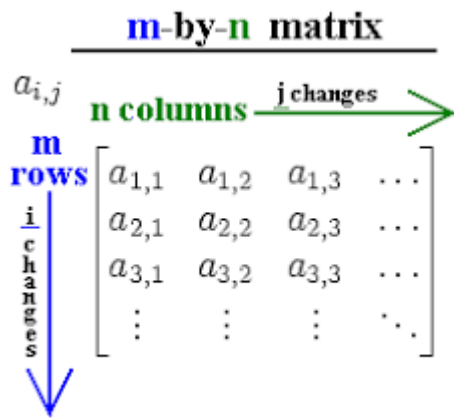


Δομημένος προγραμματισμός



<https://en.wikipedia.org>

Πίνακες 2 διαστάσεων

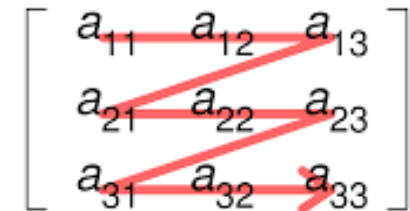
Περίγραμμα της ύλης που θα καλυφθεί

- Οι πίνακες 2 διαστάσεων στη C
- Δήλωση δισδιάστατων πινάκων
- Ανάθεση αρχικών τιμών
- Παραδείγματα

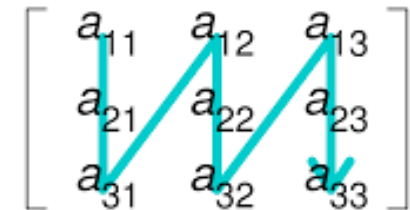


<https://www.pexels.com>

Row-major order



Column-major order



<https://en.wikipedia.org>

Δήλωση δισδιάστατου πίνακα

- Οι δισδιάστατοι πίνακες χαρακτηρίζονται από τις γραμμές και τις στήλες (οι διαστάσεις) τους.
- Πίνακας 2 διαστάσεων: 3 γραμμών και 4 στηλών:

		1	2	3	4	Στήλες
Γραμμές	1	A_{11}	A_{12}	A_{13}	A_{14}	
	2	A_{21}	A_{22}	A_{23}	A_{24}	
	3	A_{31}	A_{32}	A_{33}	A_{34}	

- Πρόκειται για μονοδιάστατο πίνακα, όπου κάθε στοιχείο του είναι ένας πίνακας.

- Η δήλωση στη C έχει ως εξής:

Στη C η αρίθμηση αρχίζει από το 0

Τύπος Όνομα_πίνακα [Πλήθος_γραμμών] [Πλήθος_στηλών];

Παράδειγμα δήλωσης: `int A [3] [4];`

Η δέσμευση στοιχείων ενός δισδιάστατου πίνακα στη C

- Με τη δήλωση ενός πίνακα δεσμεύονται αντίστοιχα τμήματα μνήμης.

int A[3][4]; // ή int A[M][N] αν έχουν οριστεί τα M & N

δεσμεύονται 12 θέσεις ακεραίων ή 48 bytes (1 int έχει εύρος 4 bytes)

char O[10][30]; // ή char O[M][N] αν έχουν οριστεί τα M & N

δεσμεύονται 300 θέσεις χαρακτήρων ή 300 bytes (1 char έχει εύρος 1 byte)

float P [4][5]; // ή float P[M][N] αν έχουν οριστεί τα M & N

δεσμεύονται 20 θέσεις πραγματικών ή 80 bytes (1 float έχει εύρος 4 bytes)

- Πλήθος στοιχείων = (πλήθος_γραμμών) x (πλήθος_στηλών)

- Αριθμός bytes = (εύρος_τύπου) x (πλήθος_στοιχείων):

bytes = sizeof(τύπος)(πλήθος_γραμμών)*(πλήθος_στηλών)*

- Αποθηκεύονται διαδοχικά πρώτα τα στοιχεία της 1ης γραμμής, στη συνέχεια τα στοιχεία της 2ης γραμμής κ.ο.κ.

Αναφορά σε στοιχεία ενός δισδιάστατου πίνακα

- Αναφερόμαστε σε κάποιο στοιχείο ενός δισδιάστατου πίνακα με το όνομα του πίνακα με τους αριθμούς της γραμμής και της στήλης σε αγκύλες: $A[i][j]$

		0	1	2	3	Στήλες
Γραμμές	0	$A[0][0]$	$A[0][1]$	$A[0][2]$	$A[0][3]$	
	1	$A[1][0]$	$A[1][1]$	$A[1][2]$	$A[1][3]$	
	2	$A[2][0]$	$A[2][1]$	$A[2][2]$	$A[2][3]$	

- Στη C η αρίθμηση των γραμμών και των στηλών ξεκινάει από το 0. Σε ένα πίνακα διαστάσεων $M \times N$ το 1^ο στοιχείο του είναι το $A[0][0]$ και το τελευταίο $A[M-1][N-1]$.
- Ο υπολογισμός της θέσης μνήμης κάποιου στοιχείου $A[i][j]$ ενός πίνακα $A[M][N]$ γίνεται σύμφωνα με τον τύπο:

$$position = i * N + j + 1$$

Αρχίζοντας την αρίθμηση από το 1

Απόδοση τιμών σε δισδιάστατους πίνακες - I

- Κατά την διάρκεια της εκτέλεσης του προγράμματος:

pinakas [i][j] = 8;

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

int A[2][3] = { {5,8,2}, {6, 3,7} };

ή

int A[2][3] = {5, 8, 2, 6, 3, 7};
A[0][0]=5; A[0][1]=8; A[0][2]=2;
A[1][0]=6; A[1][1]=3; A[1][2]=7;

- Αν δεν υπάρχουν όλες οι τιμές:

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

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

- Απόδοση αρχικών μηδενικών τιμών:

int A[2][3] = {0};

Απόδοση τιμών σε δισδιάστατους πίνακες - II

- Είναι δυνατόν να μη δηλωθεί το πλήθος των στοιχείων των γραμμών, όταν αποδίδονται αρχικές τιμές. Το πλήθος των στηλών όμως πρέπει υποχρεωτικά να δηλωθεί:

int A[][3] = {{5,8,2}, {6, 3,7}};

ή

int A[][3] = {5, 8, 2, 6, 3, 7};

(θα δημιουργηθεί ένας δισδιάστατος πίνακας ακεραίων με 2 γραμμές και 3 στήλες)

- Για την ανάγνωση (αλλά και για τη μαζική επεξεργασία) των στοιχείων ενός πίνακα χρησιμοποιούμε ένθετες εντολές επανάληψης:

*for (i=0; i<M; i++)
for (j=0; j<N; j++)*

Προσοχή:

Οι compilers της C δεν ελέγχουν αν οι δείκτες i & j υπερβαίνουν τα όρια.

Διάβασμα ενός δισδιάστατου πίνακα (πληκτρολόγιο)

```
#include <stdio.h>
#define M 4
#define N 5
main() {
    int i, j, pin[M][N];
    for (i=0;i<M; i++)    //ανάγνωση πίνακα
        for (j=0;j<N; j++) {
            printf("give me the no. %d,%d element:",i+1, j+1);
            scanf("%d",&pin[i][j]);
        }
    printf("\nYou have just read:\n");
    for (i=0;i<M; i++) {    // εμφάνιση στη οθόνη
        for (j=0;j<N; j++)
            printf("\t%d", pin[i][j]);
        printf("\n");    // γιατί;
    }
}
```

Φυσικά και με την συνάρτηση
rand():
pin[i][j] = rand()%SIZE;

Δημιουργία ενός μοναδιαίου πίνακα ($M \times M$)

```
#include <stdio.h>
#define M 4
main()
{
    int i, j, pin[M][M] = {0}; // όλα 0
    for (i=0;i<M; i++) //δημιουργία πίνακα
        pin[i][i]=1; //διαγώνιος
    printf("\nYou have just creat:\n");
    for (i=0;i<M; i++) { // εμφάνιση στη οθόνη
        for (j=0;j<M; j++)
            printf("\t%d ", pin[i][j]);
        printf("\n"); // γιατί;
    }
}
```

Μοναδιαίος 4x4:

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Εύρεση μέσου όρου στοιχείων δισδιάστατου πίνακα

```
#define M 4
#define N 5
main()
{
    int i, j, pin[M][N], sum;
    float MO;
    for (i=0;i<M; i++) //ανάγνωση πίνακα
        for (j=0;j<N; j++) {
            printf("give me the no. %d,%d element:",i+1, j+1);
            scanf("%d",&pin[i][j]);
        }
    for (sum=0, i=0;i<M; i++) //υπολογισμός αθροίσματος
        for (j=0;j<N; j++)
            sum=sum+pin[i][j];
    MO=(float) sum /(M*N); // υπολογισμός Μέσου Όρου
    printf("MO=%f\n", MO);
}
```

Ποια είναι η θέση του μικρότερου στοιχείου

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

Εύρεση μεγαλύτερου και μικρότερου κάθε γραμμής

```
int i, j, pin[M][N], MIN, MAX;
for (i=0;i<M; i++) //ανάγνωση πίνακα
    for (j=0;j<N; j++) {
        printf("give me the no. %d,%d element:",i+1, j+1);
        scanf("%d",&pin[i][j]);
    }
for (i=0;i<M; i++) {
    MIN=MAX=pin[i][0]; // αρχικές τιμές σε κάθε γραμμή
    for (j=0;j<N; j++) {
        if (MAX < pin[i][j]) MAX= pin[i][j];
        if (MIN > pin[i][j]) MIN = pin[i][j];
    }
    printf ("line %d, MAX=%d, MIN=%d\n", i+1, MAX,MIN);
}
```

Εύρεση μέσου όρων των διαγώνιων τετράγωνου πίνακα

```
int i, j, pin[M][M], sum;
float MO1, MO2;
for (i=0;i<M; i++) //ανάγνωση πίνακα
    for (j=0;j<M; j++) {
        printf("give me the no. %d,%d element:",i+1, j+1);
        scanf("%d",&pin[i][j]);
    }
sum=0;                // 1η διαγώνιος
for (i=0;i<M; i++)
    sum=sum+pin[i][i];
MO1=(float)sum/M;
sum=0;                //2η διαγώνιος
for (i=0;i<M; i++) // αν βάλουμε for (i = M-1 ; i >= 0 ; --i) ?
    sum=sum +pin[i][(M-1)-i];
MO2=(float) sum/M;
printf ("1st diagonal, average: %f\n", MO1);
printf ("2nd diagonal, average: %f\n", MO2);
```

Εύρεση αθροισμάτων κάθε γραμμής και κάθε στήλης

```
for(i=0;i<m;i++) {  
    R[i]=0; //άθροισμα κάθε γραμμής  
    for(j=0;j<n;j++)  
        R[i]=R[i]+A[i][j];  
}  
for(i=0;i<n;i++){  
    C[i]=0; //άθροισμα κάθε στήλης  
    for(j=0;j<m;j++)  
        C[i]=C[i]+A[j][i];  
}  
for(i=0;i<m;i++) {  
    for(j=0;j<n;j++)  
        printf("%4d",A[i][j]);  
    printf("%8d",R[i]);  
    printf("\n");  
}  
printf("\n");  
for(j=0;j<n;j++)  
    printf("%4d",C[j]);  
}
```

```
int i, j, A[m][n],R[m],C[n];
```

- **R[]**: για την αποθήκευση των αθροισμάτων κάθε γραμμής
- **C[]** : για την αποθήκευση των αθροισμάτων κάθε στήλης

Φυσικά πρέπει πρώτα να γεμίσει ο πίνακας (διάβασμα από το πληκτρολόγιο ή με τυχαίες τιμές).