

Δομημένος προγραμματισμός



<https://www.geograph.org.uk>

Πίνακες στη C
(πίνακες μιας διάστασης, διανύσματα)

Περίγραμμα της ύλης που θα καλυφθεί

- Οι πίνακες στη C
- Πίνακες μιας διάστασης
- Ανάθεση τιμών σε πίνακες
- Μαζική επεξεργασία στοιχείων
- Παραδείγματα



<https://pixabay.com>

Γενικά για τους πίνακες (arrays)

- Οι πίνακες είναι μια αρκετά διαδεδομένη δομή που προσφέρεται από σχεδόν κάθε γλώσσα προγραμματισμού.
- Πρόκειται για μια διατεταγμένη ακολουθία δεδομένων ίδιου τύπου στα οποία αναφερόμαστε με ένα όνομα.
 - δηλαδή με μια μεταβλητή, μπορούμε να δηλώσουμε πολλές μεταβλητές.
- Η ιδέα αυτού της μαζικής αναφοράς στοιχείων, προέρχεται από τα μαθηματικά (διανύσματα, πίνακες).
- Υλοποιούνται με τη διαδοχική δέσμευση κελιών της κύριας μνήμης κατά τη διάρκεια της εκτέλεσης του προγράμματος.
- Μας δίνουν την δυνατότητα αποτελεσματικής οργάνωσης πολλών όμοιου τύπου δεδομένων.
- Η επεξεργασία τους γίνεται μαζικά με χρήση των εντολών επανάληψης.

Χαρακτηριστικά στοιχεία των πινάκων

- Βασικά χαρακτηριστικά:
 - Ονομασία
 - Διάσταση (1, 2, ...πολλών διαστάσεων)
 - Τύπος δεδομένων που φιλοξενούνται (int, float, char,...)
 - Αριθμός των στοιχείων σε κάθε διάσταση του.
- Ένα στοιχείο ενός πίνακα διακρίνεται από τα υπόλοιπα με τις συντεταγμένες του (εντοπισμός θέσης).
- Η διάσταση προσδιορίζει και τον αριθμό των ακεραίων δεικτών θέσης που χρειάζονται για γίνει διακριτό κάποιο στοιχείο του πίνακα (δηλαδή, οι συντεταγμένες του στοιχείου):
 - για πίνακες 1 διάστασης απαιτείται 1 δείκτης θέσης: A_i
 - για πίνακες 2 διαστάσεων απαιτούνται 2 δείκτες θέσης: $A_{i,j}$
 - κ.ο.κ

Πίνακες μιας διάστασης

- Ένας πίνακας μιας διάστασης είναι ουσιαστικά ένα διάνυσμα.

Ένας πίνακας μιας διάστασης με όνομα A που φιλοξενεί 10 πραγματικούς αριθμούς (βαθμούς φοιτητών) έχει τη δομή:

A =

5.4	3.2	4.5	6.7	6.5	8.2	9.3	4.3	7.3	5.6
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Βαθμός 1^{ου} φοιτητή

Βαθμός 2^{ου} φοιτητή

Βαθμός 10^{ου} φοιτητή

- Αναφορά σε κάποιο στοιχείο του πίνακα (διάκριση) γίνεται με τη βοήθεια του δείκτη θέσης του στοιχείου:

A_1 : 1^ο στοιχείο (5.4), A_5 : 5^ο στοιχείο (6.5)

- Μαζική αναφορά:

A_j : το $j^{\text{ο}}$ στοιχείο, όπου $j=1,\dots,10$ (με εντολές επανάληψης)

Δήλωση πινάκων στη C

- Η δήλωση γίνεται ως εξής:

Τύπος_δεδομένων Όνομα_Πίνακα [Πλήθος_στοιχείων]

- Παραδείγματα:

float vathmoi[10]; //για φιλοξενία 10 πραγματικών

int foitites[20]; //για φιλοξενία 20 ακεραίων

- Συχνά, για το πλήθος των στοιχείων ενός πίνακα χρησιμοποιείται η οδηγία *#define*:

#define N 10

float vathmoi [N];

- Το μέγιστο πλήθος στοιχείων εξαρτάται από τον compiler.
- Βρίσκουμε το μέγεθος ενός πίνακα με τον τελεστή *sizeof()*:

sizeof(vathmoi)/sizeof(float) // πλήθος στοιχείων N

Αναφορά σε στοιχεία ενός πίνακα

- Με το όνομα του πίνακα και τον δείκτη θέσης του στοιχείου μέσα σε αγκύλες []: *vathmoi [j]*
 - Το πρώτο στοιχείο: *vathmoi [0]*
 - Το δεύτερο στοιχείο: *vathmoi [1]*
 - Το τελευταίο στοιχείο: *vathmoi [9]*
- Ο δείκτης θέσης εκτός από ακέραιος μπορεί να είναι μια ακέραια μεταβλητή ή μια αριθμητική έκφραση με ακέραιο αποτέλεσμα:

j=2; i=4;

vathmoi [i] = 6.5; // το 5ο στοιχείο

vathmoi [i+j] = 9.3; // το 7ο στοιχείο

float vathmoi[10];

<i>vathmoi[0]</i>	5.4
<i>vathmoi[1]</i>	3.2
<i>vathmoi[2]</i>	4.5
<i>vathmoi[3]</i>	6.7
<i>vathmoi[4]</i>	6.5
<i>vathmoi[5]</i>	8.2
<i>vathmoi[6]</i>	9.3
<i>vathmoi[7]</i>	4.3
<i>vathmoi[8]</i>	7.3
<i>vathmoi[9]</i>	5.6

Το όνομα ενός πίνακα χωρίς τις αγκύλες δείχνει στο 1^ο στοιχείο του πίνακα (δείκτης μνήμης).

Απόδοση τιμών (πίνακες)

- Κατά την διάρκεια της εκτέλεσης του προγράμματος:

pinakas [j] = 8;

- Με την δήλωση του πίνακα (απόδοση αρχικών τιμών):

int A[5] = {5,8,2,6,3};

(A[0]=5; A[1]=8; A[2]=2; A[3]=6; A[4]=3;)

- Αν δεν υπάρχουν όλες οι τιμές:

int A[5] = {5,8,2};

(A[0]=5; A[1]=8; A[2]=2; A[3]=0; A[4]=0;)

- Απόδοση αρχικών μηδενικών τιμών:

int A[5] = {0};

- Είναι δυνατόν να μη δηλωθεί το πλήθος των στοιχείων όταν αποδίδονται αρχικές τιμές:

int A[] = {5,8,2,6,3};

Αρχικές τιμές σε στοιχεία ενός πίνακα

```
#include <stdio.h>
main()
{
    int i, N;
    float pin[] = {2.3, 3.1, 4.0, 5.7, 6.4, 7.5, 8.2};
    //υπολογίζω πόσα στοιχεία έχει ο πίνακας
    N= sizeof(pin)/sizeof(float);
    // εμφάνιση των στοιχείων του πίνακα στη οθόνη
    printf("\nYour array is:\n");
    for (i=0;i<N; i++)
        printf("%4.2f\n", pin[i]);
}
```

Τι αλλαγές πρέπει να γίνουν αν πρόκειται για πίνακα ακεραίων;

Διάβασμα ενός πίνακα(πληκτρολόγιο)

```
#include <stdio.h>
#define N 5 // N ο αριθμός των στοιχείων της pin[]
main()
{
    int i, pin[N];
    for (i=0;i<N; i++) { //ανάγνωση της pin[]
        printf("give me the no. %d element:",i+1);
        scanf("%d",&pin[i]);
    }
    printf("\nYou have just read:\n");
    for (i=0;i<N; i++) // εμφάνιση στη οθόνη
        printf("%d\n", pin[i]);
    getchar();getchar();
}
```

Αποδίδοντας τυχαίες τιμές

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10 // ο αριθμός των στοιχείων
#define SIZE 50
main()
{
    int i, pin[N];
    //αρχικοποίηση της rand()
    srand(time(NULL));
    for(i=0;i<N;i++)
        pin[i] = 1+rand()%SIZE; //τυχαίοι 1-50
    for(i=0;i<N;i++)
        printf("%d ", pin[i]);
    printf("\n");
    system("PAUSE");
}
```

Δημιουργούμε τυχαίους αριθμούς με την συνάρτηση *rand()* η οποία επιστρέφει έναν ακέραιο από 0 έως μια σταθερά που βρίσκεται στο αρχείο κεφαλίδας *stdlib.h* (*RAND_MAX* που συνήθως είναι 32767).

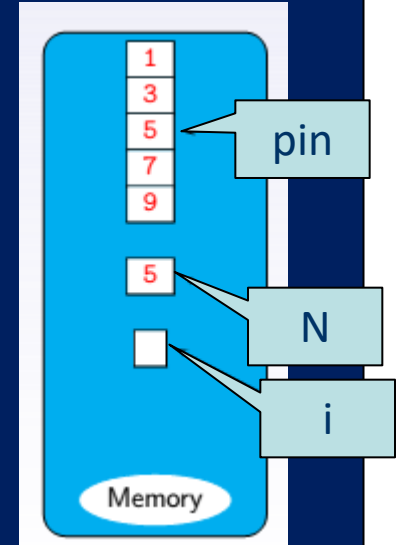
Πριν τη χρήση της *rand()* πρέπει να κληθεί από το πρόγραμμα σας η συνάρτηση *srand()*, η οποία αρχικοποιεί την γεννήτρια τυχαίων αριθμών *rand()*.

Για να είναι η ακολουθία των τυχαίων αριθμών διαφορετική, κάθε φορά που την καλούμε την *srand()*, της δίνουμε σαν παράμετρο (όρισμα) τον χρόνο του συστήματος *time(NULL)*.

– αρχείο κεφαλίδας: *<time.h>*

Αντίστροφη εμφάνιση των στοιχείων ενός πίνακα

```
#include <stdio.h>
#define N 5
main()
{
    int i, pin[N];
    for (i=0;i<N; i++) { //ανάγνωση του πίνακα
        printf("give me the no. %d element:",i+1);
        scanf("%d",&pin[i]);
    }
    printf("\nYou have just read in reverse:\n");
    for (i=N-1;i>=0; i--) // εμφάνιση (αντίστροφα) στη οθόνη
        printf("%d\n", pin[i]);
}
```



Αν δεν γνωρίζουμε πόσα στοιχεία έχει ο πίνακας...

```
#include <stdio.h>
#define N 100 //δίνουμε μια πολύ μεγάλη τιμή στο N
main()
{
    int i, n, pin[N]; //δηλώνω ένα μεγάλο πίνακα (N στοιχείων)
    printf("# of elements –pin:");
    scanf("%d", &n); //το πλήθος των στοιχείων της pin που θα χρησιμοποιηθούν
    for (i=0;i<n; i++) { //ανάγνωση των n στοιχείων της pin
        printf("pin -->give me the no. %d element:",i+1);
        scanf("%d", &pin[i]);
    }
    printf("\nYou have just read :\n");
    for(i=0;i<n;i++)
        printf("%d ", pin[i]);
}
```

Στη C99 μπορούμε να δημιουργήσουμε πίνακες δίνοντας τον αριθμό των στοιχείων τους:
printf("# of elements –pin:");
scanf("%d", &n);
int pin[n]; // δημιουργία πίνακα!
Όμως το πλήθος των στοιχείων δεν μπορεί να αλλάξει όταν εκτελείται το πρόγραμμα.

Εύρεση μέσου όρου των στοιχείων ενός πίνακα

```
int i, pin[N], sum;
float MO;    //Ο μέσος όρος είναι πάντα float...
for (i=0;i<N; i++) {    //ανάγνωση πίνακα
    printf("give me the no. %d element:",i+1);
    scanf("%d",&pin[i]);
}
sum=0;        //υπολογισμός αθροίσματος
for (i=0;i<N; i++)
    sum=sum+pin[i];
MO= (float) sum /N;    // υπολογισμός Μέσου Όρου
printf("SUM=%d\n", sum);
printf("MO=%f\n", MO);
```

Βέβαια πρέπει:
#define N 10

Πόσα στοιχεία ενός πίνακα είναι μικρότερα του 100

```
int i, pin[N], M; //M: μετρητής των στοιχείων που
                  // χαρακτηρίζονται από μια ιδιότητα (<100)
for (i=0;i<N; i++) { //ανάγνωση του πίνακα
    printf("give me the no. %d element:",i+1);
    scanf("%d", &pin[i]);
}
M=0; //αρχική τιμή στο μετρητή M
for (i=0;i<N; i++)
    if (pin[i] < 100)
        M++;
if (M==0)
    printf(" Δεν βρήκα κανένα στοιχείο μικρότερο του 100\n");
else
    printf("Βρήκα %d στοιχεία μικρότερα του 100\n", M );
```

Και εδώ πρέπει:
#define N 10

Πόσα στοιχεία είναι μεγαλύτερα του ΜΟ

```
int i, pin[N], sum, M=0;
float MO;
for (i=0;i<N; i++)           //ανάγνωση του πίνακα
{
    printf("give me the no. %d element:", i+1);
    scanf("%d",&pin[i]);
}
sum=0;                       //υπολογισμός αθροίσματος
for (i=0;i<N; i++)
    sum=sum+pin[i];
MO=(float) sum /N;           // υπολογισμός Μέσου Όρου
for (i=0;i<N; i++)
    if (pin[i] > MO) M++;
printf("Found %d elements greater than MO=%f\n", M, MO );
```

Μην ξεχνάμε το:
#define N 10

Ποια είναι η θέση του μικρότερου στοιχείου

```
int i, pin[N], MIN, iMIN;
for (i=0; i<N; i++) {
    printf("give me the no. %d element:", i+1);
    scanf("%d", &pin[i]);
}
MIN=pin[0];
iMIN=0; //η θέση του μικρότερου στοιχείου
for (i=1; i<N; i++)
    if (MIN>pin[i]) {
        MIN=pin[i];
        iMIN=i; //αλλάζει η θέση του μικρότερου στοιχείου
    }
printf ("MIN=%d at position %d\n", MIN, iMIN+1);
```

printf ("MIN=%d at position %d", pin[iMIN], iMIN);

Μην ξεχνάμε το:
#define N 10

Εναλλακτικά:
if (pin[iMIN]>pin[i]) iMIN=i;

Αντιγράφοντας ένα πίνακα σε έναν άλλο (νέο)

```
int i, pinA[N], pinB[N] ;
for (i=0;i<N; i++)          //ανάγνωση pinA[]
{
    printf("give me the no. %d element:",i+1);
    scanf("%d", &pinA[i]);
}
for (i=0;i<N; i++) // αντιγράφω τα στοιχεία του pinA στον pinB
    pinB[i] = pinA[i];

printf("\n Printing the new:\n");
for (i=0;i<N; i++) // εμφάνιση στη οθόνη του pinB
    printf("%d\n", pinB[i]);
}
```

Λείπει το:

`#define N 10`

Ενώνοντας 2 πίνακες (ο ένας στο τέλος του άλλου)

```
int pinA[N1], pinB[N2];
printf("# of elements -PinA:"); scanf("%d", &n1);
for (i=0;i<n1; i++) {           //Ανάγνωση του pinA
    printf("PinA -->give me the no. %d element:",i+1);
    scanf("%d", &pinA[i]);
}
printf("# elements -pinB:"); scanf("%d", &n2);
for (i=0;i<n2; i++){           // Ανάγνωση του pinB
    printf("PinB -->give me the no. %d element:",i+1);
    scanf("%d", &pinB[i]);
}
for (i=0;i<n2; i++)           //Ο pinB στο τέλος του pinA
    pinA[n1+i]= pinB[i];
printf("\n\n Printing the new pinA:\n");
for (i=0;i<n1+n2; i++)       //Εμφάνιση του νέου pinA
    printf("%d\n", pinA[i]);
```

Πρέπει πχ:

```
#define N1 100
```

```
#define N2 50
```

Το N1 πρέπει να
αρκετά μεγαλύτερος
του N2, διότι ο 1^{ος}
πίνακας θα πάρει και
τα στοιχεία του 2^{ου}
πίνακα.

Αντιστρέφοντας ένα μονοδιάστατο πίνακα

```
int i, j, A[N], ;  
for (i=0; i<N; i++) {           //ανάγνωση πίνακα  
    printf("give me the no. %d element:", i+1);  
    scanf("%d", &A[i]);  
}  
j = N-1; // j δείχνει στο τελευταίο στοιχείο  
i = 0;   // i δείχνει στο 1ο στοιχείο  
while(i < j) {  
    temp = A[i]; //κρατώ το A[i]  
    A[i] = A[j]; // αλλάζω τη τιμή  
    A[j] = temp; // δίνω τη τιμή που έχω κρατήσει στο A[j]  
    i++;        // αυξάνω το i  
    j--;        // μειώνω το j  
}  
printf("\n Printing the new:\n");  
for (i=0; i<N; i++) // εμφάνιση στη οθόνη  
    printf("%d\n", A[i]);
```

Παράδειγμα

Ο πίνακας:

A = (3,4,5,2,6,7,8)

θα γίνει:

A = (8,7,6,2,5,4,3)

Ξανά το:

#define N 10

Άλλος τρόπος:

```
for (i=0; i<N/2; i++) {  
    temp=A[i];  
    A[i]= A[N-i-1];  
    A[N-i-1]=temp;  
}
```

Παρεμβάλλοντας ένα νέο στοιχείο

```
int i, A[N], n, L, newa; // N: το μέγεθος του πίνακα – μεγάλος αριθμός
printf("Enter no of elements :");
scanf("%d",&n);        //n: πόσα στοιχεία θέλουμε να χρησιμοποιήσουμε
for (i=0;i<n; i++) {    //ανάγνωση των n στοιχείων
    printf("give me the no. %d element:", i+1); scanf("%d", &A[i]);
}
// διαβάζει το νέο στοιχείο (new) και τη θέση (L) που θέλουμε να παρεμβάλουμε
printf("\nEnter the new element to be inserted :"); scanf("%d",&newa);
printf("\nEnter the location:"); scanf("%d",&L);
for (i = n ; i >= L ; i--)    // δημιουργώ χώρο για το νέο στοιχείο
    A[i] = A[i-1];
n++;    // αυξάνω το n
A[L-1] = newa; //παρεμβάλω το στοιχείο
printf("\n Printing the new:\n");
for (i=0;i<n; i++)          // εμφάνιση στη οθόνη του νέου πίνακα
    printf("%d\n", A[i]);
```

Διαγράφοντας ένα στοιχείο

```
int i, A[N], n, L; // N: το μέγεθος του πίνακα – μεγάλος αριθμός
printf("Enter no of elements :");
scanf("%d",&n);    //n: πόσα στοιχεία θέλουμε να έχει αρχικά ο πίνακας n<N
for (i=0;i<n; i++) { //ανάγνωση των n στοιχείων του πίνακα
    printf("give me the no. %d element:",i+1); scanf("%d", &A[i]);
}
// διαβάζει τη θέση του στοιχείου που θέλουμε να διαγράψουμε
printf("\nEnter the location of the element to be deleted:");
scanf("%d",&L);
while(L < n) { // διαγράφω το στοιχείο
    A[L-1]=A[L];
    L++;
}
n--;
printf("\n Printing the new:\n");
for (i=0;i<n; i++)    // εμφάνιση στη οθόνη του νέου πίνακα
    printf("%d\n", A[i]);
```

Ενώνοντας 2 πίνακες σε έναν τρίτο

```
int i, pinA[N1], pinB[N2], pinNEW[N1+N2];
for (i=0;i<N1; i++) { //ανάγνωση pinA
    printf("PinA -->give me the no. %d element:",i+1);
    scanf("%d", &pinA[i]);
}
for (i=0;i<N2; i++){ //ανάγνωση pinB
    printf("PinB -->give me the no. %d element:",i+1);
    scanf("%d", &pinB[i]);
}

for (i=0;i<N1; i++) //Ο pinA στον pinNEW
    pinNEW[i]= pinA[i];
for (i=0;i<N2; i++) //Ο pinB στον pinNEW
    pinNEW[N1+i]= pinB[i];

printf("\n Printing the pinNEW:\n");
for (i=0;i<N1+N2; i++) //Εμφανίζω τον pinNEW
    printf("%d\n", pinNEW[i]);
```

Πρέπει:

```
#define N1 5
#define N2 5
```

Άλλος τρόπος δημιουργίας του pinNEW:

```
for (i=0;i<N1+N2; i++)
    if (i<N1) pinNEW[i]=
        pinA[i];
    else pinNEW[i]=pinB[i-
        N1];
```