

## 5. Πίνακες – αλφαριθμητικά

### Σύνοψη

Στο κεφάλαιο αυτό ο αναγνώστης εισάγεται στις έννοιες του πίνακα και του πίνακα χαρακτήρων (αλφαριθμητικού). Μελετώνται τόσο οι μονοδιάστατοι όσο και οι πολυδιάστατοι πίνακες σε ό,τι αφορά τα χαρακτηριστικά τους και τις λειτουργίες τους (δήλωση, ορισμός, ανάγνωση και εκτύπωση στοιχείων πίνακα, ανάθεση τιμής). Περιγράφεται σε αδρές γραμμές (χωρίς δείκτες) η χρήση πινάκων ως όρισμα συναρτήσεων. Αναλύονται τα αλφαριθμητικά και οι πίνακες αλφαριθμητικών και παρουσιάζονται οι βασικές συναρτήσεις διαχείρισης αλφαριθμητικών, καθώς και οι συναρτήσεις μετατροπής αλφαριθμητικών σε αριθμητικές τιμές.

### Λέξεις κλειδιά

μονοδιάστατοι πίνακες, πολυδιάστατοι πίνακες, αλφαριθμητικά, πίνακες αλφαριθμητικών, αρχικοποίηση πινάκων, gets, puts, strlen, strcpy, strcat, strcmp, atoi, atol, atof, πίνακες μεταβλητού μήκους, καθοριστές.

### Προαπαιτούμενη γνώση

Λεξιλόγιο της γλώσσας C – μεταβλητές – εκφράσεις – τελεστές – έλεγχος ροής προγράμματος – συναρτήσεις

### 5.1. Μονοδιάστατοι πίνακες

Ο πίνακας είναι μία συλλογή δεδομένων του ίδιου τύπου, τα οποία είναι αποθηκευμένα σε διαδοχικές θέσεις μνήμης.

Η δήλωση του πίνακα ακολουθεί τον εξής φορμαλισμό:

**τύπος\_δεδομένου όνομα\_πίνακα[μέγεθος];**

Διακρίνονται τρία τμήματα: α) ο τύπος δεδομένων (**float**, **int**, **char**, **double**), β) το όνομα του πίνακα και γ) ο αριθμός των στοιχείων του πίνακα. Έτσι, μία τυπική δήλωση ενός πίνακα 100 στοιχείων κινητής υποδιαστολής απλής ακρίβειας είναι η ακόλουθη:

**float array[100];**

Η αναφορά σε στοιχείο πίνακα γίνεται με συνδυασμό του ονόματος και ενός δείκτη (ή αλλιώς αριθμοδείκτη, index), ο οποίος εκφράζει τη σειρά του στοιχείου μέσα στον πίνακα:

**array[0]:** *πρώτο* στοιχείο του πίνακα

**array[1]:** *δεύτερο* στοιχείο του πίνακα

**array[99]:** *τελευταίο (εκατοστό)* στοιχείο του πίνακα

Η απόδοση αρχικής τιμής κατά τη δήλωση του πίνακα γίνεται με χρήση του τελεστή ανάθεσης ως εξής:

**float array[5]={1.0,2.22,-4.2,6.7,338};** αρχικοποιούνται και τα 5 στοιχεία του πίνακα **array**.

**float array[5]={1.0,2.22,-4.2};** αρχικοποιούνται τα 3 πρώτα στοιχεία του πίνακα **array**, δηλαδή τα **array[0]**, **array[1]**, **array[2]**.

Στο πρότυπο της γλώσσας C99 έχουν εισαχθεί οι **καθοριστές** (designators), οι οποίοι επιτρέπουν τη στοχευμένη αρχικοποίηση στοιχείων σε έναν πίνακα. Είναι ακέραιοι αριθμοί και λειτουργούν ως αριθμοδείκτες, καθώς γράφονται μέσα σε αγκύλες και καθορίζουν τη θέση στον πίνακα στην οποία θα τοποθετηθεί η αρχική τιμή. Η πρόταση

```
float array[5]={ [2]=1.7, [4]=-19.3};
```

δημιουργεί τον πίνακα **array** και αρχικοποιεί το τρίτο και το πέμπτο στοιχείο του πίνακα με τις τιμές **1.7** και **-19.3**, αντίστοιχα.

Ένα επιπρόσθετο πλεονέκτημα των καθοριστών είναι ότι μπορούν να τοποθετηθούν με οποιαδήποτε σειρά κι όχι αποκλειστικά με αύξουσα, όπως φαίνεται παρακάτω:

```
float array[9]={ [2]=1.7, [7]=-19.3, [5]=13};
```

Εάν το μέγεθος ενός πίνακα έχει οριστεί με τη δήλωση, π.χ. **array[9]**, ο καθοριστές μπορούν να λάβουν τιμή από 0 έως και 8. Εάν όμως δεν οριστεί το μέγεθος του πίνακα, καθορίζεται έμμεσα από τη μέγιστη τιμή καθοριστή, ο οποίος δεν πρέπει να είναι αρνητικός ακέραιος. Στην ακόλουθη πρόταση

```
float array[]={ [2]=1.7, [15]=-19.3, [5]=13};
```

το μέγεθος του πίνακα καθορίζεται στο **16 (15+1)**.

Η ανάγνωση και εκτύπωση ενός πίνακα γίνονται κατά στοιχείο, με τους κανόνες που ισχύουν για κάθε τύπο δεδομένου:

```
for ( i=0;i<arraySize;i++ ) {  
    scanf( "%f",&array[i] );  
    printf( "array[%d]=%f",i,array[i] );  
}
```

#### Παρατηρήσεις:

1. Όταν αποδίδονται αρχικές τιμές μπορεί να παραληφθεί το μέγεθος του πίνακα. Το μέγεθος του πίνακα καθορίζεται από τον αριθμό των αρχικών τιμών που δίδονται. Η παρακάτω δήλωση

```
char array[ ] = {'D','k','$','q'};
```

έχει ως αποτέλεσμα τη δημιουργία ενός πίνακα χαρακτήρων (**char**) τεσσάρων στοιχείων με αρχικές τιμές:

```
array[0]='D'  
array[1]='k'  
array[2]='$'  
array[3]='q'
```

2. Το γεγονός ότι οι δείκτες των στοιχείων ενός πίνακα ξεκινούν από το **0** κι όχι από το **1** μπορεί να αποτελεί μία ιδιόζουσα ιδιότητα της γλώσσας C αλλά πηγάζει από τη φιλοσοφία της να αποτελεί μεν γλώσσα προγραμματισμού υψηλού επιπέδου αλλά ταυτόχρονα ο προγραμματισμός σε C να βρίσκεται κοντά στην αρχιτεκτονική του υπολογιστή. Εφόσον το **0** αποτελεί το σημείο εκκίνησης για τους υπολογιστές, εάν η αρίθμηση των στοιχείων πίνακα ξεκινούσε από το **1**, ο μεταγλωττιστής θα έπρεπε να αφαιρέσει τη μονάδα από κάθε αναφορά σε δείκτη στοιχείου (οι δείκτες θα παρουσιαστούν στο επόμενο κεφάλαιο), για να ληφθεί η διεύθυνση ενός στοιχείου.

3. Υπάρχει διαφορά ανάμεσα στη δήλωση πίνακα και στην αναφορά στοιχείου πίνακα. Σε μία δήλωση, ο αριθμός μέσα στις αγκύλες καθορίζει το μέγεθος του πίνακα. Σε μία αναφορά στοιχείου πίνακα, ο αριθμοδείκτης προσδιορίζει το στοιχείο του πίνακα, στο οποίο αναφερόμαστε. Π.χ. στη δήλωση **int array[100]**; το **100** δηλώνει τον αριθμό των στοιχείων του πίνακα. Αντίθετα, στη **array[7]=167**; το **7** δηλώνει το 8<sup>ο</sup> στοιχείο του πίνακα, στο οποίο αποδίδεται η τιμή **167**.

4. Μπορεί να βρεθεί το μέγεθος σε bytes ενός πίνακα χρησιμοποιώντας τον τελεστή **sizeof**. Για παράδειγμα, εάν θεωρηθεί ο πίνακας **int array[100]**; η έκφραση **sizeof(array)** δίνει τιμή **400**, επειδή ο πίνακας αποτελείται από 100 ακεραίους των 4 bytes.

Στον **sizeof** θα πρέπει να περιλαμβάνεται μόνο το όνομα του πίνακα. Αν περιληφθεί δείκτης ενός στοιχείου, τότε θα εξαχθεί το μέγεθος του στοιχείου. Για παράδειγμα, η έκφραση **sizeof(array[0])** δίνει τιμή **4**.

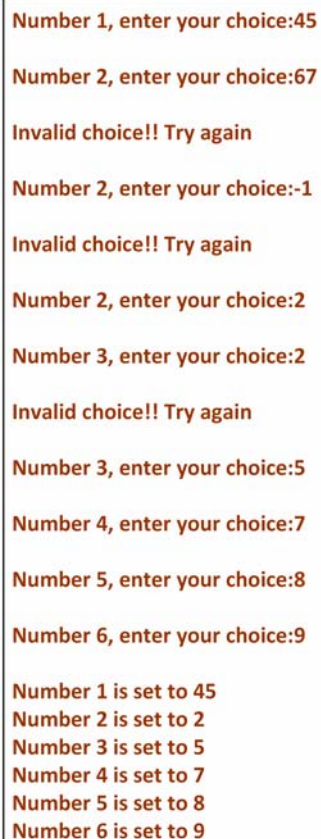
Χρησιμοποιώντας έναν συνδυασμό των παραπάνω μπορεί να βρεθεί ο αριθμός των στοιχείων του πίνακα. Η έκφραση **sizeof(array)/sizeof(array[0])** δίνει **100**, τον αριθμό δηλαδή των στοιχείων του πίνακα **array**.

### 5.1.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα δέχεται από το πληκτρολόγιο διαδοχικά 6 ακέραιους αριθμούς του ΛΟΤΤΟ. Θα πρέπει να λαμβάνεται πρόνοια, ώστε, όταν ένας αριθμός που δίνεται από το πληκτρολόγιο είτε δεν ανήκει στο [1, 49] είτε έχει δοθεί προηγουμένως, να ζητείται νέα τιμή γι' αυτόν. Οι αριθμοί θα αποθηκεύονται σε πίνακα ακεραίων και θα εμφανίζονται στην οθόνη μετά το πέρας της εισαγωγής τους.

```
#include <stdio.h>
#define N 6

int main() {
    int lotto[6],j,deikt,i=0;
    while (i<N) {
        deikt=0;
        printf( "\nNumber %d, enter your choice:",i+1 );
        scanf( "%d",&lotto[i] );
        if ((lotto[i]<1) || (lotto[i]>49)) deikt++; /* [1,49] */
        for (j=0;j<i;j++) /* Να μην επαναληφθεί προηγούμενος αριθμός */
            if (lotto[j]==lotto[i]) deikt++;
        if (deikt) printf( "\nInvalid choice!! Try again\n" );
        else i++;
    } /* τέλος της while */
    for (i=0;i<N;i++) printf( "\nNumber %d is set to %d",i+1,lotto[i] );
    return 0;
}
```



Number 1, enter your choice:45  
Number 2, enter your choice:67  
Invalid choice!! Try again  
Number 2, enter your choice:-1  
Invalid choice!! Try again  
Number 2, enter your choice:2  
Number 3, enter your choice:2  
Invalid choice!! Try again  
Number 3, enter your choice:5  
Number 4, enter your choice:7  
Number 5, enter your choice:8  
Number 6, enter your choice:9  
  
Number 1 is set to 45  
Number 2 is set to 2  
Number 3 is set to 5  
Number 4 is set to 7  
Number 5 is set to 8  
Number 6 is set to 9

Εικόνα 5.1 Η έξοδος του προγράμματος του παραδείγματος 5.1.1

### 5.1.2. Παράδειγμα

Να γραφεί πρόγραμμα, με το οποίο θα δίνονται από το πληκτρολόγιο 6 πραγματικοί αριθμοί, θα αποθηκεύονται στον πίνακα **array[]** και θα τυπώνονται: (α) οι θετικοί εξ αυτών, (β) ο μεγαλύτερος και (γ) ο αριθμός των στοιχείων του **array[]**, τα οποία έχουν τιμές στο διάστημα [1.05, 50.8].

```
#include <stdio.h>
#include <math.h> /* για τη συνάρτηση fabs() */

#define N 6
#define lower 1.05
#define upper 50.8

int main()
{
    float arrayay[N],maxim;
    int i,count=0;
    for (i=0;i<N;i++)
    {
        switch(i) /* για να εμφανιστεί σωστά η αρίθμηση */
        {
            case 0:
                printf( "\nGive 1st number: " );
                break;
            case 1:
                printf( "\nGive 2nd number: " );
                break;
            case 2:
                printf( "\nGive 3rd number: " );
                break;
            default:
                printf( "\nGive %dth number: ",i+1 );
                break;
        }
        scanf( "%f",&arrayay[i] );
    }
    maxim=arrayay[0];
    printf( "\n" );
    for (i=0;i<N;i++) /* i=0 για να μουν όλοι οι έλεγχοι που
απαιτούν τα σκέλη (α), (β), (γ) σε ένα βρόχο. Εάν ο βρόχος αφορούσε μόνο
την εξαγωγή του μεγίστου, ο μετρητής θα ξεκινούσε από το 1 */
    {
        if (arrayay[i]==fabs(arrayay[i]))
            printf( "arrayay[%d]>0: %f\n",i,arrayay[i] );
        if (arrayay[i]>maxim) maxim=arrayay[i];
        if ((arrayay[i]>=lower) && (arrayay[i]<=upper)) count++;
    }
    printf( "Maximum=%f\n",maxim );
    printf( "Numbers within [%d,%d]: %d\n",lower,upper,count );

    return 0;
}
```

```
Give 1st number: -45632.4

Give 2nd number: 34.43

Give 3rd number: 12.34

Give 4th number: -11111.6

Give 5th number: 45.564999

Give 6th number: 43

array[1]>0: 34.430000
array[2]>0: 12.340000
array[4]>0: 45.564999
array[5]>0: 43.000000
Maximum=45.564999
Numbers within [1,05,50.8]: 4
```

Εικόνα 5.2 Η έξοδος του προγράμματος του παραδείγματος 5.1.2

### 5.1.3. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα δέχεται από το πληκτρολόγιο έναν θετικό ακέραιο αριθμό  $n$  ψηφίων  $x = x_{n-1}x_{n-2} \dots x_1x_0$ , θα αποθηκεύει τα ψηφία του  $x_k, k = 0, 1, \dots, n-1$  σε πίνακα και θα εμφανίζει στην οθόνη τον αριθμό με αντεστραμμένα τα ψηφία του, δηλαδή τον αριθμό  $y = y_{n-1}y_{n-2} \dots y_1y_0 = x_0x_1 \dots x_{n-2}x_{n-1}$ .

```
#include <stdio.h>
#define N 7

int main()
{
    int x,y,array[N],z[N],i;
    printf( "Give a %d digit positive integer: ",N );
    scanf( "%d",&x );
    z[0]=1;
    for (i=1;i<N;i++) z[i]=z[i-1]*10;
    y=x;
    for (i=(N-1);i>=1;i-)
    {
        array[i]=y/z[i];
        y=y%z[i];
    }
    array[0]=y;
    y=array[N-1];
    for (i=1;i<N;i++) y=y+z[i]*array[N-(i+1)];
    printf( "\nx=%d, y=%d\n",x,y );

    return 0 ;
}
```

Give a 7 digit positive integer: 1234567  
x=1234567, y=7654321

Εικόνα 5.3 Η έξοδος του προγράμματος του παραδείγματος 5.1.3

## 5.2. Πολυδιάστατοι πίνακες

Οι πολυδιάστατοι πίνακες είναι πίνακες, τα στοιχεία των οποίων είναι επίσης πίνακες. Η πρόταση `int array[4][12]`; δηλώνει τη μεταβλητή `array` ως πίνακα 4 στοιχείων, όπου το κάθε στοιχείο είναι πίνακας 12 στοιχείων ακεραίων. Η γλώσσα C δεν θέτει περιορισμό στον αριθμό των διαστάσεων των πινάκων.

Ο πολυδιάστατος πίνακας N διαστάσεων αποθηκεύεται στη μνήμη ως μία ακολουθία στοιχείων μίας διάστασης, ωστόσο μπορεί να θεωρηθεί ως ένας μονοδιάστατος πίνακας, όπου κάθε στοιχείο του είναι ένας πίνακας N-1 διαστάσεων, δηλαδή ένας πολυδιάστατος πίνακας είναι ένας *πίνακας πινάκων*. Για παράδειγμα, έστω ο επόμενος τριδιαγώνιος πίνακας, οποίος εμφανίζει μη μηδενικά στοιχεία μόνο στην κύρια διαγώνιο και τις δύο διαγωνίους εκατέρωθέν της:

17	62	0	0	0
25	28	9	0	0
0	6	33	28	0
0	0	123	49	3
0	0	0	76	13

Εικόνα 5.4 Τριδιαγώνιος πίνακας.

Για να αποθηκευτεί το τετράγωνο αυτό σε πίνακα θα μπορούσε να γίνει η ακόλουθη δήλωση:

```
int tridiag[5][5]= { {17, 62, 0, 0, 0},  
                    {25, 28, 9, 0, 0},  
                    {0, 6, 33, 28, 0},  
                    {0, 0, 123, 49, 3},  
                    {0, 0, 0, 76, 13}  
};
```

Από τον προηγούμενο κώδικα γίνεται αντιληπτό ότι στην απόδοση των αρχικών τιμών οι τιμές των στοιχείων κάθε γραμμής περικλείονται σε άγκιστρα.

Για την αναφορά σε στοιχείο ενός πολυδιάστατου πίνακα θα πρέπει να καθοριστούν τόσοι δείκτες όσοι είναι αναγκαίοι. Έτσι, η έκφραση

`tridiag[1]`

αναφέρεται στη δεύτερη γραμμή του πίνακα, ενώ η έκφραση

`tridiag[1][3]`

αναφέρεται στο τέταρτο στοιχείο της δεύτερης γραμμής του πίνακα.

Οι πολυδιάστατοι πίνακες αποθηκεύονται κατά γραμμές, που σημαίνει ότι ο τελευταίος δείκτης θέσης μεταβάλλεται ταχύτερα κατά την προσπέλαση των στοιχείων. Για παράδειγμα, ο πίνακας που δηλώνεται ως:

```
int array[2][3]= { {0, 1, 2},
                  {3, 4, 5}
                };
```

αποθηκεύεται όπως φαίνεται στο **Σχήμα 5.1**:

	διεύθυνση	τιμή
array[0][0]	1000	0
array[0][1]	1004	1
array[0][2]	1008	2
array[1][0]	1012	3
array[1][1]	1016	4
array[1][2]	1020	5

**Σχήμα 5.1** Αποθήκευση πολυδιάστατου (δισδιάστατου) πίνακα

Όταν δηλώνεται ένας πίνακας χωρίς να αρχικοποιηθεί, το περιεχόμενο των θέσεών του είναι απροσδιόριστο. Κατά συνέπεια, δεν πρέπει ποτέ να χρησιμοποιείται ένα στοιχείο πίνακα, εάν δεν έχει λάβει πρώτα τιμή. Για παράδειγμα, ο ακόλουθος κώδικας

```
int i,j,array[3][3];
for (i=0;i<3;i++)
    for (j=0;j<3;j++) printf( "arr[%d] [%d]=%d\n",i,j,array[i][j] );
```

δίνει τα ακόλουθα αποτελέσματα, που παρουσιάζονται στην **Εικόνα 5.5**, τα οποία προφανώς είναι μη αναμενόμενα και μπορούν να οδηγήσουν σε ανεπιθύμητες καταστάσεις:

```
array[0][0]=1
array[0][1]=0
array[0][2]=4200201
array[1][0]=0
array[1][1]=0
array[1][2]=0
array[2][0]=3
array[2][1]=0
array[2][2]=21
```

**Εικόνα 5.5** Η έξοδος του προγράμματος

### 5.2.1. Αρχικοποίηση πολυδιάστατων πινάκων

Για την αρχικοποίηση ενός πολυδιάστατου πίνακα κάθε γραμμή αρχικών τιμών περικλείεται σε άγκιστρα. Εάν δεν υπάρχουν οι αναγκαίες αρχικές τιμές, τα επιπλέον στοιχεία λαμβάνουν αρχική τιμή **0**. Έτσι, στη δήλωση και αρχικοποίηση του ακόλουθου πίνακα

```
int array[4][4]= { {1, 2, 3, 4},
                   {5, 6},
                   {7, 8, 9}
                 };
```

η μεταβλητή **array** δηλώνεται ως πίνακας 4 γραμμών και 4 στηλών. Ωστόσο, έχουν αποδοθεί τιμές μόνο για τις 3 πρώτες γραμμές του πίνακα και μάλιστα για τη δεύτερη γραμμή έχει οριστεί η τιμή μόνο των δύο πρώτων στοιχείων, ενώ για την τρίτη γραμμή δεν έχει οριστεί η τιμή του τελευταίου στοιχείου της. Η ανωτέρω δήλωση οδηγεί στον ακόλουθο πίνακα:

1	2	3	4
5	6	0	0
7	8	9	0
0	0	0	0

Εάν δεν συμπεριληφθούν τα ενδιάμεσα άγκιστρα:

```
int array[4][4]= { 1, 2, 3, 4
                  5, 6
                  7, 8, 9 };
```

τα στοιχεία θα αποθηκευτούν στις διαδοχικές θέσεις μνήμης που έχουν δεσμευτεί για τον πίνακα **array** και θα προκύψει ο ακόλουθος πίνακας:

1	2	3	4
5	6	7	8
9	0	0	0
0	0	0	0

### Παρατηρήσεις:

1. Όπως και με τους πίνακες μίας διάστασης, έτσι και στους πολυδιάστατους πίνακες, εάν δεν δοθεί το μέγεθος (οι διαστάσεις) του πίνακα, ο μεταγλωττιστής θα το καθορίσει αυτόματα με βάση τον αριθμό αρχικών τιμών που παρουσιάζονται. Στους πολυδιάστατους πίνακες μπορεί να παραληφθεί ο αριθμός των στοιχείων μόνο της πρώτης διάστασης, καθώς ο μεταγλωττιστής μπορεί να τον υπολογίσει από τον αριθμό των αρχικών τιμών που διατίθενται. Η παρακάτω δήλωση αξιοποιεί τη δυνατότητα αυτή του μεταγλωττιστή:

```
int array[ ][4][2]= { { {1, 2}, {3, 4}, {5, 6}, {7, 8} },
                      { {9, 10}, {11, 12}, {13, 14}, {15, 16} }
                    };
```

Με την ανωτέρω δήλωση ο **array** δηλώνεται αυτόματα ως πίνακας **2x4x2**.

2. Η δήλωση

```
int array[ ][ ]={ 1, 2, 3, 4, 5, 6};
```

δεν είναι αποδεκτή, καθώς ο μεταγλωττιστής δεν μπορεί να γνωρίζει τι είδους θα ήταν αυτός ο πίνακας. Θα μπορούσε να το θεωρήσει είτε πίνακα **2x3** είτε **3x2**.

3. Οι καθοριστές εφαρμόζονται και στους πολυδιάστατους πίνακες. Η ακόλουθη πρόταση

```
float identityMatrix[2][2]={ [0][0]=1.0, [1][1]=1.0};
```

δημιουργεί έναν μοναδιαίο πίνακα διαστάσεων **2x2**. Ως συνήθως, η τιμή των μη αρχικοποιούμενων στοιχείων τίθεται αυτόματα στο **0**.

4. Η πρόταση

```
printf( "%d",array[1,2] );
```

είναι λανθασμένη στη γλώσσα C, αλλά δεν εντοπίζεται από τον μεταγλωττιστή και οδηγεί σε μη αναμενόμενα – επομένως ανεπιθύμητα – αποτελέσματα. Η σωστή πρόταση είναι

```
printf( "%d",array[1][2] );
```



## 5.2.2. Παράδειγμα

Σε μία εταιρεία εργάζονται 5 πωλητές. Οι συνολικές μηνιαίες αποδοχές κάθε πωλητή προκύπτουν από το άθροισμα του βασικού μισθού (700 €) και του 9% επί του μηνιαίου τζίρου που πέτυχε ο πωλητής τον εκάστοτε μήνα. Π.χ. εάν ο μηνιαίος τζίρος που πέτυχε ένας πωλητής είναι 12000€, τότε οι αποδοχές του αυτόν τον μήνα θα είναι  $700 + 0.09 \cdot 12000 = 1780$  €.

Με βάση τα παραπάνω να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Θα δημιουργεί πίνακα αριθμών κινητής υποδιαστολής, διαστάσεων **5x3**.
- Με χρήση κατάλληλης επαναληπτικής πρότασης θα δίνεται από το πληκτρολόγιο ο μηνιαίος τζίρος κάθε πωλητή και ακολούθως θα υπολογίζονται οι μηνιαίες αποδοχές, οι οποίες θα αποθηκεύονται στον ανωτέρω πίνακα και θα εμφανίζονται στην οθόνη. Ταυτόχρονα θα υπολογίζεται ο συνολικός τζίρος και θα εμφανίζεται στην οθόνη.
- Ο τζίρος κάθε πωλητή και το ποσοστό που συνεισφέρει κάθε πωλητής στον συνολικό τζίρο θα αποθηκεύονται στον ανωτέρω πίνακα. Τα ποσοστά θα εμφανίζονται στην οθόνη.
- Θα υπολογίζεται και θα εμφανίζεται στην οθόνη ο αριθμός των πωλητών, για τους οποίους οι μηνιαίες αποδοχές εντάσσονται σε κάποια από τις ακόλουθες κατηγορίες.

I. 700-1500€

II. 1501-2000€

III. περισσότερα από 2000€

```
#include <stdio.h>
#define N 5

#define BS 700
#define COEF 0.09
#define FC 1500
#define SC 1500

int main()
{
    float salesman[N][3], gross=0.0;
    int i, cat[3]={0,0,0};

    for (i=0; i<N; i++)
    {
        printf( "\nSalesman no %d, give the gross sales:  ", i+1 );
        scanf( "%.2f", &salesman[i][1] );
        salesman[i][0]=BS+COEF*salesman[i][1];
        gross=gross+salesman[i][1];
    }
    printf( "\nTotal gross sales: %.2f", gross );

    for (i=0; i<N; i++)
    {
        printf( "\nSalesman no %d, salary: %.2f", i+1, salesman[i][0] );
        salesman[i][2]=100*salesman[i][1]/gross;
        printf( " \nSalesman no %d contribution to the total gross sales: %.2f\n", i+1, salesman[i][2] );
        if (salesman[i][0]>2000) cat[2]++;
        else if (salesman[i][0]>1500) cat[1]++;
        else cat[0]++;
    }

    for (i=0; i<3; i++)
```

```

{
    if (cat[i]==0)
        printf( "\nNo salesman falls into category %d\n",i+1 );
    else if (cat[i]==1)
        printf( "\n1 salesman falls into category %d\n",i+1 );
    else printf( "\n%d salesmen fall into category %d\n",cat[i],i+1 );
}

return 0;
}

```

```

Salesman no 1, give the gross sales: 12000

Salesman no 2, give the gross sales: 15000

Salesman no 3, give the gross sales: 17000

Salesman no 4, give the gross sales: 30000

Salesman no 5, give the gross sales: 25000

Total gross sales: 99000.00
Salesman no 1, salary: 1780.00
Salesman no 1 contribution to the total gross sales: 12.12

Salesman no 2, salary: 2050.00
Salesman no 2 contribution to the total gross sales: 15.15

Salesman no 3, salary: 2230.00
Salesman no 3 contribution to the total gross sales: 17.17

Salesman no 4, salary: 3400.00
Salesman no 4 contribution to the total gross sales: 30.30

Salesman no 5, salary: 2950.00
Salesman no 4 contribution to the total gross sales: 25.25

No salesman falls into category 1

1 salesman falls into category 2

4 salesmen fall into category 3

```

**Εικόνα 5.6** Η έξοδος του προγράμματος του παραδείγματος 5.2.2

### 5.2.3. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Θα λαμβάνει από το πληκτρολόγιο τις τιμές ενός πίνακα αριθμών κινητής υποδιαστολής `arrFloat[3][3]`. Μέσω κατάλληλης επαναληπτικής πρότασης, οι τιμές θα πρέπει υποχρεωτικά να κείνται στο διάστημα [-12, 24]. Ο προκύπτων πίνακας θα εμφανίζεται στην οθόνη.

- Για κάθε στήλη του πίνακα `arrFloat` θα υπολογίζει το άθροισμα των στοιχείων της στήλης, οι τιμές των οποίων ανήκουν στο διάστημα `[-8, 8]`, και θα εμφανίζει το άθροισμα στην οθόνη.

```
#include <stdio.h>
#include <stdlib.h>

#define N 3

#define LOWER_1 -12
#define UPPER_1 24

#define LOWER_2 -8
#define UPPER_2 8

int main()
{
    int i,j,ln[N],count_1,count_2;
    float arrFloat[N][N],columnSum;
    char arrChar[N][26];

    for (i=0;i<N;i++)
        for (j=0;j<N;j++)
        {
            do
            {
                printf( "\narr_float[%d][%d] (it should belong to [%d,%d]):",
",i+1,j+1,LOWER_1,UPPER_1 );
                scanf("%f",&arrFloat[i][j]);
            } while ((arrFloat[i][j]<LOWER_1) ||
                (arrFloat[i][j]>UPPER_1));
        }

    printf( "\n\nArray of float numbers:" );
    for (i=0;i<N;i++)
    {
        printf("\n");
        for (j=0;j<N;j++) printf( "\t%10.3f",arrFloat[i][j] );
    }

    for (j=0;j<N;j++)
    {
        columnSum=0;
        for (i=0;i<N;i++)
        {
            if ((arrFloat[i][j]>LOWER_2) && (arrFloat[i][j]<=UPPER_2))
                columnSum+=arrFloat[i][j];
        }
        printf( "\nColumn %d:\n",j+1 );
        printf( "\tsum of elements within [%d,%d] =
%f\n",LOWER_2,UPPER_2,columnSum );
    }

    return 0;
}
```

```

arr_float[1][1] (it should belong to [-12,24]: 23.99
arr_float[1][2] (it should belong to [-12,24]: -61
arr_float[1][2] (it should belong to [-12,24]: 10
arr_float[1][3] (it should belong to [-12,24]: 345
arr_float[1][3] (it should belong to [-12,24]: -6.78
arr_float[2][1] (it should belong to [-12,24]: 0
arr_float[2][2] (it should belong to [-12,24]: 2.13
arr_float[2][3] (it should belong to [-12,24]: -11.65
arr_float[3][1] (it should belong to [-12,24]: 34.6
arr_float[3][1] (it should belong to [-12,24]: 5.6
arr_float[3][2] (it should belong to [-12,24]: 13
arr_float[3][3] (it should belong to [-12,24]: -0.98

Array of float numbers:
      23.990      10.000      -6.780
      0.000       2.130     -11.650
      5.600      13.000      -0.980

Column 1:
sum of elements within [-8,8] = 5.600000

Column 2:
sum of elements within [-8,8] = 2.130000

Column 3:
sum of elements within [-8,8] = -7.760000

```

Εικόνα 5.7 Η έξοδος του προγράμματος του παραδείγματος 5.2.3

### 5.3. Πίνακες μεταβλητού μήκους

Έως τώρα η δήλωση ενός πίνακα προϋπέθετε ότι οι διαστάσεις του θα ήταν εκ των προτέρων γνωστές, δηλαδή κατά τον χρόνο μεταγλώττισης του προγράμματος. Αυτός είναι ένας σημαντικός περιορισμός, καθώς σε πλήθος εφαρμογών οι διαστάσεις ενός πίνακα *προκύπτουν* κατά την εκτέλεση του προγράμματος (π.χ. ως αποτέλεσμα της εκτέλεσης μίας πρότασης **if-else**).

Στο πρότυπο C99 της γλώσσας C οι πίνακες ενισχύθηκαν με τη δυνατότητα να δηλώνονται οι διαστάσεις τους κατά τον χρόνο εκτέλεσης του προγράμματος. Οι πίνακες αυτοί ονομάζονται «πίνακες μεταβλητού μήκους» (variable length arrays, VLA). Σημειώνεται ότι οι διαστάσεις τους καθορίζονται *άπαξ*, όπως στους κλασικούς πίνακες, και δεν μπορούν να μεταβληθούν. Μία τέτοια δυνατότητα δίνει η δυναμική διαχείριση μνήμης, που θα παρουσιαστεί στο 7<sup>ο</sup> Κεφάλαιο.

### 5.3.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα χρησιμοποιήσει πίνακες μεταβλητού μήκους, για να επιτελέσει τα ακόλουθα:

- Θα δέχεται από το πληκτρολόγιο τις διαστάσεις δύο πινάκων **a** και **b**.
- Θα δημιουργεί τους δύο πίνακες κατά τον χρόνο εκτέλεσης του προγράμματος.
- Θα τροφοδοτεί τους πίνακες με δεδομένα και θα τους εμφανίζει στην οθόνη.
- Θα υλοποιεί τις πράξεις της πρόσθεσης, αφαίρεσης και πολλαπλασιασμού πινάκων, Προς τούτο θα υπάρχει μενού επιλογής πράξης. Μετά την επιλογή θα προηγείται έλεγχος των διαστάσεων των πινάκων και θα ακολουθεί η εκτέλεση των υπολογισμών και η εμφάνιση του προκύπτοντος πίνακα στην οθόνη.

```
#include <stdio.h>
#include <stdlib.h>

#define ADD 1
#define SUB 2
#define MUL 3

int main()
{
    int i,j,k,a_ln,a_cl,b_ln,b_cl,c_ln,c_cl,d_ln,d_cl;
    float counter;

    /* Ανάγνωση των δεδομένων: Χάριν ευκολίας, τα δεδομένα
       δεν αναγιγνώσκονται αλλά δίνονται από έναν αλγόριθμο */
    printf( "\nGive the number of lines for array a: " );
    scanf( "%d",&a_ln );
    printf( "\nGive the number of columns for array a: " );
    scanf( "%d",&a_cl );
    printf( "\nGive the number of lines for array b: " );
    scanf("%d",&b_ln);
    printf( "\nGive the number of columns for array b: " );
    scanf( "%d",&b_cl );
    float a[a_ln][a_cl],b[b_ln][b_cl]; /* δημιουργία πινάκων VLA */
    for (i=0;i<a_ln;i++) {
        for (j=0;j<a_cl;j++) {
            a[i][j]=2.0*(i+2)*(j+4)/((i+1)*(i+1));
        }
    }
    for (i=0;i<b_ln;i++) {
        for (j=0;j<b_cl;j++) {
            b[i][j]=4.0*(i+2)*(j+1)/((i+1)*(i+1));
        }
    }
    /* Εκτύπωση των πινάκων */
    printf( "\n\n\nArray a:\n\t" );
    for (i=0;i<a_ln;i++) {
        for (j=0;j<a_cl;j++) {
            printf( "%.3f\t",a[i][j] );
        }
        printf( "\n\t" );
    }
    printf( "\n\n\nArray b:\n\t" );
```

```

for (i=0;i<b_ln;i++) {
    for (j=0;j<b_cl;j++) {
        printf("%.3f\t",b[i][j]);
    }
    printf("\n\t");
}
int choice;
do
{
    printf( "\n Select one of the following:" );
    printf( "\n\t\t\t\t %d -> + (addition)\n",ADD );
    printf( "\n\t\t\t\t %d -> - (subtraction)\n",SUB );
    printf( "\n\t\t\t\t %d -> * (multiplication)\n",MUL );
    scanf( "%d",&choice );
} while ((choice!=1) && (choice!=2) && (choice!=3));
switch(choice)
{
    case ADD:
        if ((a_ln==b_ln) && (a_cl==b_cl))
        {
            printf( "\n\n\nArray a+b:\n\t" );
            for (i=0;i<a_ln;i++)
            {
                for (j=0;j<a_cl;j++)
                {
                    printf(" %.3f\t",a[i][j]+b[i][j] );
                }
                printf(" \n\t" );
            }
        }
        else
            printf( "\nERROR!! Arrays' dimensions are inconsistent!\n" );
        break;
    case SUB:
        if ((a_ln==b_ln) && (a_cl==b_cl))
        {
            printf( "\n\n\nArray a-b:\n\t" );
            for (i=0;i<a_ln;i++)
            {
                for (j=0;j<a_cl;j++)
                {
                    printf( "%.3f\t",a[i][j]-b[i][j] );
                }
                printf( "\n\t" );
            }
        }
        else
            printf( "\nERROR!! Arrays' dimensions are inconsistent!\n" );
        break;
    default:
        if (a_cl==b_ln)
        {
            printf( "\n\n\nArray axb:\n\t" );
            for (i=0;i<a_ln;i++)
            {

```

```

        for (j=0;j<b_cl;j++)
        {
            counter=0.0;
            for (k=0;k<a_cl;k++) counter=counter+(a[i][k]*b[k][j]);
            printf( "%8.3f  ",counter );
        }
        printf("\n  ");
    }
    else
        printf( "\nERROR!! Arrays' dimensions are inconsistent!\n" );
    break;
} /* τέλος της switch */

return 0;
}

```

```

Give the number of lines for array a: 3

Give the number of columns for array a: 4

Give the number of lines for array b: 4

Give the number of columns for array b: 3


Array a:
    16.000 20.000 24.000 28.000
    6.000  7.500  9.000 10.500
    3.556  4.444  5.333  6.222


Array b:
    8.000 16.000 24.000
    3.000  6.000  9.000
    1.778  2.500  5.333
    1.250  2.500  3.750


Select one of the following:
                                1 -> + (addition)
                                2 -> - (subtraction)
                                3 -> * (multiplication)
3


Array axb:
    265.667 531.333 797.000
    99.625 199.250 298.875
    59.037 118.074 177.111

```

**Εικόνα 5.8** Η έξοδος του προγράμματος του παραδείγματος 5.3.1

## 5.4. Πίνακες ως παράμετροι συναρτήσεων

Εάν κατά την κλήση μίας συνάρτησης η πραγματική παράμετρος είναι όνομα πίνακα (πχ. **arr**), δεν αποστέλλεται στη συνάρτηση ολόκληρος ο πίνακας αλλά μόνο η διεύθυνση μνήμης του πρώτου byte του πρώτου στοιχείου του πίνακα. Η παράμετρος στη δήλωση της συνάρτησης είναι ένα όνομα τοπικού πίνακα (π.χ. **localArr**), ο οποίος κρατά ένα αντίγραφο της ίδιας διεύθυνσης. Με τον τρόπο αυτό η μεταβλητή **localArr** στην πραγματικότητα διαχειρίζεται τις θέσεις μνήμης που καταλαμβάνει ο πίνακας της καλούσας συνάρτησης, κατά συνέπεια μπορεί να μεταβάλλει με έμμεσο τρόπο τις τιμές του πίνακα και αυτές οι μεταβολές να διατηρηθούν και μετά την ολοκλήρωση της κλήσης της συνάρτησης.

Επιπρόσθετα, στη δήλωση δεν αναφέρεται το ακριβές μέγεθός του αλλά μόνο το γεγονός ότι είναι πίνακας καθώς και ο τύπος στοιχείων του. Επομένως, ουσιαστικά γνωστοποιείται στον μεταγλωττιστή η διεύθυνση του πρώτου byte και η απόσταση (αριθμός bytes) μεταξύ των στοιχείων. Ο μηχανισμός κλήσης συνάρτησης με όρισμα πίνακα στηρίζεται στην έννοια των δεικτών και θα μελετηθεί εκτενώς στο επόμενο κεφάλαιο.

### 5.4.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα λαμβάνει 4 ακέραιους από το πληκτρολόγιο και θα τους αποδίδει σε πίνακα ακεραίων (**array[4]**). Στη συνέχεια, θα καλείται η συνάρτηση **void pwr(int localArray[], int arraySize)**, η οποία θα μεταβάλλει τις τιμές των στοιχείων του πίνακα, υψώνοντας κάθε τιμή στοιχείου του πίνακα **array** στο τετράγωνο. Η **main()** θα τελειώνει με την εμφάνιση των νέων τιμών του **array** στην οθόνη.

```
#include <stdio.h>
#include <stdlib.h>

void pwr (int localArray[], int arraySize);

int main()
{
    int array[4],i;
    for (i=0;i<4;i++)
    {
        printf( "\narray[%d]= ",i );
        scanf( "%d",&array[i] );
    }

    printf( "\n\nInitial array:\n" );
    for (i=0;i<4;i++) printf( "\t%d\n",array[i] );

    pwr(array,4);

    printf( "\n\nFinal array:\n" );
    for (i=0;i<4;i++) printf( "\t%d\n",array[i] );

    return 0;
}

void pwr (int localArray[], int arraySize)
{
    int i;
    for (i=0;i<4;i++) localArray[i]=localArray[i]*localArray[i];
}
```



```

array[0]= 12
array[0]= 11
array[0]= 6
array[0]= 8

Initial array:
    12
    11
    6
    8

Final array:
    144
    121
    36
    64

```

Εικόνα 5.9 Η έξοδος του προγράμματος του παραδείγματος 5.4.1

## 5.5. Το αλφαριθμητικό

Το αλφαριθμητικό ή **συμβολοσειρά (string)** είναι ένας πίνακας χαρακτήρων, ο οποίος τερματίζει με τον **μηδενικό χαρακτήρα (null)**. Ο μηδενικός χαρακτήρας έχει ASCII κωδικό **0** και αναπαρίσταται από την ακολουθία διαφυγής `'\0'`.

Η δήλωση του αλφαριθμητικού ακολουθεί τον εξής φορμαλισμό:

```
char όνομα_αλφαριθμητικού[μήκος_αλφαριθμητικού];
```

Η δήλωση περιλαμβάνει τρία τμήματα: α) τον τύπο δεδομένου, ο οποίος είναι πάντοτε `char`, β) το όνομα του αλφαριθμητικού και γ) το μήκος του, το οποίο πάντοτε περιλαμβάνει μία θέση παραπάνω από το μέγιστο σε μήκος προσδοκώμενο αλφαριθμητικό, καθώς πρέπει να λαμβάνεται πρόνοια για την αποθήκευση του μηδενικού χαρακτήρα. Έτσι, μία τυπική δήλωση ενός αλφαριθμητικού 100 χαρακτήρων έχει την ακόλουθη μορφή:

```
char stringName[101];
```

Τα αλφαριθμητικά μπορούν να εμφανίζονται μέσα στον κώδικα όπως οι αριθμητικές σταθερές, αποτελώντας τις **αλφαριθμητικές σταθερές**. Αλφαριθμητικές σταθερές χρησιμοποιήθηκαν στο Κεφάλαιο 2, περικλειόμενες σε διπλά εισαγωγικά. Για την αποθήκευσή τους χρησιμοποιούνται πίνακες χαρακτήρων, με τον μεταγλωττιστή να θέτει αυτόματα στο τέλος των αλφαριθμητικών έναν μηδενικό χαρακτήρα, για να προσδιορίσει το τέλος του. Έτσι, η αλφαριθμητική σταθερά **"Constant string"** απαιτεί για αποθήκευση 16 bytes, όπως φαίνεται παρακάτω:

```
'C','o','n','s','t','a','n','t',' ','s','t','r','i','n','g','\0'
```

**Παρατήρηση:** Θα πρέπει να σημειωθεί ότι υπάρχει διαφορά ανάμεσα στη σταθερά χαρακτήρα **'A'** και την αλφαριθμητική σταθερά **"A"**. Η πρώτη απαιτεί ένα byte για αποθήκευση, ενώ η δεύτερη απαιτεί ένα byte για τον χαρακτήρα **A** κι ένα byte για το **null**.

Ένας **πίνακας αλφαριθμητικών** είναι ένας διδιάστατος πίνακας, κάθε γραμμή του οποίου φιλοξενεί ένα αλφαριθμητικό. Μία τυπική δήλωση ενός πίνακα 10 αλφαριθμητικών των 50 χαρακτήρων έχει την ακόλουθη μορφή:

```
char stringArray[10][51];
```

## 5.6. Αρχικοποίηση αλφαριθμητικού

Η αρχικοποίηση αλφαριθμητικού μπορεί να γίνει με δύο τρόπους:

1. Με τη δήλωση, όπου ακολουθείται ο γενικός κανόνας απόδοσης αρχικής τιμής σε πίνακα:

```
char stringName[ ] = {'m','y',' ','s','t','r','i','n','g','\0'};
```

2. Με χρήση αλφαριθμητικής σταθεράς (προσφιλέστερος τρόπος):

```
char stringName[ ] = {"my string"};
```

Θα πρέπει να σημειωθεί ότι στη χρήση του γενικού κανόνα απόδοσης αρχικής τιμής σε πίνακα, ο προγραμματιστής πρέπει να περιλάβει ως τελευταία τιμή το **null**. Στον δεύτερο τρόπο απόδοσης αρχικής τιμής, αυτό το έργο το εκτελεί αυτόματα ο μεταγλωττιστής.

## 5.7. Είσοδος – έξοδος αλφαριθμητικών

### 5.7.1. Ανάγνωση αλφαριθμητικού

Η εισαγωγή αλφαριθμητικού από την κύρια είσοδο γίνεται με τη μορφοποιούμενη συνάρτηση **scanf()** και τον προσδιοριστή **%s**. Η πρόταση

```
scanf( "%s", stringName );
```

διαβάζει από την κύρια είσοδο ένα αλφαριθμητικό και το αποθηκεύει στη μεταβλητή **stringName**. Δεν χρειάζεται ο τελεστής & πριν από το όνομα της μεταβλητής isbn όπως συνέβαινε με τους άλλους τύπους δεδομένων, γιατί το όνομα του αλφαριθμητικού αναπαριστά τη διεύθυνση του πρώτου στοιχείου του.

Εναλλακτικά, η εισαγωγή αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης **gets()**, το πρωτότυπο της οποίας βρίσκεται στο αρχείο κεφαλίδας **stdio.gr** και έχει τη γενική μορφή

```
gets(όνομα_αλφαριθμητικού)
```

Καλείται η **gets()** με το όνομα του πίνακα χαρακτήρων ως όρισμα. Η **gets()** αναθέτει το αλφαριθμητικό στον πίνακα χαρακτήρων **όνομα\_αλφαριθμητικού**. Η **gets()** θα διαβάζει χαρακτήρες από το πληκτρολόγιο, έως ότου πατηθεί το ENTER.

Για να διαβαστεί ένα στοιχείο πίνακα αλφαριθμητικών, χρησιμοποιούνται οι προτάσεις

```
scanf( "%s", stringArray[i] ); και gets(stringArray[i]);
```

#### Παρατηρήσεις:

1. Η συνάρτηση **scanf()** διαβάζει εσφαλμένα τα αλφαριθμητικά που περιέχουν λευκούς χαρακτήρες, καθώς σταματά την ανάγνωση στην εμφάνιση του πρώτου λευκού χαρακτήρα. Αντίθετα, η συνάρτηση **gets()** διαβάζει τους λευκούς χαρακτήρες.
2. Θα πρέπει να σημειωθεί ότι τόσο η **scanf()** όσο και η **gets()** δεν εκτελούν έλεγχο ορίων στον πίνακα χαρακτήρων, με τον οποίο καλούνται. Εάν π.χ. δηλωθεί **char stringName[100]** και το αλφαριθμητικό είναι μεγαλύτερο από το μέγεθος του **stringName**, ο πίνακας θα ξεπεραστεί. Επαφίεται στον προγραμματιστή να φροντίζει, ώστε να μην προκληθεί υπέρβαση ορίων.
3. Όταν χρησιμοποιούνται πίνακες αλφαριθμητικών, η πρόταση

```
scanf( "%c", stringArray[i][j] );
```

διαβάζει τον χαρακτήρα του αλφαριθμητικού **i+1**, ο οποίος βρίσκεται στη θέση **j+1**.

### 5.7.2. Εκτύπωση αλφαριθμητικού

Η εκτύπωση αλφαριθμητικής σταθεράς γίνεται με τη συνάρτηση `printf()` χωρίς τη χρήση προσδιοριστή. Απλώς της δίνεται η προς εκτύπωση αλφαριθμητική σταθερά:

```
printf( "My string" );
```

Η εκτύπωση αλφαριθμητικού γίνεται με την `printf()` χρησιμοποιώντας τον προσδιοριστή `%s`. Η παρακάτω πρόταση

```
printf( "This is %s!!", stringName );
```

θα έχει ως αποτέλεσμα να τυπωθεί στην οθόνη η πρόταση

```
This is my string!!
```

Εναλλακτικά, η εκτύπωση αλφαριθμητικής σταθεράς και αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης `puts()`, το πρωτότυπο της οποίας βρίσκεται στο αρχείο κεφαλίδας `stdio.gr` και έχει τη γενική μορφή

```
puts(όνομα_αλφαριθμητικού)
```

Καλείται η `puts()` με το όνομα του πίνακα χαρακτήρων ως όρισμα, χωρίς δείκτη, π.χ. `puts(stringName)`. Σε αντιδιαστολή με τη συνάρτηση `printf()`, η `puts()` δεν παρέχει δυνατότητες μορφοποίησης της εξόδου.

Για να εκτυπωθεί ένα στοιχείο πίνακα αλφαριθμητικών, χρησιμοποιούνται οι προτάσεις

```
printf( "%s", stringArray[i] );      και      puts(stringArray[i]);
```

ενώ η πρόταση

```
printf( "%c", stringArray[i][j] );
```

εκτυπώνει τον χαρακτήρα του αλφαριθμητικού `i+1`, ο οποίος βρίσκεται στη θέση `j+1`.

#### 5.7.3.1. Παράδειγμα

Στο ακόλουθο πρόγραμμα γίνεται εισαγωγή και εκτύπωση αλφαριθμητικών με όλους τους τρόπους που περιγράφηκαν ανωτέρω. Θα πρέπει να προσεχθεί η χρήση της `define` για τη χρήση αλφαριθμητικής σταθεράς.

```
#include <stdio.h>

#define STA "Hello"

int main()
{
    char str1[ ]="First String";
    char str2[81];
    puts(STA);
    printf( "\nstr1 is: %s\nGive str2:",str1 );
    gets(str2);
    printf( "\nstr2 is: %s\nGive another str2:",str2 );
    scanf( "%s",str2 );
    printf( "\nNew str2 is: " );
    puts(str2);

    return 0;
}
```

```

Hello

Str1 is: First String
Give str2:Second string

Str2 is: Second string
Give another str2:another_string

New str2 is: another_string

```

Εικόνα 5.10 Η έξοδος του προγράμματος του παραδείγματος 5.7.3.1

## 5.8. Μετατροπές αλφαριθμητικών σε αριθμητικές τιμές

Ένα αλφαριθμητικό που αποτελείται από ψηφία μπορεί να μετατραπεί σε αριθμό με χρήση των ακόλουθων συναρτήσεων, τα πρότυπα των οποίων βρίσκονται στο αρχείο κεφαλίδας **stdlib.h**:

- Η συνάρτηση **atoi()** δέχεται ως όρισμα ένα αλφαριθμητικό και επιστρέφει την ακέραια τιμή του ή, εφόσον δεν είναι εφικτή η μετατροπή, το μηδέν.
- Η συνάρτηση **atol()** δέχεται ως όρισμα ένα αλφαριθμητικό και επιστρέφει την τιμή του ως **long int** ή, εφόσον δεν είναι εφικτή η μετατροπή, το μηδέν.
- Η συνάρτηση **atof()** δέχεται ως όρισμα ένα αλφαριθμητικό και επιστρέφει την τιμή του ως αριθμό κινητής υποδιαστολής μονής ακρίβειας ή, εφόσον δεν είναι εφικτή η μετατροπή, το μηδέν.

Το αλφαριθμητικό μπορεί να περιέχει κενά στην αρχή και το τέλος του. Η μετατροπή σταματά με την εμφάνιση του πρώτου μη αποδεκτού χαρακτήρα.

### 5.8.1. Παράδειγμα

Στο πρόγραμμα που ακολουθεί παρουσιάζονται τα χαρακτηριστικά των συναρτήσεων **atoi()**, **atol()**, **atof()**:

```

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int val1;
    long int val2;
    double val3;

    char str1[20]="123456",str2[20]="abcd",str3[20]=" 1234bn";
    printf( "atoi example:\n" );
    val1 = atoi(str1);
    printf( "Str1 = \"%s\", Integer value = %d\n", str1, val1 );
    val1 = atol(str2);
    printf( "Str2 = \"%s\", Integer value = %d\n", str2, val1 );
    val1 = atol(str3);
    printf( "Str3 = \"%s\", Integer value = %d\n", str3, val1 );

    printf("\n\natol example:\n");
    val2 = atol(str1);

```

```

printf( "Str1 = \"%s\\", Long integer value = %ld\\n", str1, val2 );
val2 = atol(str2);
printf( "Str2 = \"%s\\", Long integer value = %ld\\n", str2, val2 );

printf( "\\n\\natof example:\\n" );
val3 = atof(str1);
printf( "Str1 = \"%s\\", Double value = %f\\n", str1, val3 );
val3 = atof(str2);
printf( "Str2 = \"%s\\", Double value = %f\\n", str2, val3 );

return(0);
}

```

```

atoi example:
Str1 = "123456", Integer value = 123456
Str2 = "abcd", Integer value = 0
Str3 = " 1234bn", Integer value = 1234

atol example:
Str1 = "123456", Long Integer value = 123456
Str2 = "abcd", Long Integer value = 0

atof example:
Str1 = "123456", Double value = 123456.000000
Str2 = "abcd", Double value = 0.000000

```

Εικόνα 5.11 Η έξοδος του προγράμματος του παραδείγματος 5.8.1

## 5.9. Συναρτήσεις αλφαριθμητικών

Η γλώσσα C υποστηρίζει μία σειρά συναρτήσεων για τον χειρισμό των αλφαριθμητικών. Οι συναρτήσεις αυτές βρίσκονται στο αρχείο κεφαλίδας **string.h**. Οι πλέον συνήθεις παρουσιάζονται στον **Πίνακα 5.1**, στον οποίο η δεύτερη και η τρίτη στήλη περιέχουν τα ονόματα των συναρτήσεων, όταν η λειτουργία τους επιδρά σε ολόκληρο το αλφαριθμητικό ή στους πρώτους **n** χαρακτήρες, αντίστοιχα:

Λειτουργία	Όλοι οι χαρακτήρες	Οι <i>n</i> πρώτοι χαρακτήρες
Εύρεση μήκους αλφαριθμητικού	<b>strlen()</b>	
Αντιγραφή αλφαριθμητικού	<b>strcpy()</b>	<b>strncpy()</b>
Συνένωση δύο αλφαριθμητικών	<b>strcat()</b>	<b>strncat()</b>
Σύγκριση δύο αλφαριθμητικών	<b>strcmp()</b>	<b>strncmp()</b>
Εύρεση χαρακτήρα σε αλφαριθμητικό	<b>strchr()</b>	<b>strrchr()</b>
Εύρεση αλφαριθμητικού σε αλφαριθμητικό	<b>strstr()</b>	

Πίνακας 5.1 Συναρτήσεις αλφαριθμητικών

Στο παρόν κεφάλαιο θα μελετηθούν οι τέσσερις πρώτες συναρτήσεις του **Πίνακα 5.1**. Οι υπόλοιπες δύο συναρτήσεις θα μελετηθούν σε επόμενο κεφάλαιο, καθώς η ανάλυσή τους στηρίζεται στη χρήση δεικτών.

**Παρατήρηση:** Στις συναρτήσεις **strcpy()** και **strcat()** ελλοχεύει ο κίνδυνος να ξεπεραστούν τα όρια του πίνακα χαρακτήρων προορισμού, με αποτέλεσμα να δημιουργηθούν απροσδιόριστες καταστάσεις. Εάν π.χ. έχει οριστεί ο πίνακας χαρακτήρων **char array[4]**, ο οποίος καταλαμβάνει τις θέσεις **1000** έως και **1003** στον χάρτη μνήμης του **Σχήματος 5.2**, και επιχειρηθεί να τοποθετηθεί σε αυτό το αλφαριθμητικό

**"character"**

θα τεθεί θέμα απροσδιοριστίας. Συγκεκριμένα, στις θέσεις **1004-1009**, τις οποίες δεν διαχειρίζεται ο **array**, θα τοποθετηθούν οι χαρακτήρες 'a', 'c', 't', 'e', 'r', '\0'. Οι θέσεις, όμως, αυτές μπορεί να χρησιμοποιούνται από άλλες μεταβλητές και με την τροποποίηση του περιεχομένου τους οι νέες τιμές να προκαλέσουν σημασιολογικά σφάλματα στο πρόγραμμα.

Η εγγραφή δεδομένων εκτός των ορίων ενός πίνακα στην περιοχή προσωρινής αποθήκευσης ονομάζεται **υπερχείλιση της περιοχής προσωρινής αποθήκευσης** (buffer overflow). Για αυτόν τον λόγο θα πρέπει ο προγραμματιστής είτε να έχει προβλέψει επαρκή χώρο είτε σε κάθε χρήση αυτών των συναρτήσεων να εκτελεί έλεγχο ορίων, πριν τις χρησιμοποιήσει, όπως χαρακτηριστικά παρουσιάζεται στα παραδείγματα 5.9.2.1 και 5.9.3.1.

	διεύθυνση	τιμή
<b>array[0]</b>	1000	c
<b>array[1]</b>	1001	h
<b>array[2]</b>	1002	a
<b>array[3]</b>	1003	r
<b>array[4]</b>	1004	a
<b>array[5]</b>	1005	c
<b>array[6]</b>	1006	t
<b>array[7]</b>	1007	e
<b>array[8]</b>	1008	r
<b>array[9]</b>	1009	\0

Σχήμα 5.2 Υπερχείλιση της περιοχής προσωρινής αποθήκευσης

### 5.9.1. Η συνάρτηση εύρεσης μήκους αλφαριθμητικού

Η συνάρτηση **strlen()** (string length) επιστρέφει τον αριθμό χαρακτήρων του αλφαριθμητικού, χωρίς να συμπεριλαμβάνει τον μηδενικό χαρακτήρα. Το παρακάτω τμήμα κώδικα

```
char stringName[20] = "My string";
printf( "%d\n", strlen(stringName) );
```

θα τυπώσει τον αριθμό των χαρακτήρων του αλφαριθμητικού **stringName**, δηλαδή **9** κι όχι 20, που είναι ο αριθμός των στοιχείων του πίνακα χαρακτήρων **stringName**.

Μία υλοποίηση της **strlen()** ως συνάρτηση, η οποία θα δέχεται ως είσοδο το αλφαριθμητικό και θα επιστρέφει ως ακέραιο τον αριθμό των χαρακτήρων του αλφαριθμητικού, δίνεται ακολούθως:

```
int strlenImplementation(char stringName[])
{
    int i=0;
    while (stringName[i]!='\0') i++;
}
```

```

        return(i) ;
    }

```

### 5.9.1.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- (α) Θα δέχεται από το πληκτρολόγιο δύο αλφαριθμητικά (μέγιστου μήκους 12) και θα τα αποθηκεύει.
- (β) Θα ελέγχει εάν οι τελευταίοι 4 χαρακτήρες του πρώτου αλφαριθμητικού είναι ίδιοι με τους τελευταίους 4 χαρακτήρες του δεύτερου αλφαριθμητικού (ελέγχοντας πρώτα εάν τα αναγνωσθέντα αλφαριθμητικά έχουν μήκος μεγαλύτερο ή ίσο του 4).
- (γ) Θα αντιγράφει σε νέο πίνακα χαρακτήρων κατάλληλου μήκους τους χαρακτήρες του πρώτου αλφαριθμητικού που βρίσκονται σε άρτια θέση και θα τυπώνει το νέο πίνακα χαρακτήρων στην οθόνη ως αλφαριθμητικό.

```

#include <stdio.h>
#include <string.h>

int main()
{
    int i,j,sum;
    char str1[13],str2[13],str3[7];
    /* α */
    printf( "\nGive first string (up to 12 characters):  ");
    scanf("%s",str1);
    printf( "\nGive second string (up to 12 characters):  ");
    scanf("%s",str2);

    /* β */
    if ((strlen(str1)<4) || (strlen(str2)<4))
        printf( "\nError!! Strings should have more at least 4 characters!! Program aborted" );
    else
    {
        sum=0;
        for (i=0;i<4;i++)
            if (str1[strlen(str1)-i]==str2[strlen(str2)-i])
                sum++;
        if (sum==4)
            printf( "\n\tThe last 4 characters of first and second string are the same\n" );
        else
        {
            printf( "\n\tThe last 4 characters of first and second string are NOT the same\n" );
        }
    }
    /* γ */
    for (i=1;i<strlen(str1);i=i+2) str3[i/2]=str1[i];
    str3[i/2]='\0'; /* για να καταστεί αλφαριθμητικό το str3 */
    printf( "\nNew string: %s",str3 );

    return 0;
}

```

```

Give first string (up to 12 characters): Quadrature

Give second string (up to 12 characters): Torture

The last 4 characters of the first and second string are the same

New string: udaue

```

Εικόνα 5.12 Η έξοδος του προγράμματος του παραδείγματος 5.9.1.1

## 5.9.2. Η συνάρτηση αντιγραφής αλφαριθμητικού

Η συνάρτηση `strcpy()` (string copy) αντιγράφει ένα αλφαριθμητικό σε ένα άλλο. Δέχεται δύο ορίσματα που είναι τα ονόματα των αλφαριθμητικών. *Ως πρώτο όρισμα τίθεται το αλφαριθμητικό προορισμού, ενώ ως δεύτερο όρισμα ορίζεται το προς αντιγραφή αλφαριθμητικό.* Στο παρακάτω τμήμα κώδικα

```

char str1[12] = "initial string";
char str[12] = "final string";
strcpy(str1, str2);
printf( "%s\n", str1 );

```

η πρόταση `strcpy(str1, str2);` αντιγράφει το περιεχόμενο του `str2` στον πίνακα χαρακτήρων `str1`. Έτσι, στην οθόνη θα εμφανιστεί η φράση **final string**.

Για να αντιγραφούν οι πρώτοι `n` χαρακτήρες του `str2` χρησιμοποιείται η σύνταξη `strncpy(str1, str2, n)`, όπου το τρίτο όρισμα είναι ο αριθμός των προς αντιγραφή χαρακτήρων.

**Παρατήρηση:** Η χρήση της συνάρτησης `strcpy()` αποτελεί τον τρόπο ανάθεσης ή εκχώρησης τιμής σε αλφαριθμητικό, καθώς η απευθείας ανάθεση τιμής επιστρέφεται μόνο στην αρχικοποίηση. Δηλαδή, η πρόταση

```
str1="Get a string";
```

δεν επιτρέπεται στη γλώσσα C. Η ανάθεση γίνεται μέσω της πρότασης:

```
strcpy(str1, "Get a string");
```

### 5.9.2.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

(α) Θα δέχεται από το πληκτρολόγιο δύο αλφαριθμητικά και θα τα αποθηκεύει στον πίνακα αλφαριθμητικών `str[2][41]`. Κατά την ανάγνωση των αλφαριθμητικών θα γίνεται έλεγχος μέσω επαναληπτικής πρότασης, ώστε το μήκος τους να μην υπερβαίνει το **20**. Τα δοθέντα αλφαριθμητικά θα εμφανίζονται στην οθόνη.

(β) Τα δύο ανωτέρω αλφαριθμητικά θα μετασχηματίζονται με αφαίρεση των χαρακτήρων που βρίσκονται σε άρτιες θέσεις (π.χ. το αλφαριθμητικό **"my string"** θα μετασχηματιστεί σε **"m tig"**). Τα μετασχηματισμένα αλφαριθμητικά θα αποθηκεύονται στις ίδιες γραμμές του πίνακα και θα εμφανίζονται στην οθόνη.

```

#include <stdio.h>
#include <string.h>

int main()
{
    int i,j,length[2];
    char str[2][41],temp[21];
    /* (α) */

```



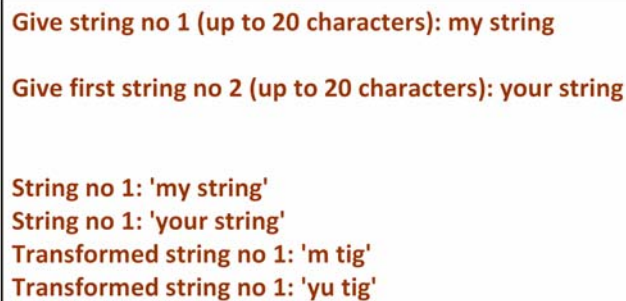
```

for (i=0;i<2;i++)
{
    do
    {
        printf( "\nGive string no %d (up to 20 characters): ",i+1);
        gets(str[i]);
        length[i]=strlen(str[i]);
    } while (length[i]>20);
}
printf("\n\n");
for (i=0;i<2;i++) printf( "\nString no %d: '%s'",i+1,str[i] );

/* (β) */
for (i=0;i<2;i++)
{
    for (j=0;j<length[i];j=j+2) temp[j/2]=str[i][j];
    temp[j/2]='\0';
    strcpy(str[i],temp);
    printf( "\nTransformed string no %d: '%s'",i+1,str[i] );
}

return 0;
}

```



```

Give string no 1 (up to 20 characters): my string

Give first string no 2 (up to 20 characters): your string


String no 1: 'my string'
String no 1: 'your string'
Transformed string no 1: 'm tig'
Transformed string no 1: 'yu tig'

```

Εικόνα 5.13 Η έξοδος του προγράμματος του παραδείγματος 5.9.2.1

### 5.9.3. Η συνάρτηση συνένωσης αλφαριθμητικών

Η συνάρτηση **strcat()** (string concatenation) δέχεται δύο ορίσματα που είναι τα ονόματα των αλφαριθμητικών, τα οποία και συνενώνει. Συγκεκριμένα, προσθέτει στο τέλος του αλφαριθμητικού που προσδιορίζεται από το πρώτο όρισμα, τα στοιχεία του αλφαριθμητικού που προσδιορίζεται από το δεύτερο όρισμα. Στο παρακάτω τμήμα κώδικα

```

char str1[30] = "first string";
char str2[30] = "second string";
strcat(str1,str2);
printf( "%sn", str1 );

```

προστίθεται στο τέλος του πίνακα χαρακτήρων **str1** το περιεχόμενο του πίνακα **str2**. Έτσι, στην οθόνη θα εμφανιστεί το αλφαριθμητικό **first stringsecond string**.

Για να προστεθούν οι πρώτοι **n** χαρακτήρες του **str2**, χρησιμοποιείται η σύνταξη **strncat(str1, str2, n)**, όπου το τρίτο όρισμα είναι ο αριθμός των προστιθέμενων χαρακτήρων.

### 5.9.3.1. Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

(α) Θα δέχεται από το πληκτρολόγιο δύο αλφαριθμητικά και θα τα αποθηκεύει στον πίνακα αλφαριθμητικών `str[2][41]`. Τα δοθέντα αλφαριθμητικά θα εμφανίζονται στην οθόνη.

(β) Το δεύτερο αλφαριθμητικό θα συνενώνεται με το πρώτο, αφού προηγηθεί έλεγχος κατά πόσον υπάρχει διαθέσιμος χώρος για τη συνένωση. Εφόσον γίνει η συνένωση, το τροποποιημένο πρώτο αλφαριθμητικό θα εμφανίζεται στην οθόνη.

```
#include <stdio.h>
#include <string.h>

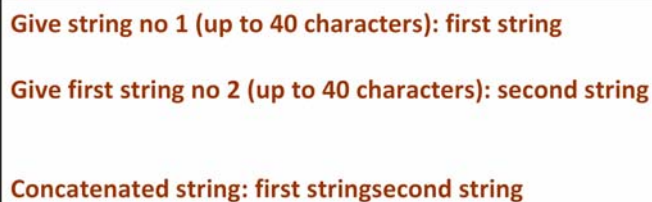
int main()
{
    char str[2][41];
    int i;

    for (i=0;i<2;i++)
    {
        printf( "\nGive string no %d (up to 40 characters):  ",i+1 );
        gets(str[i]);
    }
    printf("\n\n");

    if (strlen(str[0])+strlen(str[1])<=40)  strcat(str[0],str[1]);
    else
        printf( "\nERROR! The string cannot be concatenated." );

    printf( "\nConcatenated string: %s",str[0] );

    return 0;
}
```



```
Give string no 1 (up to 40 characters): first string
Give first string no 2 (up to 40 characters): second string
Concatenated string: first stringsecond string
```

Εικόνα 5.14 Η έξοδος του προγράμματος του παραδείγματος 5.9.3.1

### 5.9.4. Η συνάρτηση σύγκρισης αλφαριθμητικών

Η συνάρτηση `strcmp(str1, str2)` δέχεται δύο ορίσματα που είναι τα ονόματα των αλφαριθμητικών, τα οποία και συγκρίνει. Η έξοδος της είναι ένας ακέραιος αριθμός, ο οποίος λαμβάνει την τιμή **0**, εφόσον τα αλφαριθμητικά είναι όμοια.

Για να συγκριθούν οι πρώτοι **n** χαρακτήρες των αλφαριθμητικών, χρησιμοποιείται η σύνταξη `strncmp(str1, str2, n)`, όπου το τρίτο όρισμα είναι ο αριθμός των προς σύγκριση χαρακτήρων.

### 5.9.4.1. Παράδειγμα

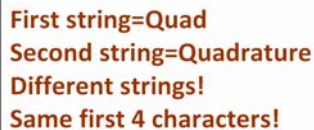
Να περιγραφεί η λειτουργία του ακόλουθου προγράμματος:

```
#include <stdio.h>
#include <string.h>

#define N 4

int main()
{
    char msg1[41] = {"Quad"};
    char msg2[41]={"Quadrature"};
    int diff;
    printf( "First string=%s\n",msg1 );
    printf( "Second string=%s\n",msg2 );
    diff=strcmp(msg1,msg2);
    if(diff==0) printf( "Same strings!\n" );
    else printf( "Different strings!\n" );
    diff=strncmp(msg1,msg2,N);
    if(diff==0) printf( "Same first %d characters!\n",N );
    else printf( "Different first %d characters!\n",N );

    return 0;
}
```



```
First string=Quad
Second string=Quadrature
Different strings!
Same first 4 characters!
```

Εικόνα 5.15 Η έξοδος του προγράμματος του παραδείγματος 5.9.4.1

Δηλώνονται δύο πίνακες χαρακτήρων **41** θέσεων, στους οποίους αποδίδονται τα περιεχόμενα **"Quad"** και **"Quadrature"**, αντίστοιχα.

Ακολούθως, συγκρίνονται τα δύο αλφαριθμητικά με χρήση της **strcmp()** και κατόπιν συγκρίνονται οι τέσσερις πρώτοι χαρακτήρες τους με χρήση της **strncmp()**.

## Ερωτήσεις αυτοαξιολόγησης - ασκήσεις

### Ερωτήσεις αυτοαξιολόγησης

Ο αναγνώστης καλείται να επιλέξει μία από τις τέσσερις απαντήσεις.

- (1) Ποιες από τις ακόλουθες παρατηρήσεις, που αφορούν στους πολυδιάστατους πίνακες, είναι λανθασμένη;
- (α) Εάν αμεληθεί να δοθεί το μέγεθος του πίνακα, ο μεταγλωττιστής θα το καθορίσει αυτόματα με βάση τον αριθμό αρχικών τιμών που παρουσιάζονται. Ωστόσο, στους πολυδιάστατους πίνακες μπορεί να παραληφθεί ο αριθμός των στοιχείων μόνο της πρώτης διάστασης, καθώς ο μεταγλωττιστής μπορεί να τον υπολογίσει από τον αριθμό των αρχικών τιμών που διατίθενται.
- (β) Η δήλωση **int ar[ ][ ]={ 1, 2, 3, 4, 5, 6};** είναι ανεπίτρεπτη, επειδή ο μεταγλωττιστής δεν μπορεί να γνωρίζει τι είδους θα ήταν αυτός ο πίνακας. Θα μπορούσε να τον θεωρήσει είτε πίνακα 2x3 είτε 3x2.

- (γ) Η πρόταση `printf( "%d",array[1,2] );` είναι λανθασμένη στη γλώσσα C αλλά δεν εντοπίζεται από τον μεταγλωττιστή και οδηγεί σε ανεπιθύμητα αποτελέσματα.
- (δ) Η πρόταση `printf( "%d",array[1][2] );` είναι λανθασμένη στη γλώσσα C και εντοπίζεται από τον μεταγλωττιστή.

(2) Τι θα εμφανίσει στην οθόνη το ακόλουθο τμήμα κώδικα;

```
char name1[12] = "abcd";
char name2[12] = "ef";
strcpy(name1,name2);
printf( "%s\n", name1 );
```

- (α) `ef`  
 (β) `abcdef`  
 (γ) `efabcd`  
 (δ) `efcd`

(3) Ποιο πρόγραμμα ικανοποιεί τις ακόλουθες προδιαγραφές:

- (i) Θα εισάγονται από το πληκτρολόγιο 4 αλφαριθμητικά σε πίνακα αλφαριθμητικών, μήκους 7 χαρακτήρων το καθένα.
- (ii) Θα λαμβάνονται οι τρεις πρώτοι χαρακτήρες κάθε αλφαριθμητικού και θα συνενώνονται σε ένα νέο αλφαριθμητικό, το οποίο και θα τυπώνεται.

(α) `#include <stdio.h>`  
`#include <string.h>`  
`int main()`  
`{`  
 `int i;`  
 `char str[4][7], str_total[13];`  
 `for (i=0;i<4;i++)`  
 `{`  
 `printf( "\nGive string no %d: ",i+1 );`  
 `scanf( "%s",str[i] );`  
 `if (i==0) strncpy(str_total,str[i],3);`  
 `else strncat(str_total,str[i],3);`  
 `}`  
 `printf( "Total string: %s\n",str_total );`  
 `return 0;`  
`}`

(β) `#include <stdio.h>`  
`int main()`  
`{`  
 `int i;`  
 `char str[4][8], str_total[13];`  
 `for (i=0;i<4;i++)`  
 `{`  
 `printf( "\nGive string no %d: ",i+1 );`  
 `scanf( "%s",str[i] );`  
 `if (i==0) strncat(str_total,str[i],3);`  
 `else strncat(str_total,str[i],3);`  
 `}`  
 `printf( "Total string: %s\n",str_total );`  
 `return 0;`  
`}`

```
(γ) #include <stdio.h>
#include <string.h>
int main()
{
    int i;
    char str[4][8], str_total[13];
    for (i=0;i<4;i++)
    {
        printf( "\nGive string no %d: ",i+1 );
        scanf( "%s",str[i] );
        if (i==0) strncpy(str_total,str[i],3);
        else strncat(str_total,str[i],3);
    }
    printf( "Total string: %s\n",str_total );

    return 0;
}
```

```
(δ) #include <stdio.h>
#include <string.h>
int main()
{
    int i;
    char str[4][7], str_total[12];
    for (i=0;i<4;i++)
    {
        printf( "\nGive string no %d: ",i+1 );
        scanf( "%s",str[i] );
        if (i==0) strncpy(str_total,str[i],3);
        else strncpy(str_total,str[i],3);
    }
    printf( "Total string: %s\n",str_total );

    return 0;
}
```

(4) Ποιο είναι το αποτέλεσμα της πρότασης `for (i=0,j=10; i<8; i++,j++) t[j]=s[i];`

(α) Αντιγραφή των οκτώ πρώτων στοιχείων του πίνακα `s` στον `t`, ξεκινώντας από το ενδέκατο στοιχείο του `t`.

(β) Αντιγραφή των οκτώ πρώτων στοιχείων του πίνακα `s` στον `t`.

(γ) Αντιγραφή των έντεκα πρώτων στοιχείων του πίνακα `s` στον `t`, ξεκινώντας από το όγδοο στοιχείο του `t`.

(δ) Αντιγραφή των τεσσάρων πρώτων στοιχείων του πίνακα `s` στον `t`, ξεκινώντας από το ενδέκατο στοιχείο του `t`.

(5) Ποιο είναι το αποτέλεσμα του ακόλουθου προγράμματος;

```
#include <stdio.h>
#define SIZE 10

int func(int b[], int p);

int main()
{
    int x;
    int a[SIZE]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    x=func(a,SIZE);
    printf( "The result is %d\n", x );
}
```

```

        return 0;
    }

    int func(int b[], int p)
    {
        if (p==1) return(b[0]);
        else return(b[p-1]+func(b,p-1));
    }

```

- (α) The result is 3628800  
 (β) The result is 55  
 (γ) The result is 433  
 (δ) The result is 0

## Ασκήσεις

### Άσκηση 1

Να γραφεί πρόγραμμα με το οποίο θα εισάγονται 6 χαρακτήρες από το πληκτρολόγιο, θα αποθηκεύονται στον πίνακα `array[]` και θα τυπώνονται διαδοχικά τα ακόλουθα:

- (α) οι χαρακτήρες με δεκαδικό ισοδύναμο μικρότερο του 75,  
 (β) ο χαρακτήρας με το μικρότερο δεκαδικό ισοδύναμο,  
 (γ) όσοι χαρακτήρες είναι διάφοροι των χαρακτήρων 'b', 'c', 'd' (υλοποίηση αποκλειστικά με χρήση της εντολής `switch-case`).

### Άσκηση 2

Να γραφεί πρόγραμμα, το οποίο θα χρησιμοποιεί έναν πίνακα 1000 στοιχείων για να καθορίσει και να προβάλει τους πρώτους αριθμούς ανάμεσα στο 1 και το 999. Το στοιχείο 0 του πίνακα θα αγνοηθεί. Η υλοποίηση στηρίζεται στα παρακάτω:

- Πρώτος αριθμός καλείται οποιοσδήποτε ακέραιος μπορεί να διαιρεθεί μόνο με τον εαυτό του και το 1.
- Ο αλγόριθμος υπολογισμού ονομάζεται *κόσκινο* (ή *κρησάρα*) του *Ερατοσθένους* και λειτουργεί ως εξής:
  - Δημιουργείται ένας πίνακας με όλα τα στοιχεία να έχουν τιμή 1. Τα στοιχεία του πίνακα με αριθμοδείκτη πρώτο αριθμό θα διατηρήσουν την τιμή 1. Τα υπόλοιπα θα αποκτήσουν σταδιακά την τιμή 0.
  - Ξεκινώντας από τον αριθμοδείκτη 2 (θεωρώντας πρώτο αριθμοδείκτη το 1 και όχι το 0), κάθε φορά που βρίσκεται ένα στοιχείο του πίνακα με τιμή 1 (έστω ότι αντιστοιχεί στον αριθμοδείκτη *k*), ενεργοποιείται ένας βρόχος, κατά τη διάρκεια του οποίου μηδενίζονται όλα τα στοιχεία του πίνακα που έχουν αριθμοδείκτες πολλαπλάσιους του *k*. Π.χ. για τον αριθμοδείκτη 2 θα μηδενιστούν οι θέσεις του πίνακα με αριθμοδείκτες 4,6,8,10 κ.λ.π., ενώ για τον αριθμοδείκτη 3 θα μηδενιστούν οι θέσεις 6,9,12,15 κ.λ.π. Όταν ολοκληρωθεί η διαδικασία, μη μηδενικές τιμές θα έχουν μόνο τα στοιχεία που βρίσκονται σε θέσεις με αριθμοδείκτες πρώτους αριθμούς.

### Άσκηση 3

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Μέσω επαναληπτικής πρότασης θα λαμβάνει από το πληκτρολόγιο τις τιμές ενός πίνακα τεσσάρων αλφαριθμητικών `arr_char[4][26]`. Τα αλφαριθμητικά θα δύνανται να περιέχουν τον χαρακτήρα του κενού. Ακολουθώς, θα εμφανίζει τον προκύπτοντα πίνακα στην οθόνη.
- Θα σαρώνει τον πίνακα ελέγχοντας εάν σε κάθε αλφαριθμητικό εμφανίζεται τουλάχιστον 3 φορές ο χαρακτήρας 'D'. Σε κάθε επιτυχή έλεγχο, θα αυξάνεται κατάλληλος μετρητής. Στο τέλος της σάρωσης του πίνακα θα εμφανίζονται στην οθόνη η τιμή του μετρητή επιτυχών ελέγχων και οι θέσεις στον πίνακα `arr_char` όπου εμφανίστηκε τουλάχιστον 3 φορές ο χαρακτήρας 'D'.

#### Άσκηση 4

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Μέσω επαναληπτικής πρότασης **do-while** θα δέχεται από το πληκτρολόγιο τρία αλφαριθμητικά και θα τα αποθηκεύει σε πίνακα αλφαριθμητικών **str[3][21]**. Ακολούθως, θα εμφανίζει στην οθόνη τα περιεχόμενα του πίνακα.
- Θα ελέγχει εάν οι χαρακτήρες που βρίσκονται στην πρώτη, δεύτερη και τρίτη θέση του του πρώτου αλφαριθμητικού είναι ίδιοι με τους χαρακτήρες που βρίσκονται στις αντίστοιχες θέσεις του δεύτερου αλφαριθμητικού (ελέγχοντας πρώτα εάν τα αναγνωσθέντα αλφαριθμητικά έχουν μήκος μεγαλύτερο ή ίσο του 3). Σε περίπτωση ισότητας θα αντιγράφει στην τρίτη γραμμή του πίνακα τους 3 αυτούς χαρακτήρες και θα τυπώνει το νέο πίνακα χαρακτήρων στην οθόνη ως αλφαριθμητικό.

#### Άσκηση 5

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Θα λαμβάνει από το πληκτρολόγιο τις τιμές ενός πίνακα δύο αλφαριθμητικών **arr\_char[2][15]** και θα εμφανίζει τον προκύπτοντα πίνακα στην οθόνη.
- Θα δημιουργεί στην πρώτη γραμμή του πίνακα **arr\_char** (δηλαδή στην **arr\_char[0]**) το αλφαριθμητικό που θα προκύψει αποκλειστικά από τους χαρακτήρες του **arr\_char[1]**, οι οποίοι βρίσκονται στις περιττές θέσεις (υπενθυμίζεται ότι η πρώτη θέση πίνακα στη γλώσσα C είναι άρτια, ήτοι η θέση 0).

#### Άσκηση 6

Να γραφεί πρόγραμμα, το οποίο θα επιτελεί τα ακόλουθα:

- Θα λαμβάνει από το πληκτρολόγιο τις τιμές ενός πίνακα ακέραιων αριθμών **arr\_int[3][3]** και θα εμφανίζει τον προκύπτοντα πίνακα στην οθόνη.
- Θα υπολογίζει σε κάθε γραμμή του πίνακα **arr\_int** το στοιχείο με την ελάχιστη απόλυτη τιμή. Για κάθε γραμμή θα απεικονίζονται στην οθόνη η απόλυτη τιμή του ελάχιστου στοιχείου και η στήλη στην οποία βρίσκεται.
- Θα υπολογίζει το γινόμενο των τιμών των στοιχείων της δεύτερης στήλης του πίνακα **arr\_int**.

#### Άσκηση 7

Να γραφεί πρόγραμμα, με το οποίο θα εισάγονται 9 ακέραιοι αριθμοί σε τετραγωνικό πίνακα **array[3][3]**. Ακολούθως, θα καλείται η συνάρτηση **float aver(int arr[3][3], int size)**, η οποία θα επιστρέφει στη **main()** τον μέσο όρο των τιμών των στοιχείων του πίνακα, ο οποίος και θα εμφανίζεται στην οθόνη.

## Βιβλιογραφία κεφαλαίου

- Θραμπουλίδης, Κ. (2002), *Διαδικαστικός Προγραμματισμός - C (Τόμος Α)*, 2<sup>η</sup> έκδοση, Εκδόσεις Τζιόλα.
- Καράκος, Αλ. (2010), *Αλγοριθμική Επίλυση Ασκήσεων με τη Γλώσσα Προγραμματισμού C*.
- Deitel, H. & Deitel, P. (2014), *C Προγραμματισμός*, 7<sup>η</sup> έκδοση, Εκδόσεις Γκιούρδα.
- Deitel, H. & Deitel, P. (2005), *Ασκήσεις - Προγράμματα σε C*, Εκδόσεις Γκιούρδα.
- Kelley, A. & Pohl, I. (1998), *A Book on C*, 4<sup>th</sup> ed, Addison-Wesley.
- King, K. (2008), *C Programming: A Modern Approach*, 2<sup>nd</sup> ed., W.W. Norton & Company.
- Prinz, P. & Crawford, T. (2005), *C in a Nutshell*, O'Reilly.
- Roberts, E. (2008), *Η Τέχνη και Επιστήμη της C*, Εκδόσεις Κλειδάριθμος.
- Waite, M., Prata, S. & Martin, D. (2000), *Πλήρης Οδηγός Χρήσης της C*, 6<sup>η</sup> έκδοση, Εκδόσεις Γκιούρδα.