους Nmax με τυχαίους αριθμούς

α) με ακέραιους από 0 ως 99

β) με ακέραιους μεταξύ δύο τιμών min και max

γ) με floats στο διάστημα (0,1)

Κώδικας ***prog44random.c***

Άσκηση 4.21.

Δηλώστε ένα πίνακα ακεραίων με 20 στοιχεία. Γεμίστε τον πίνακα με τυχαίους αριθμούς από το 0 ως το 1000. Βρείτε τον μέγιστο, τον ελάχιστο και υπολογίστε το μέσο όρο των αριθμών.

Πίνακες και Δείκτες

Η μεταβλητή ενός πίνακα είναι μεταβλητή δείκτη που δείχνει στο πρώτο στοιχείο του πίνακα. Με τη δήλωση πίνακα κατοχυρώνεται ελεύθερος χώρος στη μνήμη για τα δεδομένα του πίνακα.

```
int a[10];
int* p;
p=a;
p=&a[0];
```

ισοδύναμα

```
printf("a=%p   p=%p    address a[0]=%p", a,p,&a[0]);
```

Τα στοιχεία του πίνακα γράφονται σε διαδοχικές θέσεις μνήμης και μπορούμε να τα προσπελάσουμε με δείκτες

```
p=&a[0]; q=&a[1]; r=&a[2] ….
(p+1)==q==(r-1)   (TRUE)
```

```
int a[5]={50,60,70,80,90};
int *p, i;
p=&a[0];
for(i=0;i<5;i++) {
    printf("%d ",*p);
    p++;
}
```

Ασκήσεις

(* στα παραδείγματα και ασκήσεις θεωρούμε ότι οι πίνακες έχουν κάποιες τιμές που έχουν δοθεί με τυχαίο τρόπο όπως στην άσκηση 4.20)

Άσκηση 4.31. Αντιγράψτε τα στοιχεία ενός πίνακα `a`, μεγέθους `N`, σε έναν πίνακα `b`. (a=b?)

Άσκηση 4.32. Θεωρήστε δύο πίνακες, τον πίνακα `a`, μεγέθους `N`, και τον πίνακα `b` μεγέθους `M`. Αντιγράψτε τα στοιχεία του `a` και τα στοιχεία του `b` σε έναν πίνακα `c` (με μέγεθος `N+M`), τοποθετώντας πρώτα τα στοιχεία του `a` και μετά τα στοιχεία του `b`

Άσκηση 4.33. Θεωρήστε ένα πίνακα `a` με τυχαίες τιμές από το 1 έως το 100. Κάντε ένα πρόγραμμα όπου ο χρήστης να δίνει έναν αριθμό και το πρόγραμμα να ελέγχει αν ο αριθμός αυτός υπάρχει στα στοιχεία του πίνακα και σε ποια θέση.

Swapping : ανταλλαγή τιμών μεταβλητών `a` ⟷ `b`

`a[j]` ⟷ `a[i]`

```
double a=150.1;
double b=400.9;
double tmp;

tmp=a;   a=b; b=tmp;    /* swap */
printf("a=%f  b=%f\n",a,b);
```

Άσκηση 4.35.
Αντιστρέψτε τη θέση των στοιχείων ενός πίνακα *α* μεγέθους *N* (`0..N-1`). Το πρώτο στοιχείο να τοποθετηθεί στο τέλος, το τελευταίο στο πρώτο και, γενικά το στοιχείο *i* να τοποθετηθεί στη θέση *N-i-1* (`A[i]` ⟷ `A[N-i-1]`).

Ταξινόμηση πίνακα

Sorting : ταξινόμηση στοιχείων πίνακα κατά αύξουσα (ή φθίνουσα σειρά)

Μέθοδος 1 (selection). Βρίσκουμε τον μικρότερο μεταξύ όλων των στοιχείων, έστω στη θέση `k`. Τοποθετούμε το στοιχείο a_k στη θέση a_0 και το a_0 στη θέση a_k (swap). Επαναλαμβάνουμε τον ίδιο έλεγχο από το δεύτερο στοιχείο (a_1) και βρίσκουμε το ελάχιστο στην νέα θέση k. Τοποθετούμε το στοιχείο a_k στη θέση a_1 και το a_1 στη θέση a_k (swap). κ.ο.κ.

```
>>Δίνεται πίνακας a με N στοιχεία

(επανάληψη 1) Για i=0 έως N-2 με βήμα 1
                θέσε min=a[i]
                (επανάληψη 2) Για j=i+1 έως N-1 με βήμα 1
                                Αν a[j]<min τότε
                                                θέσε min=a[j]
                                                 swap a[j] ⟷ a[i]
                 τέλος επανάληψης 2
τέλος επανάληψης 1
Τέλος ταξινόμησης
```

Κώδικας ***selectionsort.c***

Ταξινόμηση πίνακα

Μέθοδος 2 (bubble sort).
Συγκρίνουμε τα δύο πρώτα στοιχεία. Αν $a_0 > a_1$ τοποθετούμε το στοιχείο a_1 στη θέση a_0 και το a_0 στη θέση a_1 (swap). Στη συνέχεια, συγκρίνουμε το a_1 με το a_2 και εκτελούμε την ίδια ενέργεια. Συνεχίζουμε μέχρι το τελευταίο στοιχείο. Προφανώς ο μέγιστος αριθμός έχει μεταφερθεί στο τέλος. Ξαναεκτελούμε τον ίδιο αλγόριθμο από την αρχή. Σταματάμε αν κατά την τελευταία εκτέλεση δεν σημειώθηκε κανένα swap.

```
>>Δίνεται πίνακας a με N στοιχεία

(επανάληψη 1) Για i=1 έως N-1 με βήμα 1
                θέσε Flag=FALSE
                (επανάληψη 2) Για j=0 έως N-1-i με βήμα 1
                              Αν a[j]>a[j+1] τότε
                                                swap a[j] ←→ a[i]
                                                θέσε Flag=TRUE
                 τέλος επανάληψης 2
                 Αν Flag=FALSE τέλος ταξινόμησης (break)
τέλος επανάληψης 1
Τέλος ταξινόμησης
```

Κώδικας ***bubblesort.c***

Ασκήσεις

Άσκηση 4.40. Στα προγράμματα ***selectionsort.c*** και ***bubblesort.c*** προσθέστε ένα μετρητή count που να μετράει πόσες συγκρίσεις (if) έγιναν συνολικά μέχρι να ταξινομηθεί ο πίνακας. Ποιος αλγόριθμος είναι γρηγορότερος;

Άσκηση 4.41. Θεωρήστε ένα πίνακα `a` με 20 στοιχεία και τυχαίες τιμές από το 1 έως το 100. Κάντε ένα πρόγραμμα που να βρίσκει τους αριθμούς που παρουσιάζονται περισσότερες από μια φορές στον πίνακα.

Άσκηση 4.44. Θεωρήστε ένα πίνακα `a` με 52 στοιχεία τα οποία αντιστοιχούν στα χαρτιά μιας τράπουλας*. Κάντε ένα πρόγραμμα που να ανακατεύει την τράπουλα
(* θεωρήστε ότι κάθε τραπουλόχαρτο αντιστοιχεί σε έναν αριθμό από το 1 έως το 52)

Άσκηση 4.44. Γεμίστε ένα πίνακα `a`, 20 στοιχείων, με αριθμούς από το 0 έως το 9 και έτσι ώστε ο κάθε αριθμός να παρουσιάζεται στο πίνακα δύο φορές αλλά με ανάκατη (τυχαία) σειρά.

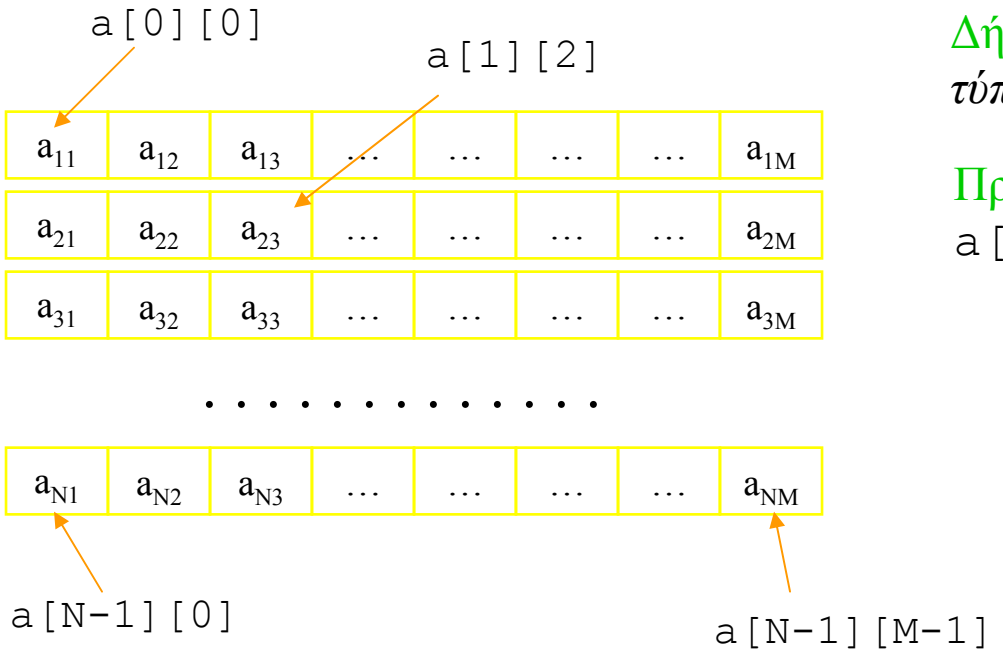
Πίνακες Δύο διαστάσεων

Μία Διάσταση. Μπορούμε να αντιστοιχίσουμε ένα διάνυσμα διάστασης N σε έναν πίνακα `a`, μεγέθους `N`

$$\vec{a} = (a_1, a_2, ..., a_N) \quad \leftrightarrow \quad a[N], \quad a[0] = a_1,\ a[1] = a_2,\ ...,\ a[N-1] = a_N$$

a_1	a_2	a_3	…	…	…	…	a_N

Δύο Διαστάσεις. Μπορούμε να αντιστοιχίσουμε έναν «αλγεβρικό» πίνακα **A**, με N γραμμές και M στήλες, σε ένα πίνακα (του υπολογιστή) δύο διαστάσεων με μέγεθος NxM



Δήλωση πίνακα

τύπος `a[N][M]`, (`N`,`M` σταθερές)

Προσπέλαση στοιχείου

`a[i][j]`, (`i`,`j` ακέραιοι δείκτες)

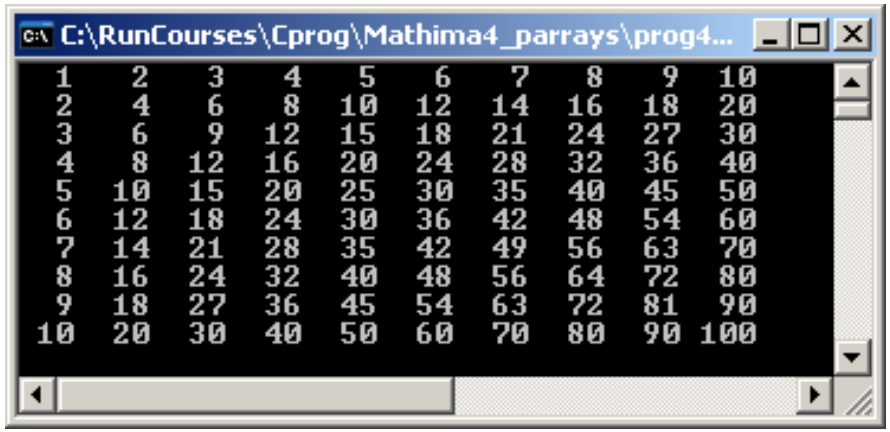
Πίνακες Δύο διαστάσεων – Παραδείγματα

Η προσπέλαση όλων των στοιχείων ενός δισδιάστατου πίνακα NxM γίνεται με ένθετη επανάληψη (συνήθως for). Πχ, η εσωτερική επανάληψη προσπελαύνει κάθε στοιχείο μιας δεδομένης γραμμής k (από το 1ο μέχρι το M-οστό στοιχείο), ενώ η εξωτερική επανάληψη προσπελαύνει την κάθε γραμμή (από το 1ο έως το N-οστο στοιχείο).

Παράδειγμα 4.80: σε ένα πίνακα 10x10 δίνουμε τις τιμές $a_{ij} = i*j$ (i,j=1,2,..10), και εκτυπώνουμε τα στοιχεία του πίνακα αλλάζοντας γραμμή εκτύπωσης κατά την αλλαγή γραμμής του πίνακα.

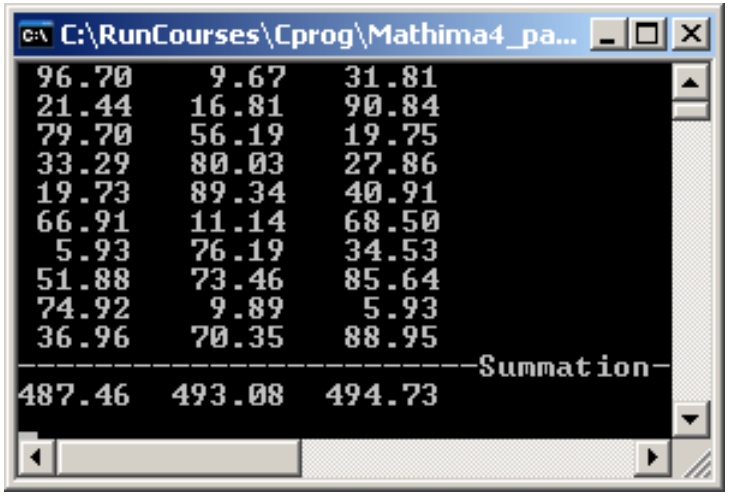
```
#include <stdio.h>
#define N 10
#define M 10

main()
{
        int i,j;
        int a[N][M];
        /* fill array with values */
        for(i=0;i<N;i++)
                  for(j=0;j<M;j++) a[i][j]=(i+1)*(j+1);
        /* print array element values */
         for(i=0;i<N;i++) {
                             for(j=0;j<M;j++) printf("%3d ",a[i][j]);
                             printf("\n");
                             }
         getchar();
}
```



Πίνακες Δύο διαστάσεων – Παραδείγματα

Παράδειγμα 4.81: Γεμίστε ένα πίνακα a, 10 γραμμών και 3 στηλών, με τυχαίους πραγματικούς αριθμούς από το 0 έως το 100. Εκτυπώστε τον πίνακα και υπολογίστε το άθροισμα της κάθε στήλης του πίνακα.




```c
#include <stdio.h>
#include <stdlib.h>
#define N 10
#define M 3

main()
{
        int i,j;
        double a[N][M];
        double sum1, sum2, sum3;
        /* fill array with values */
        srand(100001);
        for(i=0;i<N;i++)
                for(j=0;j<M;j++) a[i][j]= 100.0*rand()/RAND_MAX;
        /* print array element values */
         for(i=0;i<N;i++) {
                        for(j=0;j<M;j++) printf("%6.2f  ",a[i][j]);
                        printf("\n");
                        }
         sum1=0; for(i=0;i<N;i++) sum1+=a[i][0]; /* summation of 1st column*/
         sum2=0; for(i=0;i<N;i++) sum2+=a[i][1]; /* summation of 2nd column*/
         sum3=0; for(i=0;i<N;i++) sum3+=a[i][2]; /* summation of 3rd column*/
         printf("----------Summation--\n");
         printf("%6.2f  %6.2f  %6.2f\n",sum1,sum2,sum3);
         getchar();
}
```

Πίνακες Δύο διαστάσεων – Παραδείγματα

Παράδειγμα 4.85: Δημιουργήστε έναν πίνακα για να αποθηκεύσετε μια σειρά από *N* ονόματα (πχ George, Nike, Maria, Eleni, Dimitris)


```c
#include <stdio.h>
/* example :  names with 10 characters maximum */

main()
{
      int i;
      char onoma[][10]={"George","Nike","Maria","Eleni","Dimitris"};
      for(i=0;i<5;i++) printf("%d  %s\n",i, onoma[i]);
      getchar();
}
```

- Ένας δισδιάστατος πίνακας στη C (με μέγεθος NxM) είναι ένας δείκτης που δείχνει στο 1ο δείκτη ενός πίνακα από N δείκτες, που ο καθένας δείχνει σε ένα πίνακα με M στοιχεία (δηλαδή ο κάθε δείκτης δείχνει σε ένα δείκτη μετά τον οποίο υπάρχουν M ελεύθερες θέσεις για αποθήκευση δεδομένων!)

- Η διαχείριση πινάκων με ονόματα είναι λίγο περίπλοκη γιατί απαιτεί ουσιαστικά δισδιάστατους πίνακες χαρακτήρων. Γι' αυτό είναι εύχρηστο να κάνουμε χρήση των συναρτήσεων της βιβλιοθήκης "string" (βλ επόμενο μάθημα).

Ασκήσεις

Άσκηση 4.90: Γεμίστε δύο πίνακες `a` και `b`, 10 γραμμών και 3 στηλών, με τυχαίους πραγματικούς αριθμούς από το 0 έως το 10. Αθροίστε τους δύο πίνακες, δηλαδή δημιουργείστε έναν τρίτο πίνακα `c` του ίδιου μεγέθους με στοιχεία $c_{ij}=a_{ij}+b_{ij}$.

Άσκηση 4.91: Γεμίστε έναν πίνακα α (με μέγεθος 10x3) με ακεραίους στο διάστημα [0,9]. Βρείτε πόσες φορές εμφανίζεται ο αριθμός 0 ως στοιχείο του πίνακα.

Άσκηση 4.92: Γεμίστε έναν πίνακα 10x20 όπως στην άσκηση 4.91. Βρείτε ποια στήλη έχει τα περισσότερα μηδενικά στοιχεία.

Ορίζω ακέραιο πίνακα **a** (10x20)
Ορίζω ακέραιο πίνακα **c** με 10 στοιχεία (για αποθήκευση του αριθμού των μηδενικών σε κάθε γραμμή)
Γεμίζω τον πίνακα **a** με στοιχεία (τυχαίοι αριθμοί [0,9]) (*δες προηγούμενη άσκηση*)
(**επανάληψη1**) Για κάθε γραμμή i (από 0 έως N-1)
μηδένισε μετρητή μηδενικών για τη γραμμή i (**c**[i]=0)
(επανάληψη2) Για κάθε στήλη j (από 0 έως M-1)
Αν **a**`[i][j]` είναι ίσο με μηδέν αύξησε τον μετρητή κατά 1
τέλος επανάληψης 2
Τέλος επανάληψης 1
Βρες σε ποιο στοιχείο ο πίνακας **c** έχει την μεγαλύτερη τιμή (*δες προηγούμενες ασκήσεις*)

*** Όταν εκτυπώνουμε** τον αριθμό μιας στήλης ή γραμμής θεωρούμε αρίθμηση από το 1 και όχι από το 0 όπως δουλεύει εσωτερικά η C.

Εφαρμογή – Πολλαπλασιασμός πινάκων

•Ο πολλαπλασιασμός (*A.B*) δύο πινάκων γίνεται όταν ο πίνακας *A* έχει τόσες στήλες όσες γραμμές έχει ο πίνακας *B*.
•Αν ο *A* έχει *N* γραμμές και *M* στήλες και ο *B* έχει *M* γραμμές και *K* στήλες, τότε ο πίνακας *C=A.B* έχει *N* γραμμές και *K* στήλες και τα στοιχεία του δίνονται από τη σχέση

$$C_{ij} = \sum_{m=1}^{M} A_{im} B_{mj}$$

```
/* Calculate A.B */
for(i=0;i<N;i++)
      for(j=0;j<K;j++){
            sum=0;
            for(m=0;m<M;m++)  sum+=A[i][m]*B[m][j];
            C[i][j]=sum;
      }
```

Δες κώδικα ***matrixmult.c***