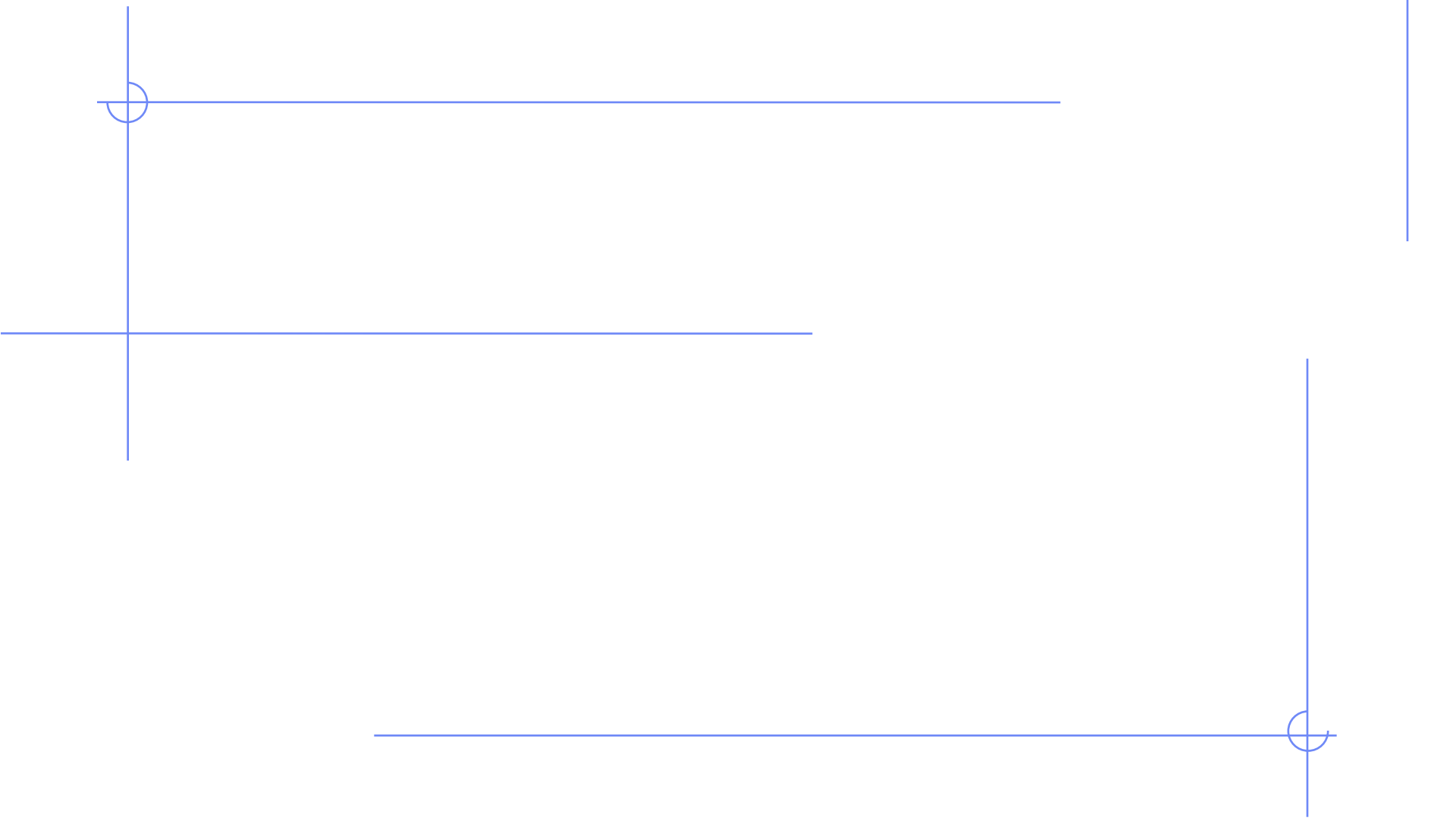
«Προγραμματισμός Ι»

5 - *Πίνακες*

Δρ. Χαράλαμπος Καρανίκας
Τμήμα Μηχανικών Πληροφορικής



Οι διαφάνειες βασίζονται στο βιβλίο: C: Από τη Θεωρία στην Εφαρμογή 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** 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 array[10];
//πίνακας με όνομα array, ο οποίος περιέχει 10 ακέραιους

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

char pin[20];
//πίνακας με όνομα pin, ο οποίος περιέχει 20 χαρακτήρες
```

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

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

Στοιχεία μονοδιάστατου Πίνακα

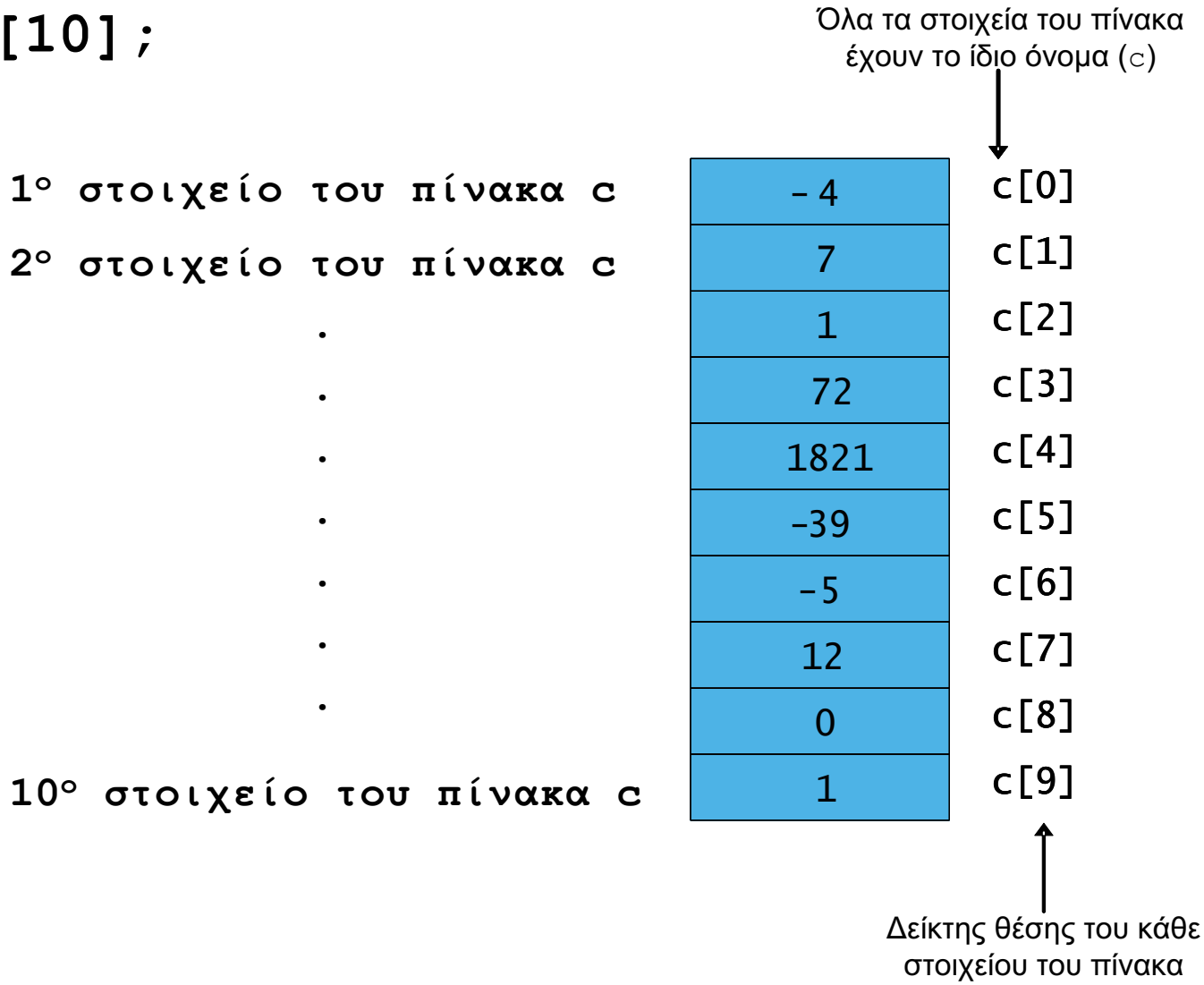
- Για να αναφερθούμε σε κάποιο στοιχείο του πίνακα γράφουμε το **όνομα του πίνακα** συνοδευόμενο από τον **δείκτη θέσης** του στοιχείου μέσα σε αγκύλες `[]`

- *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

Π.χ. `int c[10];`



Παράδειγμα

```
int arr[10]; /* Δήλωση πίνακα με στοιχεία 10 ακεραίους. */
int i,j;

i = j = 2;
arr[0] = 100; /* Η τιμή του 1ου στοιχείου του πίνακα
γίνεται 100. */
arr[2+3] = 200; /* Η τιμή του 6ου στοιχείου του πίνακα
γίνεται 200. */
arr[9] = arr[0]; /* Η τιμή του τελευταίου στοιχείου του
πίνακα, το οποίο είναι αποθηκευμένο στην 9η θέση του
πίνακα, γίνεται ίση με την τιμή του 1ου στοιχείου, δηλαδή
100. */
arr[i+j] = 300; /* Η τιμή του i+j στοιχείου του πίνακα,
δηλαδή του 5ου στοιχείου αφού i+j= 2+2 = 4 γίνεται 300 */
```

Παρατηρήσεις (I)

- Για τη δήλωση του μεγέθους ενός πίνακα χρησιμοποιείται πολύ συχνά η οδηγία **#define**

- Π.χ.

```
#define SIZE 10
int arr[SIZE]; /* Ο μεταγλωττιστής αντικαθιστά τη
λέξη SIZE με την τιμή 10 και δημιουργεί έναν πίνακα
με στοιχεία 10 ακεραίους. */
```

Σημειώστε επίσης ότι το πλήθος των στοιχείων του πίνακα **δεν μπορεί να αλλάξει** κατά την εκτέλεση του προγράμματος

Παρατηρήσεις (ΙΙ)

- Το μέγιστο μέγεθος μνήμης που μπορεί να δεσμευτεί με τη δήλωση ενός πίνακα εξαρτάται από τον τρόπο κατανομής μνήμης που χρησιμοποιεί το λειτουργικό σύστημα για να καθορίσει το μέγεθος της στοίβας

- Π.χ. το επόμενο πρόγραμμα μπορεί να μην εκτελείται σε κάποιον υπολογιστή γιατί δεν υπάρχει διαθέσιμη μνήμη στη στοίβα για την αποθήκευση 300000 ακεραίων

```
#include <stdio.h>
int main()
{
        int arr[300000];
        return 0;
}
```

- Ωστόσο, μπορεί να εκτελείται σε κάποιον άλλο υπολογιστή με διαφορετικό λειτουργικό σύστημα που παρέχει μεγαλύτερο μέγεθος στοίβας

Παρατηρήσεις (III)

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

- Π.χ. η προηγούμενη δήλωση γράφεται ισοδύναμα:

```
int *arr;
arr = (int*)malloc(300000 * sizeof(int));
```

- Τώρα, 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

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

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

  η τιμή του `arr[0]` γίνεται μόνιμα 10

  η τιμή του `arr[1]` γίνεται μόνιμα 20

  η τιμή του `arr[2]` γίνεται μόνιμα 30

  η τιμή του `arr[3]` γίνεται μόνιμα 40

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

Παρατηρήσεις (I)

- Το **πρώτο** στοιχείο ενός πίνακα μεγέθους `n` στοιχείων αποθηκεύεται στη **θέση μηδέν** (`0`) και όχι στη θέση ένα (`1`), ενώ το **τελευταίο** στοιχείο στη **θέση** `n-1` και όχι στη θέση `n`

- Όταν δηλώνεται ένας πίνακας και τα στοιχεία του **δεν αρχικοποιούνται**, τότε ο μεταγλωττιστής θέτει **τυχαίες τιμές** στα στοιχεία του

  Π.χ. με τη δήλωση: `int arr[4];`

  ο μεταγλωττιστής θέτει τυχαίες τιμές στα στοιχεία του πίνακα `arr`

Παρατηρήσεις (II)

Μεγάλη προσοχή στην υπέρβαση των ορίων του πίνακα...

Π.χ. με τη δήλωση: `int arr[4];`
οι επιτρεπτές τιμές του δείκτη θέσης είναι από 0 έως και 3

Αν όμως στη συνέχεια του προγράμματος γράψετε: `arr[7] = 20;`
ο μεταγλωττιστής δεν εντοπίζει το λάθος και επιτρέπει την εκτέλεση του προγράμματος

Αυτή η υπέρβαση των επιτρεπτών ορίων μπορεί να επιδράσει αρνητικά στη λειτουργία άλλων τμημάτων του προγράμματος και να οδηγήσει σε απρόβλεπτα αποτελέσματα

Η υπέρβαση των επιτρεπτών ορίων ενός πίνακα είναι ένα από τα συνηθισμένα λάθη που εισάγει ένας προγραμματιστής και σε ένα μεγάλο πρόγραμμα είναι δύσκολο να εντοπιστούν

Παρατηρήσεις (III)

- Όχι στη δήλωση πινάκα με μεγαλύτερο μέγεθος από ότι χρειάζεται, ώστε να αποφεύγεται η αχρείαστη σπατάλη μνήμης

- Σε συγκεκριμένες περιπτώσεις, προτείνεται - για λόγους ευελιξίας - η χρήση της οδηγίας `#define` για τη δήλωση του μεγέθους ενός πίνακα, έτσι ώστε, σε περίπτωση μελλοντικής αλλαγής του μεγέθους του, να μην απαιτούνται αλλαγές στο υπόλοιπο πρόγραμμα

Παραδείγματα (Ι)

- Γράψτε ένα πρόγραμμα το οποίο να δηλώνει έναν πίνακα 5 ακεραίων και να δίνει τις τιμές 100,101,102,103,104 στα στοιχεία του.

  Στη συνέχεια, το πρόγραμμα να εμφανίζει τα στοιχεία του πίνακα στην οθόνη

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

        arr[0] = 100;
        arr[1] = 101;
        arr[2] = 102;
        arr[3] = 103;
        arr[4] = 104;

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

Παραδείγματα (ΙΙ)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
        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 (ΙΙΙ)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
        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 (IV)

- Ποια είναι τα περιεχόμενα του πίνακα **arr** στο παρακάτω πρόγραμμα ???

```
#include <stdio.h>
int main()
{
        int i,arr[10] = {0};
        for(i = 0; i < 10; i++)
                arr[i++] = 20;
        return 0;
}
```

Η τιμή του κάθε «άρτιου» στοιχείου:

arr[0], arr[2], arr[4], arr[6], arr[8] γίνεται **20**

Η τιμή του κάθε «περιττού» στοιχείου:

arr[1], arr[3], arr[5], arr[7], arr[9] γίνεται **0**

Παραδείγματα (V)

- Ποια είναι τα περιεχόμενα του πίνακα **arr** στο παρακάτω πρόγραμμα ???

```
#include <stdio.h>
int main()
{
        int i,arr[10] = {0};
        for(i = 0; i < 10; i++)
                arr[++i] = 20;
        return 0;
}
```

Η τιμή του κάθε «άρτιου» στοιχείου:

arr[0], arr[2], arr[4], arr[6], arr[8] γίνεται **0**

Η τιμή του κάθε «περιττού» στοιχείου:

arr[1], arr[3], arr[5], arr[7], arr[9] γίνεται **20**

Παραδείγματα (VI)

- 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()
{
        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 (VII)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
        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 (VIII)

- 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()
{
	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 (XI)

- 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()
{
	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 (XI)

```
            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 (Χ)

- Τι κάνει το παρακάτω πρόγραμμα ???

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

int main()
{
    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 (Χ)

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**)

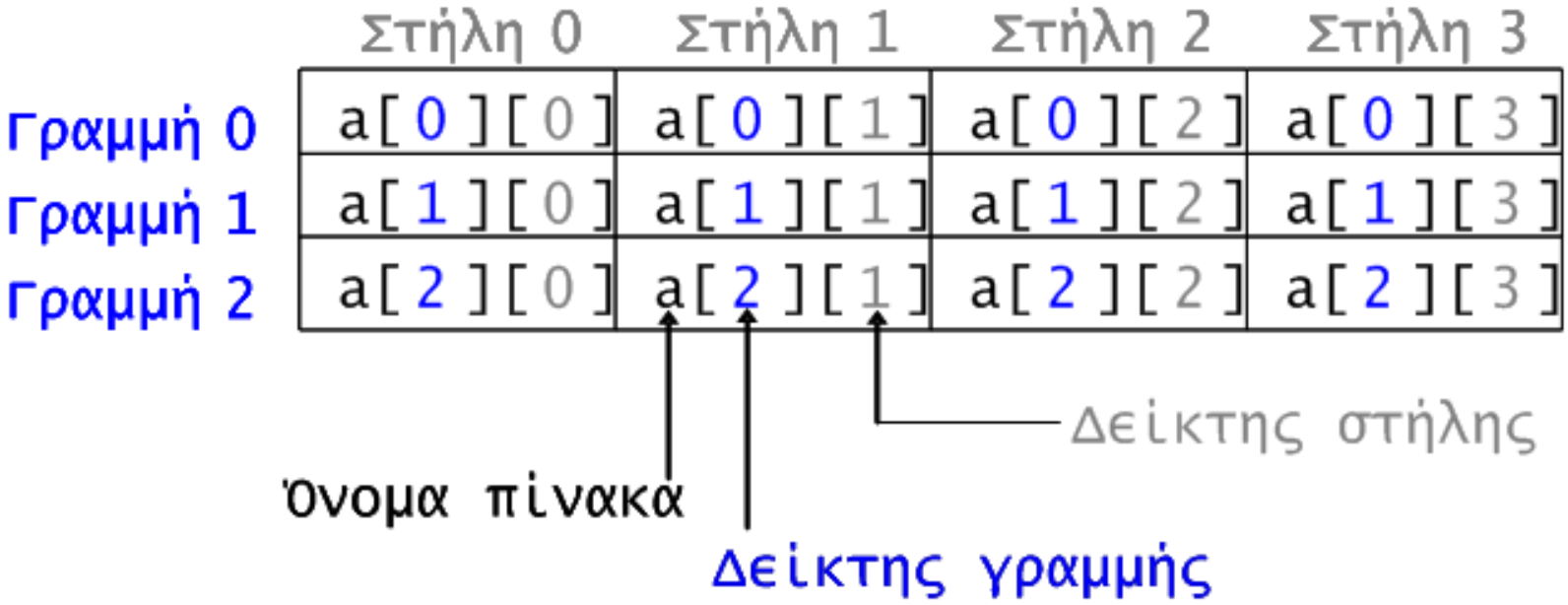
- Παρομοίως, η εντολή `char array[3][4];` δηλώνει έναν διδιάστατο πίνακα με όνομα **array**, ο οποίος περιέχει 12 στοιχεία και καθένα από αυτά τα στοιχεία είναι ένας χαρακτήρας (**char**)

- 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

- Συγκεκριμένα, η θέση μνήμης ενός τυχαίου στοιχείου `a[i][j]` του πίνακα `a[ROWS][COLS]` υπολογίζεται σύμφωνα με τον τύπο:

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

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

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

- Παρατηρήστε, ότι για να υπολογιστεί η θέση μνήμης 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`)

Παράδειγμα

- Ο πιο απλός τρόπος αρχικοποίησης ενός διδιάστατου πίνακα είναι να αποδώσουμε αρχικές τιμές σε κάθε ένα στοιχείο του, αφού προηγουμένως έχουμε ορίσει τον πίνακα

```
int  i,j,arr[3][4];  /* Δήλωση διδιάστατου πίνακα με
στοιχεία 12 ακεραίους. */
i = j = 2;

arr[0][0] = 100; /* Η τιμή του 1ου στοιχείου του πίνακα
(1ο στοιχείο της 1ης γραμμής) γίνεται 100. */
arr[1][1] = 200; /* Η τιμή του 6ου στοιχείου (2ο στοιχείο
της 2ης γραμμής) του πίνακα γίνεται 200. */
arr[2][3]  =  array[0][0];  /*  Η  τιμή  του  τελευταίου
στοιχείου του πίνακα (4ο στοιχείο της 3ης γραμμής) γίνεται
ίση με την τιμή του 1ου στοιχείου. */
arr[i][j] = 300; /* Η τιμή του προτελευταίου στοιχείου του
πίνακα (3ο στοιχείο της 3ης γραμμής) γίνεται 300. */
arr[i-2][j-2]  =  300;  /*  Η  τιμή  του  1ου  στοιχείου  του
πίνακα (1ο στοιχείο της 1ης γραμμής) γίνεται 300. */
```

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:

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 και τρεις στήλες

Συγκεκριμένα, η τιμή:

του `arr[0][0]` γίνεται 10

του `arr[0][1]` γίνεται 20

του `arr[0][2]` γίνεται 30 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, συνήθως χρησιμοποιούνται **διπλοί επαναληπτικοί βρόχοι**

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

	for(i = 0; i < 50; i++)
	{
		for(j = 0; j < 100; j++)
		{
			arr[i][j] = 300;
			printf("arr[%d][%d] = %d\n",i,j,arr[i][j]);
		}
	}
	return 0;
}
```

Παραδείγματα (I)

- 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()
{
      int  i,j,arr[SIZE][SIZE]  =  {0};  /*  Αρχικοποίηση  των
στοιχείων του πίνακα με την τιμή 0. */

      for(i = 0; i < SIZE; i++)
      {
            for(j = 0; j < SIZE; j++)
            {
                  if(i  ==  j)  /*  Για  τα  στοιχεία  της  κυρίας
διαγωνίου ισχύει i=j */
                        arr[i][j] = 1;

                  printf("%3d",arr[i][j]);
            }
            printf("\n"); /* Χρησιμοποιείται για τη δημιουργία
των διαφορετικών γραμμών του πίνακα. */
      }
      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()
{
	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++)
	{
		/* Αρχικοποίηση των min και max με το πρώτο στοιχείο
της κάθε γραμμής. */
		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;
}
```