ΠΙΝΑΚΕΣ

- Ένας πίνακας είναι μια συλλογή δεδομένων που αποτελείται από στοιχεία του ιδίου τύπου (π.χ. πίνακας ακεραίων).
- Τα στοιχεία ενός πίνακα αποθηκεύονται σε συνεχόμενες θέσεις μνήμης
- Στη C ο δείκτης που προσδιορίζει τη θέση του πρώτου στοιχείου ενός πίνακα N στοιχείων είναι το 0 και του τελευταίου στοιχείου το N-1. Αντιθέτως, στην Pascal και στο MATLAB, οι δείκτες ξεκινούν από το 1.
- Παράδειγμα δήλωσης πίνακα 10 ακεραίων με όνομα **a** και πίνακα 5 πραγματικών αριθμών με όνομα **weight**:

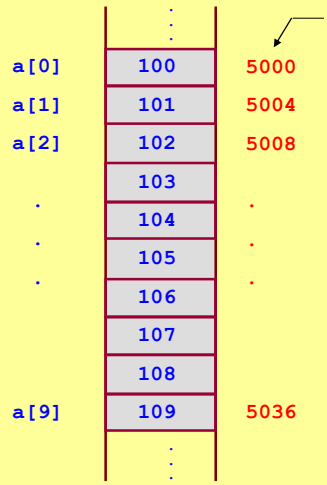
```
int a[10];
float weight[5];
```

Τα στοιχεία του πίνακα a είναι τα: `a[0], ..., a[9].`
Παράδειγμα προσπέλασης στοιχείων πίνακα: `weight[2] = 31.25;`

ΠΙΝΑΚΕΣ

- Η δήλωση ενός πίνακα δεσμεύει συνεχόμενες θέσεις στη μνήμη.
- Παράδειγμα:

```
int a[10];
```



Διευθύνσεις των στοιχείων του πίνακα στη μνήμη:

Απαιτούμενος χώρος στη μνήμη:

10 x 4 bytes = 40 bytes

ΠΙΝΑΚΕΣ

- Με τους πίνακες μπορούμε να ομαδοποιούμε πολλές μεταβλητές με το ίδιο όνομα. Π.χ. Έστω ότι θέλουμε να διαβάσουμε 100 πραγματικούς αριθμούς σε ένα πρόγραμμα, τότε αντί να δηλώσουμε 100 μεταβλητές (`float ar1, ar2, ar3, …, ar100;`) και να επαναλαμβάνουμε την είσοδο/έξοδο και επεξεργασία των αριθμών με πανομοιότυπο τρόπο, ορίζουμε έναν πίνακα τύπου float με τόσες θέσεις όσοι και οι αριθμοί: `float ar[100];` Τώρα, η είσοδος, η έξοδος και η επεξεργασία των αριθμών μπορεί να γίνει κομψά και αποτελεσματικά με χρήση των εντολών επανάληψης (βρόχων).
- Προσπέλαση στοιχείων πίνακα:

```
a[0]=0;  /* εκχώρηση τιμής στο 1ο στοιχείο του πίνακα a */
j=5;  a[j]=j;  /* εκχώρηση στο 6ο στοιχείο του a */
i=2; j=1; a[i*3+j]=a[i]+a[j];  /* υπολογισμός στοιχείου
                        πίνακα από αποτίμηση παράστασης */
```

Αρχικοποίηση πίνακα

```
/* Arxikopoihsh kai upologismos plithous psifiwn pinaka apo
   metaglwttisth */

#include <stdio.h>
int main()
{
  int i;
  int days[]={31,28,31,30,31,30,31,31,30,31,30,31};

  for(i=0; i<12; i++)
    printf("O mhnas %d exei %d meres\n",i+1,days[i]);
  return 0;
}
```

```
O mhnas 1 exei 31 meres
O mhnas 2 exei 28 meres
O mhnas 3 exei 31 meres
...
O mhnas 12 exei 31 meres
```

Ταξινόμηση αριθμών

```
#include <stdio.h>
int main()
{
 int i,j;
 float m;
 float a[6] = {-1.5, 3.4, -1.6e2, 9., .2, 2.7e-1};

 for(i=0; i<5; i++)
   for(j=i+1; j<6; j++)      /* εμφωλευμένη for */
     if(a[i] > a[j])
     {
       m = a[i];
       a[i] = a[j];
       a[j] = m;
     }

 for(i=0; i<6; i++)
   printf("%7.2f ",a[i]);
 printf("\n");

 return 0;
}
```

```
-160.00   -1.50    0.20    0.27    3.40    9.00
```

Εισαγωγή δεδομένων σε πίνακες

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
int main()
{
  int   i, pin1[10], pin2[10];
  int   flag=TRUE;

  printf("EISAGWGH DEDOMENWN\n");

  i = 0;
  while (flag)    /* while (flag == TRUE) */
  {
    printf("Input 2 integer values for ");
    printf("pin1[%d], pin2[%d]:\n",i,i);
    scanf("%d%d",&pin1[i],&pin2[i]);
    i++;
    if (i == 10)
      flag = FALSE;
  }
  ...
}
```

όχι καλός τρόπος

Εισαγωγή δεδομένων σε πίνακες

```
#include <stdio.h>
int main()
{
  int   i, pin1[10], pin2[10];

  printf("EISAGWGH DEDOMENWN\n");

  i = 0;
  while (i < 10)
  {
    printf("Input 2 integer values for ");
    printf("pin1[%d], pin2[%d]:\n",i,i);
    scanf("%d %d",&pin1[i],&pin2[i]);
    i++;
  }
  ...
}
```

καλύτερος τρόπος

Εισαγωγή δεδομένων σε πίνακες

```
#include <stdio.h>
int main()
{
  int   i, pin1[10], pin2[10];

  printf("EISAGWGH DEDOMENWN\n");

  i = 0;
  do
  {
    printf("Input 2 integer values for ");
    printf("pin1[%d], pin2[%d]:\n",i,i);
    scanf("%d %d",&pin1[i],&pin2[i]);
    i++;
  } while (i < 10);
  ...
}
```

Όχι καλός τρόπος. Ο βρόχος εκτελείται τουλάχιστον 1 φορά!

Εισαγωγή δεδομένων σε πίνακες

```
#include <stdio.h>

int main()
{
  int   i, pin1[10], pin2[10];

  printf("EISAGWGH DEDOMENWN\n");

  for(i = 0; i < 10; i++)
  {
    printf("Input 2 integer values for ");
    printf("pin1[%d], pin2[%d]:\n",i,i);
    scanf("%d %d",&pin1[i],&pin2[i]);
  }

  ...
}
```

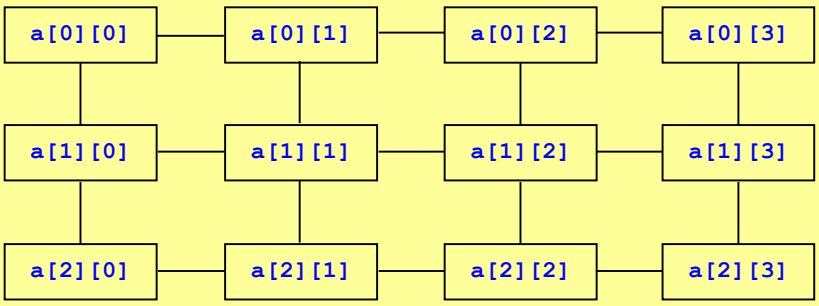
Ο καλύτερος τρόπος.

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:

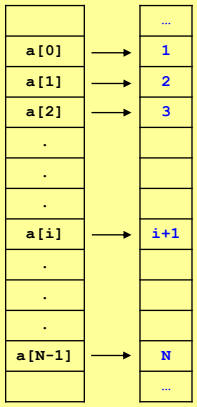


1-Δ ΠΙΝΑΚΕΣ και Διευθύνσεις Μνήμης

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**

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

2-Δ ΠΙΝΑΚΕΣ και Διευθύνσεις Μνήμης

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.

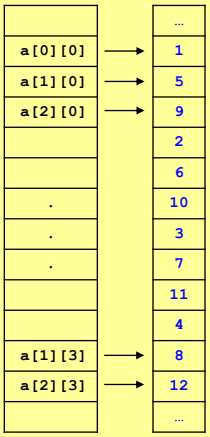
Για παράδειγμα, στην Fortran και στο Matlab, η αποθήκευση γίνεται κατά στήλες.

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

Fortran

Matlab

Μνήμη Η/Υ

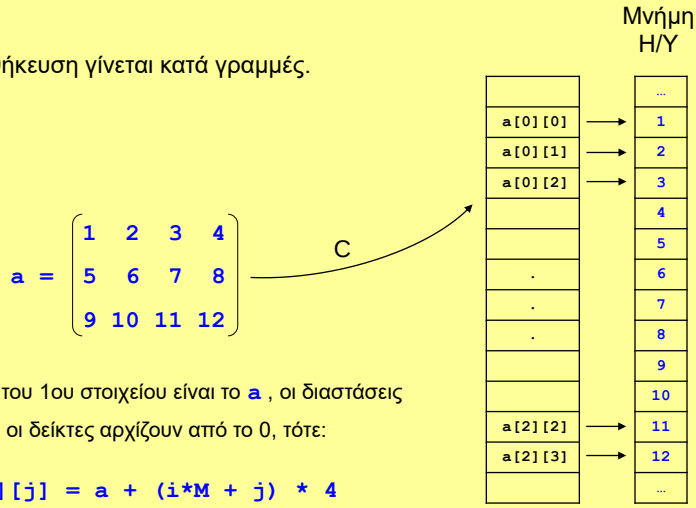


Αν η διεύθυνση του 1ου στοιχείου είναι το **a** , οι διαστάσεις είναι (NxM) και οι δείκτες αρχίζουν από το 0, τότε:

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

2-Δ ΠΙΝΑΚΕΣ και Διευθύνσεις Μνήμης

Στην C η αποθήκευση γίνεται κατά γραμμές.



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 Α από το πληκτρολόγιο:

```
. . .

/* Αρχικοποίηση του Α 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 Β = A^T (ο ανάστροφος ενός πίνακα Α έχει για στήλες τις γραμμές του Α) και εμφάνιση των Α και Β 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 Β */
  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
```