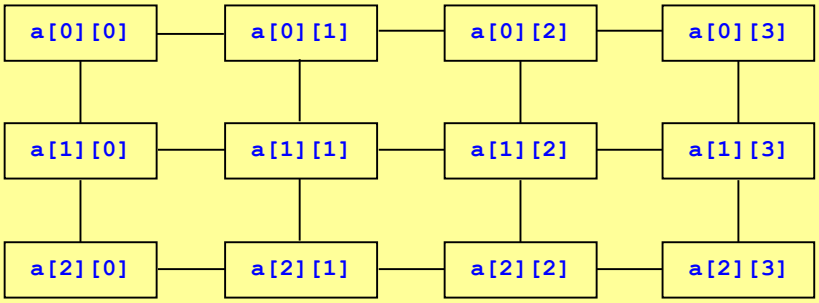
ΔΙΣΔΙΑΣΤΑΤΟΙ (2-Δ) ΠΙΝΑΚΕΣ

Η δήλωση:

```
int  a[3][4];
```

δεσμεύει 12 θέσεις μνήμης για την αποθήκευση ενός (3 x 4) 2-Δ πίνακα ακεραίων. Μπορούμε να φανταστούμε τον 2-Δ πίνακα όπως στο σχήμα:

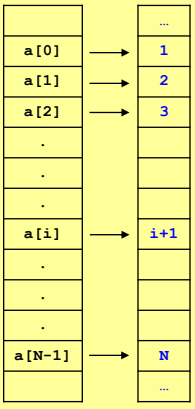


1-Δ ΠΙΝΑΚΕΣ

Έστω μονοδιάστατος πίνακας (δηλαδή, διάνυσμα) **a**, Ν στοιχείων τύπου ακεραίου και έστω ότι απαιτούνται 4 bytes ανά ακέραιο.

```
a = [1  2  3 ... N]
```

Μνήμη Η/Υ



Αν η διεύθυνση του 1ου στοιχείου είναι το ίδιο το όνομα του πίνακα (**a**) και οι δείκτες αρχίζουν από το 0, τότε:

&a[0] == a και **&a[i] = a + i*4**

ενώ αν οι δείκτες αρχίζουν από το 1:

&a[1] == a και **&a[i] = a + (i-1)*4**

Σημείωση: δεν υπάρχει καμία διαφορά ως προς την αποθήκευση αν έχουμε διάνυσμα γραμμής ή διάνυσμα στήλης.

2-Δ ΠΙΝΑΚΕΣ

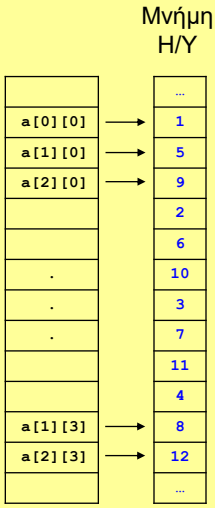
Επειδή η μνήμη ενός Η/Υ είναι γραμμική (μονο-διάστατη), ο τρόπος με τον οποίο θα αποθηκεύεται ένας 2-Δ πίνακας στη μνήμη (δηλαδή, η μετατροπή του σε κατάλληλη μονοδιάστατη δομή) εξαρτάται από την κάθε γλώσσα προγραμματισμού.

Για παράδειγμα, στην Fortran και στο Matlab, η αποθήκευση γίνεται κατά στήλες.

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

Fortran

Matlab



Αν η διεύθυνση του 1ου στοιχείου είναι το **a** , οι διαστάσεις είναι (NxM) και οι δείκτες αρχίζουν από το 0 (ή 1), τότε:

```
&a[i][j] = a + (j*N + i) * 4
```

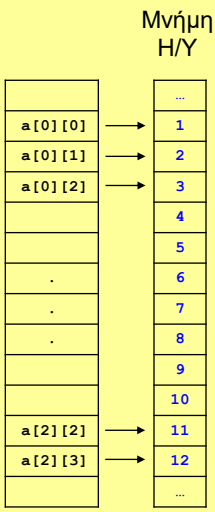
(ή `&a[i][j] = a + ((j-1)*N + (i-1)) * 4`)

2-Δ ΠΙΝΑΚΕΣ

Στην C η αποθήκευση γίνεται κατά γραμμές.

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

C



Αν η διεύθυνση του 1ου στοιχείου είναι το **a** , οι διαστάσεις είναι (NxM) και οι δείκτες αρχίζουν από το 0, τότε:

```
&a[i][j] = a + (i*M + j) * 4
```

Παράδειγμα 1

Αρχικοποίηση 2-Δ πίνακα και υπολογισμός του πίνακα B = [A A] (συνένωση του πίνακα A με τον εαυτό του ως προς τις γραμμές).

```
#include <stdio.h>
int main()
{
  int  i, j;
  int  A[2][2], B[2][4];

  /* Αρχικοποίηση του A */
  A[0][0] = 1; A[0][1] = 2; A[1][0] = 3; A[1][1] = 4;

  /* Υπολογισμός πίνακα B */
  for(i=0; i<2; i++)
    for(j=0; j<2; j++)
      B[i][j+2] = B[i][j] = A[i][j];

  return 0;
}
```

. . . συνέχεια

Παράδειγμα αρχικοποίησης του πίνακα A από το πληκτρολόγιο:

```
  . . .

  /* Αρχικοποίηση του A από το πληκτρολόγιο*/

  for(i=0; i<2; i++)
    for(j=0; j<2; j++)
    {
      printf("\nA[%d][%d] = ",i,j);
      scanf("%d",&A[i][j]);
    }

  . . .
```

Παράδειγμα 2

Υπολογισμός ανάστροφου πίνακα B = A^T (ο ανάστροφος ενός πίνακα A έχει για στήλες τις γραμμές του A) και εμφάνιση των A και B στην οθόνη.

```
#include <stdio.h>
int main()
{
  int  i, j;
  int  A[2][3]={{1,2,3}, {4,5,6}};
  int  B[3][2]; /* anastrofos */

  /* Υπολογισμός ανάστροφου πίνακα B */
  for(i=0; i<2; i++)
    for(j=0; j<3; j++)
      B[j][i] = A[i][j];
```

Αρχικοποίηση μαζί με τη δήλωση του **A**. *Προσοχή: ο διαχωριστής είναι το κόμμα ',' και όχι το κενό!*

συνέχεια …

. . . συνέχεια

```
  printf("PINAKAS A:\n");
  for(i=0; i<2; i++)
  {
    for(j=0; j<3; j++)
      printf("%d ",A[i][j]);
    printf("\n");
  }

  printf("\nANASTROFOS A:\n");
  for(i=0; i<3; i++)
  {
    for(j=0; j<2; j++)
      printf("%d ",B[i][j]);
    printf("\n");
  }

   return 0;
}
```

Έξοδος προγράμματος:

```
PINAKAS A:
1 2 3
4 5 6

ANASTROFOS A:
1 4
2 5
3 6
```