C: Από τη Θεωρία στην Εφαρμογή

Κεφάλαιο 7ο
Πίνακες

Γ. Σ. Τσελίκης – Ν. Δ. Τσελίκας

Πίνακες στη C

- Ένας πίνακας στη C είναι **μία δομή δεδομένων που αποτελείται από στοιχεία του ίδιου τύπου** (π.χ. πίνακας ακεραίων αριθμών, πίνακας πραγματικών αριθμών, πίνακας χαρακτήρων, ...)

- Όλοι οι πίνακες **δεσμεύουν συνεχόμενες θέσεις στη μνήμη** (στην περιοχή μνήμης που ονομάζεται **στοίβα** ή **stack**) του υπολογιστή και διακρίνονται σε πίνακες μίας διάστασης και πίνακες πολλών διαστάσεων

- Συνηθέστερα είδη είναι οι μονοδιάστατοι και οι διδιάστατοι πίνακες

Ορισμός Μονοδιάστατου Πίνακα

- Για να ορίσουμε έναν μονοδιάστατο πίνακα πρέπει να δηλώσουμε το όνομα του πίνακα, τον τύπο δεδομένων των στοιχείων του πίνακα και το πλήθος των στοιχείων του πίνακα

- Η γενική περίπτωση ορισμού ενός μονοδιάστατου πίνακα είναι:

```
τύπος_δεδομένων  όνομα_πίνακα [πλήθος_στοιχείων_πίνακα];
```

Παρατηρήσεις

- Το όνομα_πίνακα πρέπει να είναι μοναδικό (να μην υπάρχει άλλη μεταβλητή στο πρόγραμμα με το ίδιο όνομα)
- Το πλήθος_στοιχείων_πίνακα (λέγεται και μήκος του πίνακα) πρέπει να περιβάλλεται από αγκύλες [ ] αμέσως μετά το όνομα_πίνακα
- Ο τύπος_δεδομένων μπορεί να είναι οποιοσδήποτε τύπος δεδομένων (π.χ. int, float, char, κτλ…)

Παραδείγματα Ορισμών

```
int a[100];
/* πίνακας με όνομα a, ο οποίος περιέχει 100 ακέραιους
τύπου int */

double arr[5];
/* πίνακας με όνομα arr, ο οποίος περιέχει 5 πραγματικούς
αριθμούς τύπου double */

char x[2000];
/* πίνακας με όνομα x, ο οποίος περιέχει 2000 ακεραίους
τύπου char */
```

- Επαναλαμβάνεται ότι **κάθε πίνακας δεσμεύει συνεχόμενες θέσεις μνήμης** στον υπολογιστή

- Ερώτηση: Πόση μνήμη καταλαμβάνει ο κάθε ένας από τους παραπάνω πίνακες???

Στοιχεία μονοδιάστατου Πίνακα

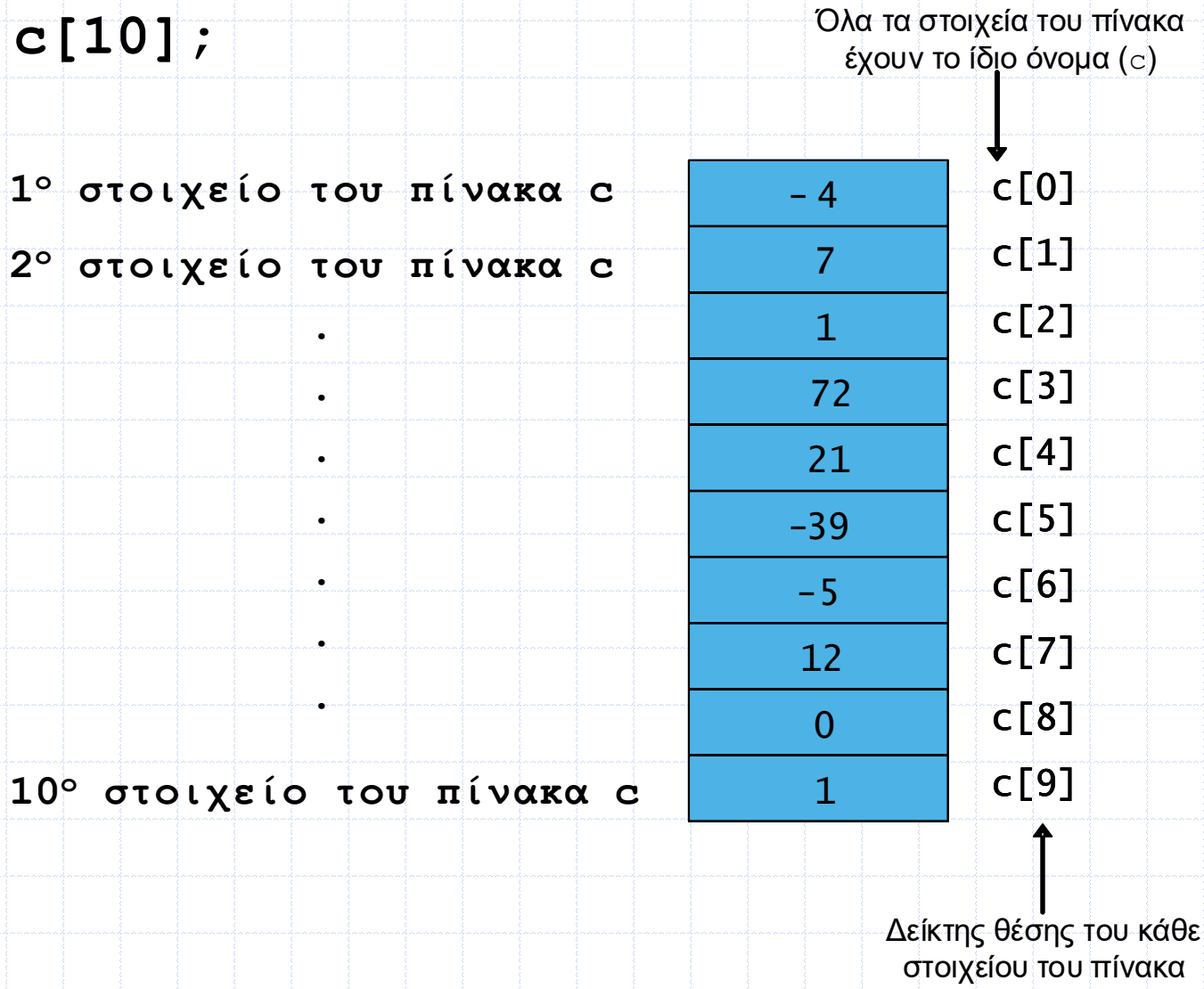
- Για να αναφερθούμε σε κάποιο στοιχείο του πίνακα γράφουμε το **όνομα του πίνακα** συνοδευόμενο από τον **δείκτη θέσης** του στοιχείου μέσα σε αγκύλες `[]`

- **Ο δείκτης θέσης** είναι ένας ακέραιος αριθμός ή μία ακέραια μεταβλητή ή έκφραση, η οποία **προσδιορίζει τη θέση** του συγκεκριμένου στοιχείου στον πίνακα

- Το πρώτο στοιχείο ενός πίνακα με μέγεθος `n` στοιχεία αποθηκεύεται στη θέση `[0]` του πίνακα, το δεύτερο στοιχείο στη θέση `[1]`, το τρίτο στη θέση `[2]`, ... κ.ο.κ., με αποτέλεσμα το τελευταίο στοιχείο να αποθηκεύεται στη θέση `[n-1]`

Γραφική Αναπαράσταση (Ι)

Π.χ. `char c[10];`



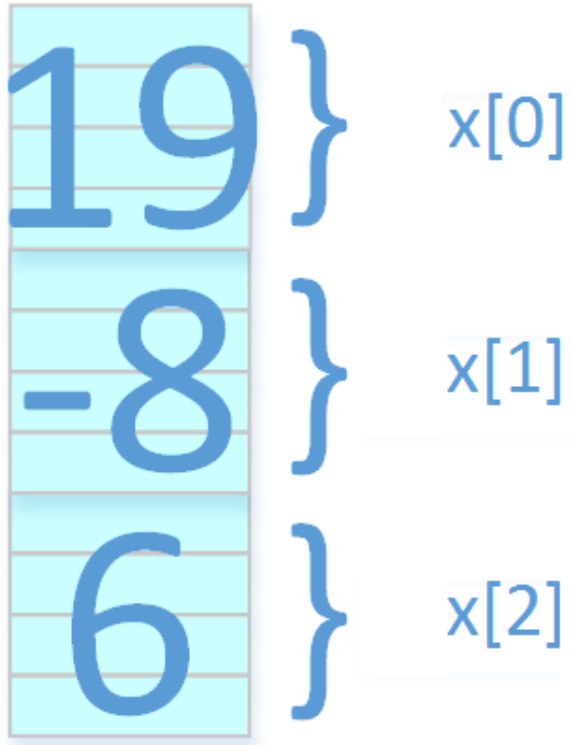
Γραφική Αναπαράσταση (ΙΙ)

Π.χ. `int x[3];`

1ο στοιχείο του πίνακα x

2ο στοιχείο του πίνακα x

3ο στοιχείο του πίνακα x



Παρατηρήσεις (I)

Σημειώστε ότι το πλήθος των στοιχείων του πίνακα (δηλαδή το μήκος του πίνακα) δεν μπορεί να αλλάξει κατά την εκτέλεση του προγράμματος. Παραμένει σταθερό.

- Για τη δήλωση του μεγέθους ενός πίνακα χρησιμοποιείται πολύ συχνά η μακροεντολή `#define` , π.χ.:

```
#define SIZE 1000
int arr[SIZE]; /* Ο μεταγλωττιστής αντικαθιστά τη
λέξη SIZE με την τιμή 1000 και δημιουργεί έναν
πίνακα με στοιχεία 1000 ακεραίους. */
```

- Σε αντίθεση με τη χρήση μακροεντολής, απαγορεύεται να χρησιμοποιηθεί σταθερά δηλωμένη με τη λέξη `const` για να προσδιορίσει το πλήθος των στοιχείων του πίνακα, π.χ., η ακόλουθη δήλωση δεν επιτρέπεται:

```
const int size = 100;
float arr[size]; /* Απαγορεύεται!!! */
```

Παρατηρήσεις (ΙΙ)

- Όταν δηλώνεται ένας πίνακας, ο μεταγλωττιστής δεσμεύει ένα κομμάτι μνήμης για να αποθηκεύσει τα στοιχεία του πίνακα
  - Οι τιμές των στοιχείων του πίνακα αποθηκεύονται η μία μετά την άλλη σε διαδοχικές θέσεις μνήμης
  - Τυπικά, αυτό το κομμάτι μνήμης δεσμεύεται από συγκεκριμένο μέρος της μνήμης που ονομάζεται **στοίβα** (**stack**), και αποδεσμεύεται όταν τερματιστεί η λειτουργία της συνάρτησης μέσα στην οποία έχει δηλωθεί ο πίνακας

- Π.χ., με την παρακάτω δήλωση ο μεταγλωττιστής δεσμεύει 40 bytes για να αποθηκεύσει τις τιμές των 10 ακεραίων στοιχείων του

```
int arr[10];
```

- Για την εύρεση του μεγέθους της δεσμευμένης μνήμης από έναν πίνακα, μπορούμε να χρησιμοποιήσουμε τον τελεστή `sizeof` (π.χ., `sizeof(arr)`)

Παρατηρήσεις (III)

Το μέγιστο μέγεθος μνήμης που μπορεί να δεσμευτεί με τη δήλωση ενός πίνακα εξαρτάται από το μέγεθος της στοίβας

- Π.χ. το επόμενο πρόγραμμα μπορεί να μην εκτελείται σε κάποιον υπολογιστή, εκτός κι αν η διαθέσιμη μνήμη του στη στοίβα είναι αρκετά μεγάλη, έτσι ώστε να μπορεί να αποθηκεύσει 500000 δεκαδικές τιμές τύπου double

```
#include <stdio.h>
int main(void)
{
        double arr[500000];
        return 0;
}
```

Όταν στο σύστημά σας η μνήμη είναι περιορισμένη, μη δηλώνετε πίνακες με μέγεθος μεγαλύτερο απ' ότι χρειάζεστε (να αποφεύγετε, δηλαδή, την κατασπατάληση μνήμης)

Παρατηρήσεις (IV)

- Στην περίπτωση που το μέγεθος του πίνακα πρέπει να είναι αρκετά μεγάλο, τότε προτείνεται η δήλωση του πίνακα με τη μορφή **δείκτη** και η δέσμευση μνήμης με χρήση της συνάρτησης `malloc()`

- Π.χ. η προηγούμενη δήλωση γράφεται ισοδύναμα:

```
double *arr;
arr = (double*)malloc(500000 * sizeof(double));
```

- Με την παραπάνω δήλωση, το τμήμα της μνήμης **δεν δεσμεύεται από τη στοίβα**, αλλά **από μία άλλη μεγαλύτερη περιοχή μνήμης** που ονομάζεται **σωρός (heap)**

- Τη σχέση μεταξύ πινάκων και δεικτών, καθώς και τη συνάρτηση `malloc()` θα τις δούμε σε επόμενες διαλέξεις

Παρατηρήσεις (V)

Να αποφεύγετε τη χρήση τελεστή αύξησης ή μείωσης στον δείκτη θέσης ενός πίνακα, π.χ. μη γράψετε ποτέ κάτι σαν αυτό:

```
b[i] = a[i++];
```

- Εξαρτάται απ' τον μεταγλωττιστή πότε θα γίνει η αύξηση του `i`

Επίσης, το λέμε και το ξαναλέμε, **μην ξεχνάτε** ότι η αρίθμηση των στοιχείων ενός πίνακα `n` στοιχείων ξεκινά απ' το `0` (όχι απ' το `1`) και φτάνει ως και το `n-1`

Για την αντιγραφή ενός πίνακα (π.χ. `int a[10]`) σε έναν άλλον (π.χ. `int b[10]`), **μην διανοηθείτε να γράψετε** κάτι σαν το παρακάτω:

```
b = a;
```

παρόλο που φαίνεται αρκετά «κομψό»

Απαγορεύεται (!!!!) και θα καταλάβετε γιατί, όταν εξηγήσουμε τη στενή σχέση μεταξύ πινάκων και δεικτών στο επόμενο κεφάλαιο

Παρατηρήσεις (VI)

Η C δεν ελέγχει αν κάνετε υπέρβαση των ορίων των θέσεων του πίνακα, αλλά αφήνει τον προγραμματιστή υπεύθυνο γι' αυτό... Σε περίπτωση, όμως, που γίνει υπέρβαση των ορίων του πίνακα, η συμπεριφορά του προγράμματος είναι απρόβλεπτη.

- Π.χ. δείτε το διπλανό πρόγραμμα

```
#include <stdio.h>
int main(void)
{
        int i, j = 20, arr[3];

        for(i = 0; i < 4; i++)
                arr[i] = 100;

        printf("%d\n", j);
        return 0;
}
```

- Ο `arr` περιέχει τρία στοιχεία και οι επιτρεπτές τιμές των δεικτών θέσης είναι από `0` έως `2`
- Στην τελευταία επανάληψη (`i=3`) η έκφραση `arr[3] = 100;` αναθέτει μία τιμή σε ένα στοιχείο που δεν ανήκει στον πίνακα και συγκεκριμένα την τιμή `100` που υπερεγγράφει (overwrites) το περιεχόμενο των 4 byte της μνήμης ακριβώς μετά το στοιχείο `arr[2]`
- Αν η συγκεκριμένη μνήμη έχει δεσμευτεί για τη μεταβλητή `j`, το `j` αλλάζει τιμή και το πρόγραμμα θα εμφανίσει `100` αντί για `20`...!!!

Παράδειγμα

- Γράψτε ένα πρόγραμμα το οποίο να δηλώνει έναν πίνακα 5 ακεραίων και να δίνει τις τιμές 10,20,30,40 και 50 στα στοιχεία του και στη συνέχεια να εμφανίζει στην οθόνη τα στοιχεία του πίνακα που έχουν τιμή μεγαλύτερη από 20.

```
#include <stdio.h>
int main(void)
{
	int i, arr[5];

	arr[0] = 10;
	arr[1] = 20;
	arr[2] = 30;
	arr[3] = 40;
	arr[4] = 50;
	for(i = 0; i < 5; i++)
	{ /* The braces are not necessary; we use them to make the code
clearer. */
		if(arr[i] > 20)
			printf("%d\n", arr[i]);
	}
	return 0;
}
```

Δήλωση Πίνακα και απόδοση αρχικών τιμών (Ι)

- Η C υποστηρίζει αρκετούς τρόπους απόδοσης αρχικών τιμών στα στοιχεία ενός πίνακα, ταυτόχρονα με τη δήλωση του πίνακα

1. Τη δήλωση του πίνακα την ακολουθεί ο τελεστής = και οι τιμές των στοιχείων διαχωρίζονται με κόμμα (,) μέσα σε άγκιστρα {}

```
int arr[4] = {10, 20, 30, 40};
```

Σε αυτό το παράδειγμα:

η τιμή του arr[0] γίνεται 10

η τιμή του arr[1] γίνεται 20

η τιμή του arr[2] γίνεται 30

και η τιμή του τελευταίου στοιχείου arr[3] γίνεται 40

Δήλωση Πίνακα και απόδοση αρχικών τιμών (ΙΙ)

2. Τη δήλωση του πίνακα την ακολουθεί ο τελεστής = και μέσα στα άγκιστρα {} δεν υπάρχουν τιμές για όλα τα στοιχεία του πίνακα

```
int arr[4] = {10, 20};
```

Σε αυτό το παράδειγμα:

η τιμή του arr[0] γίνεται 10

η τιμή του arr[1] γίνεται 20

η τιμή του arr[2] και του arr[3] γίνεται 0

(δηλ. για τα στοιχεία του πίνακα που δεν υπάρχει αντιστοίχηση "1-1", ο μεταγλωττιστής αποδίδει μηδενικές τιμές)

Δήλωση Πίνακα και απόδοση αρχικών τιμών (ΙΙΙ)

3. Σε περίπτωση που **δεν δηλωθεί** το μέγεθος του πίνακα, ο μεταγλωττιστής δημιουργεί έναν πίνακα που το μέγεθός του είναι ίσο με το πλήθος των τιμών στη λίστα (έμμεσος ορισμός μεγέθους του πίνακα)

```
int arr[] = {10, 20, 30, 40};
```

- Εδώ ο μεταγλωττιστής δημιουργεί έναν πίνακα ακεραίων με τόσες θέσεις όσες και οι αρχικές τιμές
- Επομένως, δημιουργεί έναν πίνακα 4 ακεραίων και αναθέτει στα στοιχεία του τις τιμές 10, 20, 30 και 40 αντίστοιχα, άρα:

η τιμή του arr[0] γίνεται 10
η τιμή του arr[1] γίνεται 20
η τιμή του arr[2] γίνεται 30
η τιμή του arr[3] γίνεται 40

Δήλωση Πίνακα και απόδοση αρχικών τιμών (IV)

4. Αν η δήλωση ενός πίνακα αρχίζει με τη λέξη `const`, τότε οι τιμές των στοιχείων του πίνακα **είναι σταθερές** και το πρόγραμμα δεν μπορεί να τις αλλάξει

```
const int b[] = {10, 20, 30, 40};
```

- Εδώ ο μεταγλωττιστής δεν επιτρέπει την αλλαγή τιμών στα στοιχεία του πίνακα, δηλ.

  η τιμή του `b[0]` γίνεται **μόνιμα** 10

  η τιμή του `b[1]` γίνεται **μόνιμα** 20

  η τιμή του `b[2]` γίνεται **μόνιμα** 30

  η τιμή του `b[3]` γίνεται **μόνιμα** 40

- Ο μεταγλωττιστής θα εμφανίσει μήνυμα λάθους σε οποιαδήποτε προσπάθεια αλλαγής της τιμής κάποιου στοιχείου, όπως π.χ. με την εντολή: `b[0] = 80;`

Παραδείγματα (Ι)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        int a[3] = {1,2,3};
        int b[4] = {40,50,60,70};

        b[a[2]] = 10;

        printf("%d %d %d\n",b[0],b[1],b[2]);
        return 0;
}
```

Έξοδος: 40 50 60

Παραδείγματα (ΙΙ)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        int i,arr[] = {10,20,30,40,50};

        for(i = 0; i < sizeof(arr)/sizeof(int); i++)
                printf("%d\n",arr[i]);
        return 0;
}
```

Έξοδος:

```
10
20
30
40
50
```

Παραδείγματα (III)

- Ποιες θα είναι οι τιμές των στοιχείων του πίνακα **a**, κατά την εκτέλεση του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        int i, a[3] = {4, 2, 0}, b[3] = {2, 3, 4};

        for(i = 0; i < 3; i++)
                a[b[i]-a[2-i]]++;
        return 0;
}
```

Τα **a[0]** , **a[1]**, και **a[2]** γίνονται **5**, **3**, και **1**, αντίστοιχα

Παραδείγματα (IV)

- Γράψτε ένα πρόγραμμα το οποίο να δηλώνει έναν πίνακα 5 ακεραίων και να δίνει τις τιμές 10,20,30,40,50 στα στοιχεία του. Στη συνέχεια, το πρόγραμμα να αντιγράφει τα περιεχόμενά του σε έναν δεύτερο πίνακα και να εμφανίζει τα στοιχεία του δεύτερου πίνακα στην οθόνη.

```
#include <stdio.h>
int main(void)
{
        int i,pin[5],arr[5] = {10,20,30,40,50};

        for(i = 0; i < 5; i++)
        {
                pin[i] = arr[i];
                printf("%d\n",pin[i]);
        }
        return 0;
}
```

Παραδείγματα (V)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        int i,pin[3],arr[3];

        pin[0] = 20;

        for(i = 0; i < 4; i++)
        {
                arr[i] = 100;
                printf("%d\n",arr[i]);
        }
        printf("%d\n",pin[0]);
        return 0;
}
```

Έξοδος:

```
100
100
100
100 (κακώς,υπερεγγραφή)
??? (ίσως 20, ίσως 100)
```

Παραδείγματα (VI)

- Γράψτε ένα πρόγραμμα το οποίο να διαβάζει 10 ακεραίους, να τους αποθηκεύει σε έναν πίνακα ακεραίων και στη συνέχεια να εμφανίζει τα στοιχεία του πίνακα με αντίστροφη σειρά.

```
#include <stdio.h>
int main(void)
{
	int i,num,arr[10];

	for(i = 0; i < 10; i++)
	{
		printf("Enter number: ");
		scanf("%d",&num);
		arr[i] = num;
	}

	printf("\nArray elements in reverse order:\n");
	for(i = 9; i >= 0; i--)
		printf("arr[%d] = %d\n",i,arr[i]);
	return 0;
}
```

Παραδείγματα (VII)

- Γράψτε ένα πρόγραμμα το οποίο να διαβάζει 10 ακεραίους αριθμούς, να τους αποθηκεύει σε έναν πίνακα ακεραίων και να τους εμφανίζει στην οθόνη. Το πρόγραμμα πριν αποθηκεύσει κάποιον αριθμό στον πίνακα πρέπει να ελέγχει αν αυτός ο αριθμός ήδη υπάρχει και μόνο αν δεν υπάρχει να τον αποθηκεύει στον πίνακα. Δηλαδή, όλα τα στοιχεία του πίνακα πρέπει να είναι διαφορετικά μεταξύ των.

```
#include <stdio.h>

#define SIZE 10

int main(void)
{
	int i,j,num,found,arr[SIZE];
	i = 0;
	while(i < SIZE)
	{
		printf("Enter number: ");
		scanf("%d",&num);

		found = 0;
		/* Η μεταβλητή i μετράει τους αριθμούς που έχουν καταχωρηθεί στον πίνακα, οπότε αυτός ο for βρόχος ελέγχει αν ο εισαγόμενος αριθμός υπάρχει ήδη στον πίνακα. Αν υπάρχει, η μεταβλητή found γίνεται 1 και ο for βρόχος τερματίζεται. */
```


(συνεχίζεται →)

Παραδείγματα (VII)

```
            for(j = 0; j < i; j++)
            {
                  if(arr[j] == num)
                  {
                        printf("Error: Number %d exists. ",num);
                        found = 1;
                        break; /* Τερματισμός του for βρόχου. */
                  }
            }
            /* Αν ο αριθμός δεν υπάρχει στον πίνακα, τότε τον
αποθηκεύουμε και αυξάνουμε τον δείκτη θέσης κατά ένα. */
            if(found == 0)
            {
                  arr[i] = num;
                  i++;
            }
      }/* Τέλος της while */
      printf("\nArray elements: ");
      for(i = 0; i < SIZE; i++)
            printf("%d ",arr[i]);

      printf("\n");
      return 0;
}
```

Παραδείγματα (VIII)

- Τι κάνει το παρακάτω πρόγραμμα ???

```
#include <stdio.h>
#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

int main(void)
{
    int answer;
    int rating;

    int frequency[ FREQUENCY_SIZE ] = { 0 };

    int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3,
8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

    for ( answer = 0; answer < RESPONSE_SIZE; answer++ )
        ++frequency[ responses [ answer ] ];


    printf( "%s%17s\n", "Rating", "Frequency" );

    for ( rating = 1; rating < FREQUENCY_SIZE; rating++ )
        printf( "%6d%17d\n", rating, frequency[ rating ] );

    return 0;

}
```

Παραδείγματα (VIII)

Έξοδος:

Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

Διδιάστατοι Πίνακες στη C

- Οι **διδιάστατοι πίνακες** μοιάζουν με τους γνωστούς μαθηματικούς πίνακες δύο διαστάσεων της άλγεβρας και αποτελούνται και αυτοί από **γραμμές** και **στήλες**

- Για να ορίσουμε έναν διδιάστατο πίνακα πρέπει να δηλώσουμε το όνομα του πίνακα, τον **τύπο δεδομένων** των στοιχείων του πίνακα, καθώς και το πλήθος των γραμμών και των στηλών του

- Η γενική περίπτωση ορισμού ενός διδιάστατου πίνακα είναι:

```
τύπος_δεδομένων  όνομα_πίνακα [πλήθος_γραμμών] [πλήθος_στηλών]
```

- Το **πλήθος των στοιχείων** ενός διδιάστατου πίνακα είναι ίσο με το **γινόμενο** του πλήθους **των γραμμών** του επί το πλήθος **των στηλών** του

Δήλωση διδιάστατου Πίνακα

- Π.χ. η εντολή `int array[10][5];` δηλώνει έναν διδιάστατο πίνακα με όνομα `array`, ο οποίος περιέχει 50 στοιχεία και καθένα από αυτά τα στοιχεία είναι ένας ακέραιος αριθμός (`int`)

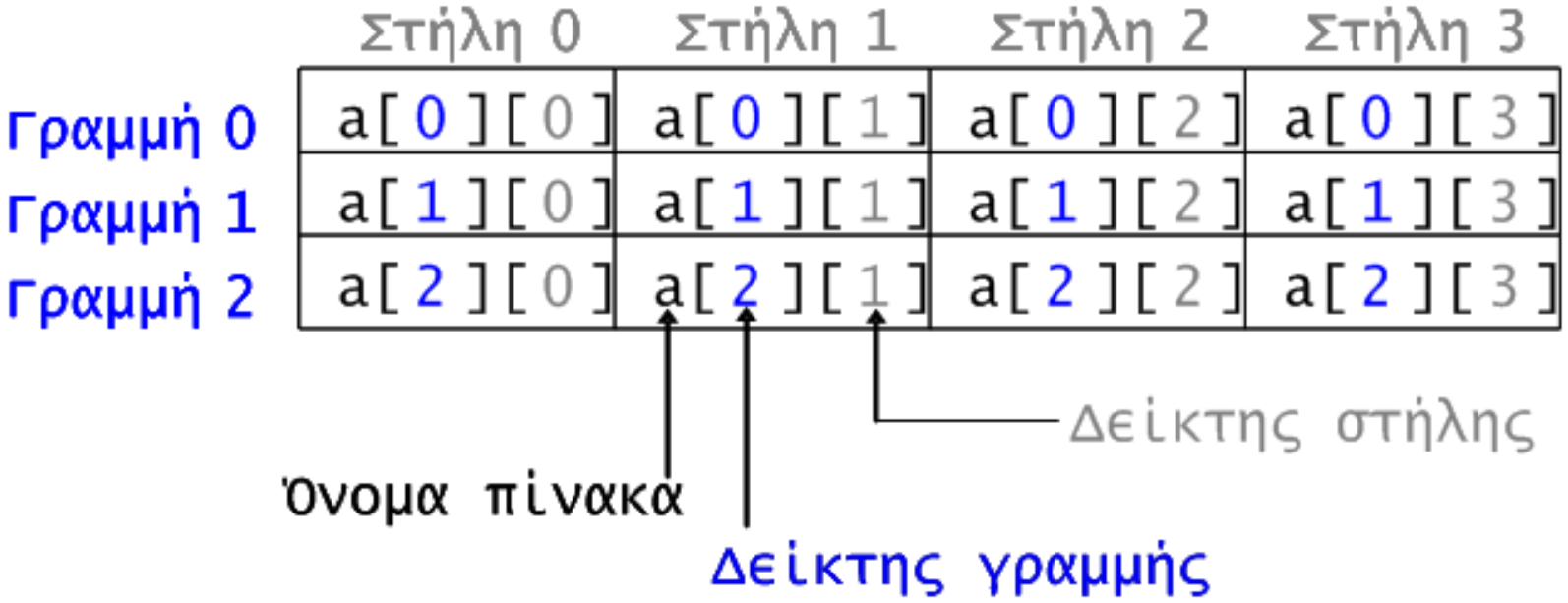
- Παρομοίως, η εντολή `char a[3][4];` δηλώνει έναν διδιάστατο πίνακα με όνομα `a`, ο οποίος περιέχει 12 στοιχεία και καθένα από αυτά τα στοιχεία είναι μία μεταβλητή τύπου `char`

- Για να αναφερθούμε σε κάποιο στοιχείο ενός διδιάστατου πίνακα γράφουμε το όνομα του πίνακα συνοδευόμενο από τον αριθμό γραμμής (ή αλλιώς τον δείκτη γραμμής) και τον αριθμό στήλης (ή αλλιώς τον δείκτη στήλης) του συγκεκριμένου στοιχείου του πίνακα

- Οι αριθμοί γραμμών και στηλών πρέπει να δηλώνονται μέσα σε αγκύλες `[][]`

Γραφική Αναπαράσταση

Π.χ. αν έχουμε δηλώσει τον πίνακα: `int a[3][4]`



Παρατηρήστε ότι – όπως και στους μονοδιάστατους πίνακες έτσι και στους διδιάστατους – η αρίθμηση για τις γραμμές και τις στήλες ξεκινάει από το 0

Διδιάστατοι Πίνακες και Μνήμη

- Κατά τη δήλωση του πίνακα, ο μεταγλωττιστής – όπως και στην περίπτωση των μονοδιάστατων πινάκων – δεσμεύει ένα **τμήμα μνήμης** από τη **στοίβα** (**stack**) για να αποθηκεύσει τα στοιχεία του

- Π.χ. με τη δήλωση `int array[10][5];` ο μεταγλωττιστής δεσμεύει 200 bytes για να αποθηκεύσει τα 50 στοιχεία του πίνακα (αφού κάθε στοιχείο του πίνακα – ως ακέραια μεταβλητή – απαιτεί 4 bytes).

- Τα στοιχεία του πίνακα αποθηκεύονται **σε διαδοχικές θέσεις στη μνήμη** ξεκινώντας από τα στοιχεία της 1ης γραμμής, συνεχίζοντας με τα στοιχεία της 2ης γραμμής, κ.ο.κ.

Διδιάστατοι Πίνακες και Μνήμη (1/2)

- Συγκεκριμένα, η θέση ενός τυχαίου στοιχείου `a[i][j]` του πίνακα `a[ROWS][COLS]` όπως αποθηκεύεται ο πίνακας στη μνήμη υπολογίζεται σύμφωνα με τον τύπο:

```
θέση_στη_μνήμη = (i * COLS) + j + 1
```

- Π.χ. η θέση του στοιχείου `a[2][1]` ενός πίνακα με 3 γραμμές και 4 στήλες (π.χ. `double a[3][4]`) είναι η:

```
θέση_στη_μνήμη = (i * COLS) + j + 1 = 2*4 + 1 + 1 = 10
```

- Δηλαδή, για τον πίνακα `double a[3][5]` , το στοιχείο `a[2][1]` **είναι το δέκατο κατά σειρά που αποθηκεύεται στη μνήμη**

- Παρατηρήστε, ότι για να υπολογιστεί η θέση του στοιχείου στη μνήμη δεν χρειάζεται να είναι γνωστή η πρώτη διάσταση του πίνακα (π.χ. `ROWS`), αλλά μόνο το πλήθος των στηλών του (π.χ. `COLS`)

Διδιάστατοι Πίνακες και Μνήμη (2/2)

- Θεωρήστε και πάλι έναν διδιάστατο πίνακα (έστω ακεραίων τύπου int) έστω `int x[ROWS][COLS]`

- Δεδομένου ότι τα στοιχεία του πίνακα αποθηκεύονται με τη σειρά στη μνήμη και δεδομένου επίσης ότι το όνομα του πίνακα (`x`) ισούται με τη διεύθυνση του πρώτου στοιχείου του πίνακα, ο μεταγλωττιστής για να βρει τη διεύθυνση μνήμης του στοιχείου `x[i][j]` 1) υπολογίζει το μέγεθος σε bytes της γραμμής, δηλ: `row_size = COLS × sizeof(int)` και στη συνέχεια πολλαπλασιάζει με `i` 2) πολλαπλασιάζει το `j` με το μέγεθος σε bytes του ενός στοιχείου και 3) προσθέτει τα δύο γινόμενα στη διεύθυνση του πρώτου στοιχείου του πίνακα, άρα:

```
memory_address = x + (i × row_size) + (j × sizeof(int))
```

Παρατηρήστε ότι για να βρεθεί η διεύθυνση στη μνήμη ενός στοιχείου διδιάστατου πίνακα, απαιτείται μόνο ο αριθμός των στηλών του πίνακα

Δήλωση Πίνακα και απόδοση αρχικών τιμών (Ι)

- Όπως και στην περίπτωση των μονοδιάστατων πινάκων, η C υποστηρίζει **παρόμοιους** τρόπους απόδοσης αρχικών τιμών στα στοιχεία ενός διδιάστατου πίνακα, ταυτόχρονα με τη δήλωση του πίνακα

- Σε όλες τις παρακάτω περιπτώσεις **οι αρχικές τιμές αποδίδονται** στα στοιχεία του πίνακα **ανά γραμμή**, ξεκινώντας από τα στοιχεία της πρώτης γραμμής, συνεχίζοντας στα στοιχεία της δεύτερης γραμμής, κ.ο.κ.

Δήλωση Πίνακα και απόδοση αρχικών τιμών (ΙΙ)

1. Τη δήλωση του πίνακα την ακολουθεί ο τελεστής = και οι τιμές των στοιχείων κάθε γραμμής περικλείονται ανάμεσα σε εσωτερικά άγκιστρα {} διαχωριζόμενες μεταξύ τους με κόμμα (,)

```
int arr[3][3] = {{10,20,30},
                 {40,50,60},
                 {70,80,90}};
```

Σε αυτό το παράδειγμα:

η τιμή του **arr[0][0]** γίνεται **10**

η τιμή του **arr[0][1]** γίνεται **20**

η τιμή του **arr[0][2]** γίνεται **30** κ.ο.κ.

Εναλλακτικά, μπορούμε να παραλείψουμε τα εσωτερικά άγκιστρα και να γράψουμε:

```
int arr[3][3] = {10,20,30,40,50,60,70,80,90};
```

Δήλωση Πίνακα και απόδοση αρχικών τιμών (ΙΙΙ)

2. Τη δήλωση του πίνακα την ακολουθεί ο τελεστής = και μέσα στα εσωτερικά άγκιστρα {} δεν υπάρχουν τιμές για όλα τα στοιχεία της κάθε γραμμής του πίνακα

   Σε αυτή την περίπτωση ο μεταγλωττιστής αποδίδει την τιμή 0 στα υπόλοιπα στοιχεία της κάθε γραμμής του πίνακα

```
int arr[3][3] = {{10,20},
                 {40,50},
                 {70}};
```

Π.χ. στο παραπάνω παράδειγμα:

η τιμή του arr[0][0] γίνεται 10
η τιμή του arr[0][1] γίνεται 20
η τιμή του arr[1][0] γίνεται 40
η τιμή του arr[1][1] γίνεται 50
η τιμή του arr[2][0] γίνεται 70

ενώ οι τιμές των arr[0][2] , arr[1][2] , arr[2][1] και arr[2][2] γίνονται 0

Δήλωση Πίνακα και απόδοση αρχικών τιμών (IV)

3. Αν έχουμε παραλείψει τιμές για όλα τα στοιχεία κάποιας γραμμής, τότε ο μεταγλωττιστής αποδίδει την τιμή 0 σε όλα τα στοιχεία της γραμμής

```
int arr[3][3] = {{10,20,30}};
```

Π.χ. στο παραπάνω παράδειγμα:

οι τιμές όλων των στοιχείων της 2ης και της 3ης γραμμής γίνονται 0

4. Αν τέλος έχουμε παραλείψει τα εσωτερικά άγκιστρα { } και δεν αποδίδουμε τιμές για όλα τα στοιχεία του πίνακα, τότε ο μεταγλωττιστής αποδίδει την τιμή 0 στα υπόλοιπα στοιχεία του πίνακα

```
int arr[3][3] = {10,20};
```

Π.χ. στο παραπάνω παράδειγμα:

οι τιμές των `arr[0][0]` και `arr[0][1]` γίνονται 10 και 20 αντίστοιχα, ενώ όλων των υπολοίπων στοιχείων γίνονται 0

Δήλωση Πίνακα και απόδοση αρχικών τιμών (V)

5. Το πλήθος των γραμμών **δεν είναι υποχρεωτικό** να δηλωθεί. Πρέπει όμως υποχρεωτικά να δηλωθεί ο αριθμός των στηλών

```
int arr[][3] = {10,20,30,40,50,60};
```

Π.χ. στο παραπάνω παράδειγμα:

ο μεταγλωττιστής θα δημιουργήσει **αυτόματα** έναν **διδιάστατο** πίνακα ακεραίων με **δύο γραμμές** και **τρεις στήλες**, διότι οι αρχικές τιμές που αποδίδονται στα στοιχεία του πίνακα (**έξι τιμές συνολικά**) απαιτούν πίνακα με τουλάχιστον δύο γραμμές για τις τρεις ήδη δηλωμένες στήλες

Συγκεκριμένα, η τιμή:

του `arr[0][0]` γίνεται 10

του `arr[0][1]` γίνεται 20

του `arr[0][2]` γίνεται 30

του `arr[1][0]` γίνεται 40 κ.ο.κ.

Δήλωση Πίνακα και απόδοση αρχικών τιμών (VI)

6. Οι τρόποι αρχικοποίησης ενός διδιάστατου πίνακα που παρουσιάστηκαν χρησιμοποιούνται όταν θέλουμε να δώσουμε **συγκεκριμένες τιμές** στα στοιχεία ενός πίνακα.

   Για πίνακες **μεγαλύτερου μεγέθους** και όταν οι αρχικές τιμές του **δεν απαιτείται να είναι συγκεκριμένες** (π.χ. είναι είτε ίδιες για όλα τα στοιχεία του πίνακα είτε μπορούν να παραχθούν εύκολα διότι σχετίζονται μεταξύ τους) συνήθως χρησιμοποιούνται **διπλοί επαναληπτικοί βρόχοι**

```
#include <stdio.h>
int main(void)
{
        int i, j, arr[50][100];

        for(i = 0; i < 50; i++)
                for(j = 0; j < 100; j++)
                        arr[i][j] = 1;
        return 0;
}
```

Παραδείγματα (Ι)

- Γράψτε ένα πρόγραμμα το οποίο να αρχικοποιεί έναν διδιάστατο πίνακα 5×5 ως τον μοναδιαίο τετραγωνικό 5×5 πίνακα και να εμφανίζει τα στοιχεία του πίνακα στην οθόνη υπό τη μορφή πίνακα 5×5 της άλγεβρας

```
#include <stdio.h>

#define SIZE 5

int main(void)
{
	int i,j,arr[SIZE][SIZE] = {0}; /* Initialize the arr
elements with 0. */

	for(i = 0; i < SIZE; i++)
	{
		for(j = 0; j < SIZE; j++)
		{
			if(i == j) /* The row and column indexes of
the elements of the main diagonal are equal. */
				arr[i][j] = 1;

			printf("%3d",arr[i][j]);
		}
		printf("\n");/* Add it to separate the array rows.*/
	}
	return 0;
}
```

Παραδείγματα (ΙΙ)

- Γράψτε ένα πρόγραμμα το οποίο να διαβάζει ακέραιους αριθμούς, να τους αποθηκεύει σε έναν 2×4 πίνακα και να εμφανίζει τη μικρότερη και τη μεγαλύτερη τιμή κάθε γραμμής του πίνακα

```
#include <stdio.h>

#define ROWS 2
#define COLS 4

int main(void)
{
	int i,j,max,min,arr[ROWS][COLS];

	for(i = 0; i < ROWS; i++)
	{
		for(j = 0; j < COLS; j++)
		{
			printf("Enter the element arr[%d][%d]: ",i,j);
			scanf("%d",&arr[i][j]);
		}
	}
	for(i = 0; i < ROWS; i++)
	{
		/* Initialize the min and max with the first element
of each row. */
		min = max = arr[i][0];
		for(j = 0; j < COLS; j++)
		{
			if(arr[i][j] >= max)
				max = arr[i][j];

			if(arr[i][j] <= min)
				min = arr[i][j];
		}
		printf("Row_%d: Max = %d Min = %d\n",i+1,max,min);
	}
	return 0;
}
```