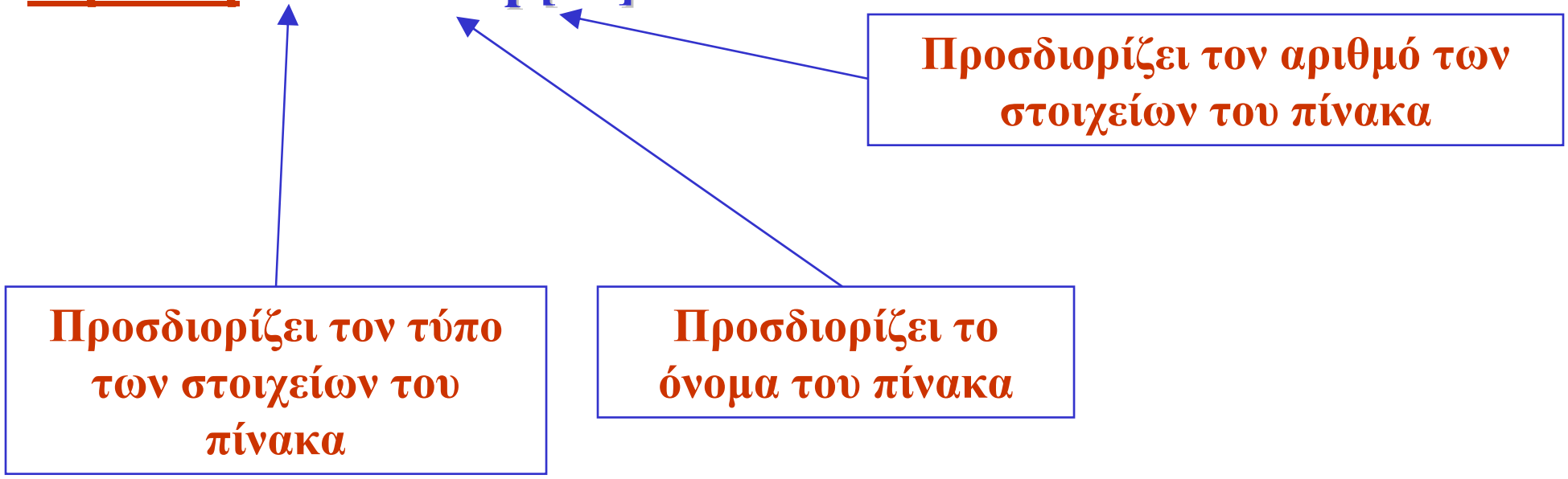
Θεματική ενότητα 6:
Πίνακες – αλφαριθμητικά

Πίνακες

- **Ο πίνακας είναι μία συλλογή μεταβλητών ίδιου τύπου, οι οποίες είναι αποθηκευμένες σε διαδοχικές θέσεις μνήμης. Χρησιμοποιείται για την αποθήκευση και διαχείριση δεδομένων κοινού τύπου και αποτελεί, μαζί με τους δείκτες, από τα πλέον ισχυρά εργαλεία της γλώσσας C.**
- **Δήλωση: float temp[31]**



- **Η δήλωση γνωστοποιεί στο μεταγλωττιστή ότι δηλώνεται ο πίνακας temp, ο οποίος περιέχει δεδομένα τύπου float και καταλαμβάνει 31 θέσεις μνήμης για την αποθήκευση των στοιχείων του.**
- **Αναφορά σε στοιχείο πίνακα: γίνεται με συνδυασμό του ονόματος και ενός δείκτη που εκφράζει τη σειρά του στοιχείου μέσα στον πίνακα:**

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

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

  temp[30]: ***τελευταίο (τριακοστό πρώτο) στοιχείο του πίνακα***

- **Απόδοση αρχικής τιμής: κατά τη δήλωση του πίνακα, γίνεται με χρήση του τελεστή ανάθεσης ως εξής:**

float temp[5] = {1,2,-4.2,6,8}: αρχικοποιούνται και τα 5 στοιχεία του πίνακα temp.

float temp[5] = {1,2,-4.2}: αρχικοποιούνται τα 3 πρώτα στοιχεία του πίνακα temp, δηλαδή τα temp[0], temp[1], temp[2].

•**Παρατήρηση: Όταν αποδίδονται αρχικές τιμές μπορεί να παραληφθεί το μέγεθος του πίνακα. Ο υπολογιστής θα υπολογίσει αυτόματα πόσα είναι τα στοιχεία του πίνακα από τον αριθμό των αρχικών τιμών που δίδονται. Η παρακάτω δήλωση**

```
char d_ar[] = {‘a’, ‘b’, ‘c’, ‘d’};
```

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

```
d_ar[0] = ‘a’
d_ar[1] = ‘b’
d_ar[2] = ‘c’
d_ar[3] = ‘d’
```

- **Το γεγονός ότι η δείκτες των στοιχείων ενός πίνακα ξεκινούν από το 0 κι όχι από το 1 μπορεί αρχικά να προκαλέσει σύγχυση αλλά αντανακλά τη φιλοσοφία της C, η οποία επιδιώκει να παραμείνει ο προγραμματισμός κοντά στην αρχιτεκτονική του υπολογιστή. Το 0 αποτελεί το σημείο εκκίνησης για τους υπολογιστές. Εάν η αρίθμηση των στοιχείων πίνακα ξεκινούσε από το 1, όπως π.χ. FORTRAN, ο μεταγλωττιστής θα έπρεπε να αφαιρέσει 1 από κάθε αναφορά σε δείκτη στοιχείου για να ληφθεί η πραγματική διεύθυνση ενός στοιχείου. Επομένως, η επιλογή της C παράγει πιο αποτελεσματικό κώδικα.**

- **Προσοχή: Υπάρχει διαφορά ανάμεσα στη *δήλωση πίνακα* και στην *αναφορά στοιχείου σε πίνακα*. Σε μία δήλωση, ο δείκτης καθορίζει το μέγεθος του πίνακα. Σε μία αναφορά στοιχείου πίνακα, ο δείκτης προσδιορίζει το στοιχείο του πίνακα στο οποίο αναφερόμαστε. Π.χ. στη δήλωση `int temp[31];` το 31 δηλώνει τον αριθμό των στοιχείων του πίνακα. Αντίθετα στη `temp[13]=21;` το 13 δηλώνει το συγκεκριμένο στοιχείο (14ο) του πίνακα, στο οποίο αναφερόμαστε και αποδίδουμε την τιμή 21.**

Πολυδιάστατοι πίνακες

- *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 C δε θέτει περιορισμό στον αριθμό των διαστάσεων των πινάκων.**
- **Αν και ο πολυδιάστατος πίνακας αποθηκεύεται στη μνήμη ως μία ακολουθία στοιχείων μίας διάστασης, μπορούμε να το θεωρούμε ως πίνακα πινάκων. Για παράδειγμα, έστω το επόμενο «μαγικό τετράγωνο», του οποίου οι γραμμές οριζόντια, κάθετα και διαγώνια δίνουν το ίδιο άθροισμα:**

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Για να αποθηκευθεί το τετράγωνο αυτό σε πίνακα θα μπορούσαμε να κάνουμε την ακόλουθη δήλωση:

```
int magic[5][5]={ {17, 24, 1, 8, 15},
                  {23, 5, 7, 14, 16},
                  {4, 6, 13, 20, 22},
                  {10, 12, 19, 21, 3},
                  {11, 18, 25, 2, 9}
                };
```

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

•Για την αναφορά σε στοιχείο ενός πολυδιάστατου πίνακα θα πρέπει να καθορισθούν τόσοι δείκτες όσοι είναι αναγκαίοι. Έτσι, η έκφραση magic[1] αναφέρεται στη δεύτερη γραμμή του πίνακα, ενώ η έκφραση magic[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 ar[2][3]={ {0, 1, 2},
               {3, 4, 5}
             };
```

αποθηκεύεται όπως φαίνεται ακολούθως:

Στοιχείο	Διεύθυνση Μνήμη	Περιεχόμενα
ar[0] [0]	1000	0
ar[0] [1]	1004	1
ar [0] [2]	1008	2
ar[1] [0]	100C	3
ar[1] [1]	1010	4
ar[1] [2]	1014	5
	1018	

Αρχικοποίηση πολυδιάστατου πίνακα

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

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

η μεταβλητή ar δηλώνεται ως πίνακας 5 γραμμών και 3 στηλών. Ωστόσο έχουν αποδοθεί τιμές μόνο για τις 3 πρώτες γραμμές του πίνακα και μάλιστα για τη δεύτερη γραμμή έχει ορισθεί η τιμή μόνο του πρώτου στοιχείου. Η παραπάνω δήλωση έχει ως αποτέλεσμα τη δημιουργία του ακόλουθου πίνακα:

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

Αν δε συμπεριληφθούν τα ενδιάμεσα άγκιστρα:

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

το αποτέλεσμα είναι ο ακόλουθος πίνακας, που είναι σαφώς διαφορετικός, <u>οπότε δημιουργείται πρόβλημα</u>:

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

Προσοχή: **Όπως και με τους πίνακες μίας διάστασης, έτσι και στους πολυδιάστατους πίνακες εαν 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. Η παρακάτω δήλωση αξιοποιεί τη δυνατότητα αυτή του μεταγλωττιστή:**

```
int ar[][3][2]={ {{1, 1}, {0, 0}, {1, 1}},
                 {{0, 0}, {1, 2}, {0, 1}}
               };
```

Με την παραπάνω δήλωση ο ar δηλώνεται αυτόματα ως πίνακας 2x3x2.

- **Η παρακάτω 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 το θεωρήσει είτε πίνακα 2x3 είτε 3x2.

- **Η παρακάτω δήλωση:**

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

είναι λανθασμένη στη γλώσσα C αλλά δεν εντοπίζεται από το μεταγλωττιστή και οδηγεί σε ανεπιθύμητα αποτελέσματα. Η σωστή είναι

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

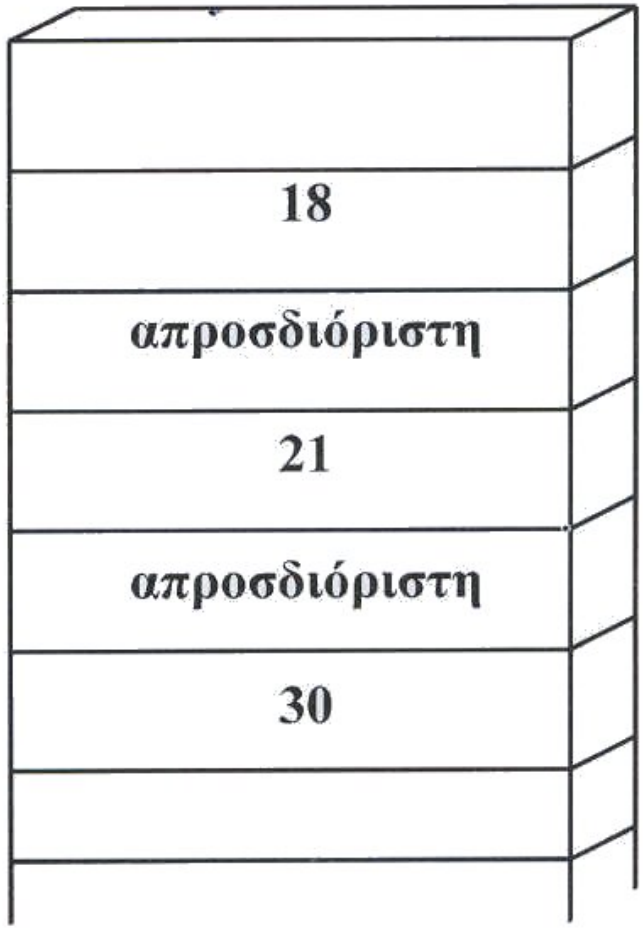
Αποθήκευση των πινάκων στη μνήμη

Η πρόταση δήλωσης int ar[5]; έχει ως αποτέλεσμα τη δέσμευση χώρου στη μνήμη για την αποθήκευση 5 μεταβλητών ακέραιου τύπου. Έστω ότι μέσα στον κώδικα του προγράμματος υπάρχουν οι παρακάτω προτάσεις ανάθεσης, που αποδίδουν τιμές σε στοιχεία του πίνακα:

```
ar[0] = 18;
ar[2] = 21;
ar[4] = ar[0] + 12;
```

Στο σχήμα που ακολουθεί φαίνεται η μορφή που έχει η μνήμη που έχει δεσμευτεί για την αποθήκευση του πίνακα ar , θεωρώντας 32 bits για κάθε ακέραιο και πως ο υπολογιστής αρχίζει στη διεύθυνση 1000.

Στοιχείο	Διεύθυνση	Περιεχόμενα (4 bytes)
	0FFC	
ar[0]	1000	**18**
ar[1]	1004	**απροσδιόριστη**
ar[2]	1008	**21**
ar[3]	100C	**απροσδιόριστη**
ar[4]	1010	**30**
	1014	



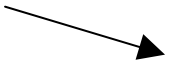
•Γίνεται αντιληπτό ότι τα ar[1] και ar[3] έχουν απροσδιόριστες τιμές. Τα περιεχόμενα αυτών των θέσεων μνήμης είναι ο,τιδήποτε έχει μείνει στις θέσεις αυτές από προηγούμενη αποθήκευση. Οι απροσδιόριστες τιμές αναφέρονται ως «σκουπίδια» (junk) και πολλές φορές αποτελούν αιτία πρόκλησης λαθών.

•Για την αποφυγή προβλημάτων αυτής της μορφής πρέπει να αποδίδονται αρχικές τιμές στους πίνακες ή να υπάρχει συνεχής αντίληψη του τι περιλαμβάνει ένας πίνακας. Θα πρέπει να σημειωθεί ότι η ANSI C διασφαλίζει ότι οι καθολικές μεταβλητές (αρχικοποιούνται από το σύστημα με την τιμή 0. Αυτό συμβαίνει βέβαια και για όλα τα στοιχεία πίνακα που έχει δηλωθεί ως καθολική μεταβλητή.

Παράδειγμα: **Θεωρούμε πίνακα που αναπαριστά τις μέσες θερμοκρασίες των μηνών των τελευταίων τριών ετών. Να δοθούν: *(α)* βρόχος για τον υπολογισμό των μέσων ετήσιων θερμοκρασιών των τριών ετών, *(β)* βρόχος για τον υπολογισμό των μέσων μηνιαίων θερμοκρασιών των τριών ετών.**

(α)

```
#define YEARS 3
#define MONTHS 12
int year, month;
float subtotal, temp[YEARS][MONTHS], avg_temp[YEARS];
for (year=0;year<YEARS;year++)  {   //ο temp θεωρείται αρχικοποιημένος
  subtotal=0.0;
  for (month=0;month<MONTHS;month++)
            subtotal+=temp[year][month];
  avg_temp[year]=subtotal/MONTHS;
}
```



(β)

```
#define YEARS 3
#define MONTHS 12
int year, month;
float subtotal, temp[YEARS][MONTHS], avg_temp[MONTHS];
//ο temp θεωρείται αρχικοποιημένος
for (month=0;month<MONTHS;month++)
{
  subtotal=0.0;
  for (year=0;year<YEARS;year++)
        subtotal+=temp[year][month];
  avg_temp[month]=subtotal/YEARS;
}
```

Διαφορετικό από την (α) περίπτωση

Γενικευμένο παράδειγμα πράξεων με πίνακες:

ch_6_arrays.c

Το αλφαριθμητικό (string)

- **Πίνακας χαρακτήρων που τερματίζει με το μηδενικό (*null*) χαρακτήρα. Ο μηδενικός χαρακτήρας έχει ASCII κωδικό 0 και αναπαρίσταται από την ακολουθία διαφυγής \0.**
- **Δήλωση: char book_title[30]**

Αλφαριθμητική σταθερά

Τα αλφαριθμητικά μπορούν να εμφανίζονται μέσα στον κώδικα όπως οι αριθμητικές σταθερές, αποτελώντας τις αλφαριθμητικές σταθερές. Η αλφαριθμητική σταθερά απαντήθηκε σε προηγούμενο στάδιο, όπου και σημειώθηκε ότι οι χαρακτήρες της περικλείονται σε διπλά εισαγωγικά. Για την αποθήκευσή της χρησιμοποιείται ένας πίνακας χαρακτήρων, με το μεταγλωττιστή να θέτει αυτόματα στο τέλος του αλφαριθμητικού ένα μηδενικό χαρακτήρα για να προσδιορίσει το τέλος του. Έτσι, η αλφαριθμητική σταθερά "Hello" απαιτεί για αποθήκευση 6 bytes, όπως φαίνεται παρακάτω:

'H' 'e' 'l' 'l' 'o' '\0'

Προσοχή: Σημειώστε τη διαφορά ανάμεσα στη σταθερά χαρακτήρα 'A' και την αλφαριθμητική σταθερά "A". Η πρώτη απαιτεί 1 byte για αποθήκευση, ενώ η δεύτερη ένα για το χαρακτήρα A κι ένα για το null.

Απόδοση αρχικής τιμής σε αλφαριθμητικό

Η ανάθεση τιμής με τη δήλωση ακολουθεί το γενικό κανόνα απόδοσης αρχικής τιμής σε πίνακα. Γράφουμε

```
char isbn[] = {'0','-','4','9','-','7','4','3','-','3','\0'};
```

Στην πράξη χρησιμοποιείται η εναλλακτική και πιο συμπαγής μορφή με χρήση αλφαριθμητικής σταθεράς:

```
char isbn[] = "0-49-743-3";
```

Θα πρέπει να προσεχθεί ότι στη δήλωση με λίστα στοιχείων ο προγραμματιστής πρέπει να περιλάβει ως τελευταία τιμή το* null*. Στο δεύτερο τρόπο απόδοσης αρχικής τιμής, αυτή την εργασία την εκτελεί αυτόματα ο μεταγλωττιστής.

Εισαγωγή αλφαριθμητικού

Η εισαγωγή αλφαριθμητικού από την κύρια είσοδο γίνεται με τη scanf() και με τον ίδιο προσδιοριστή που χρησιμοποιείται για την εκτύπωση. Η πρόταση

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

διαβάζει την κύρια είσοδο ως αλφαριθμητικό και αποθηκεύει την τιμή στη μεταβλητή isbn.

Δε χρειάζεται ο τελεστής & πριν από το όνομα της μεταβλητής isbn όπως συνέβαινε με τους άλλους τύπους δεδομένων, γιατί το όνομα του πίνακα αναπαριστά τη διεύθυνση του πρώτου στοιχείου του πίνακα.

Εισαγωγή αλφαριθμητικού

•Εναλλακτικά, η εισαγωγή αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης gets(), η γενική μορφή της οποίας είναι

gets(όνομα_πίνακα_χαρακτήρων)

•Καλείται η gets() με το όνομα του πίνακα χαρακτήρων ως όρισμα, χωρίς δείκτη. Με την επιστροφή από τη gets() το αλφαριθμητικό θα έχει περασθεί στον πίνακα χαρακτήρων. Η gets() θα διαβάζει χαρακτήρες από το πληκτρολόγιο έως ότου πατηθεί το ENTER.

•Η gets() μπορεί και διαβάζει κενά, η scanf() όχι.

Παρατηρήσεις στην εισαγωγή αλφαριθμητικού

•Το αλφαριθμητικό αποθηκεύεται σε έναν πίνακα χαρακτήρων μαζί με τον τελικό μηδενικό χαρακτήρα. Κατά συνέπεια, όταν δηλώνεται το μέγεθος του πίνακα, έστω N, σε αυτόν μπορεί να αποθηκευτεί αλφαριθμητικό μέγιστου μήκους N-1.

•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 isbn[30] και το αλφαριθμητικό είναι μεγαλύτερο από το μέγεθος του isbn, ο πίνακας θα ξεπερασθεί.

Εκτύπωση αλφαριθμητικού

•Η εκτύπωση αλφαριθμητικής σταθεράς γίνεται με την printf() χωρίς τη χρήση προσδιοριστή. Απλώς της περνάμε την προς εκτύπωση αλφαριθμητική σταθερά:

```
printf( “Hello” );
```

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

```
printf( “The ISBN code is: %s”, isbn );
```

θα έχει ως αποτέλεσμα να τυπωθεί στην οθόνη η πρόταση

```
The ISBN code is: 0-49-743-3
```

Εκτύπωση αλφαριθμητικού

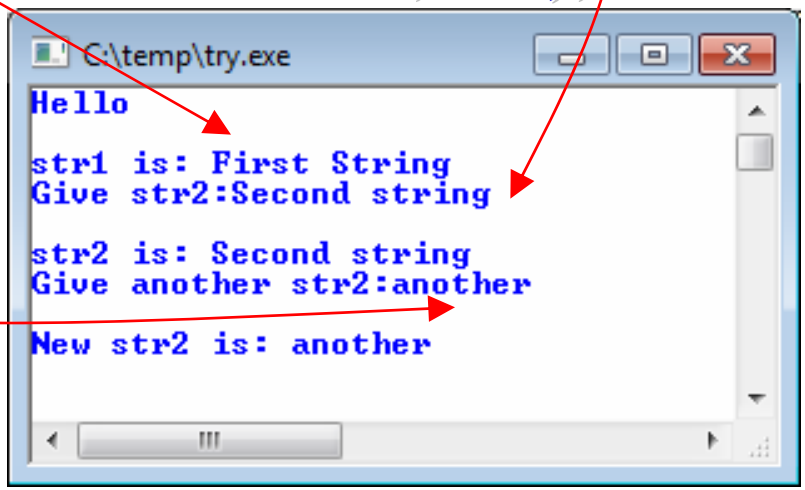
•Εναλλακτικά, η εκτύπωση αλφαριθμητικής σταθεράς και αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης puts(), η γενική μορφή της οποίας είναι

puts(όνομα_πίνακα_χαρακτήρων)

•Καλείται η puts() με το όνομα του πίνακα χαρακτήρων ως όρισμα, χωρίς δείκτη, π.χ. puts(isbn). Βέβαια, η puts() παρουσιάζει το μειονέκτημα ότι δεν παρέχει δυνατότητες μορφοποίησης της εξόδου.

Παράδειγμα: Εισαγωγή και εκτύπωση αλφαριθμητικών

```
#include <stdio.h>
#define STA "Hello"
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);
}
```



Συναρτήσεις αλφαριθμητικών
#include <string.h>

- *Εύρεση μήκους string*
- *Αντιγραφή string*
- *Συνένωση 2 strings*
- *Σύγκριση 2 strings*
- *Εύρεση χαρ. σε string*
- *Εύρεση string σε string*

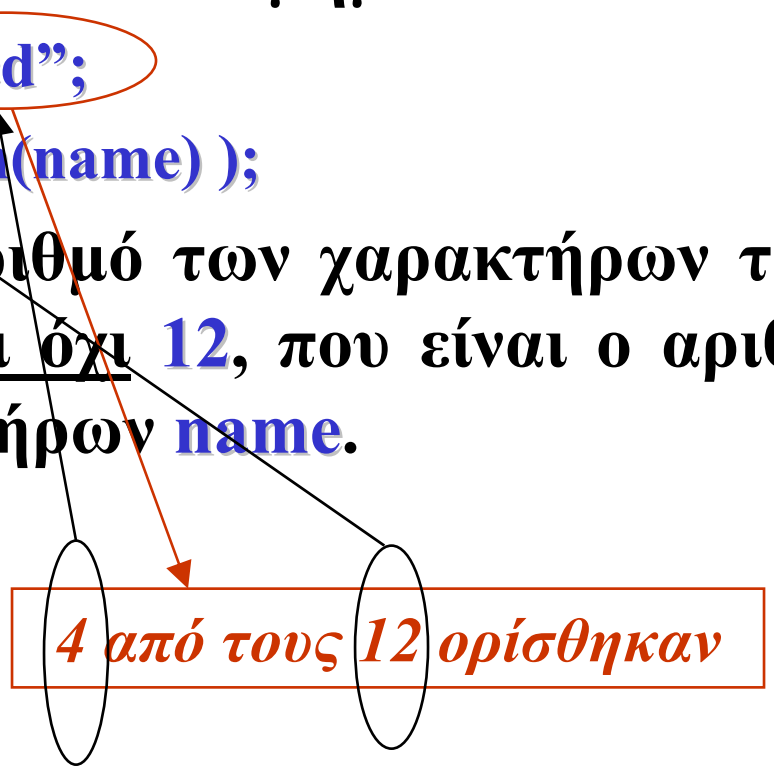
Όλοι οι χαρ.	Οι πρώτοι *n* χαρ.
`strlen()`	
`strcpy()`	`strncpy()`
`strcat()`	`strncat()`
`strcmp()`	`strncmp()`
`strchr()`	`strrchr()`
`strstr()`	

Η συνάρτηση μήκους αλφαριθμητικού (strlen)

Η συνάρτηση επιστρέφει τον αριθμό χαρακτήρων του αλφαριθμητικού, χωρίς να συμπεριλαμβάνει το μηδενικό χαρακτήρα. Το παρακάτω τμήμα κώδικα

```
char name[12] = "abcd";
printf( "%d\n", strlen(name) );
```

θα τυπώσει τον αριθμό των χαρακτήρων του αλφαριθμητικού name, δηλαδή 4 κι όχι 12, που είναι ο αριθμός των στοιχείων του πίνακα χαρακτήρων name.



Παράδειγμα:

```
#include <stdio.h>
#include <string.h>          /* για τη strlen() */

void main() {
   char msg[20] = {"Hello world!"};
    int cnt;
    cnt = strlen(msg);
    printf("length of\"%s\" is: %d\n",msg,cnt);
 }
```

Αποτέλεσμα:

```
> length of "Hello world!" is: 12
>
```

Σημείωση : ο χαρακτήρας null δεν προσμετράται

Υλοποίηση της strlen() και εύρεση χαρακτήρων σε συμβολοσειρές:

ch_6_example_strlen.c

Η συνάρτηση αντιγραφής αλφαριθμητικών (strcpy)

Η συνάρτηση αντιγράφει ένα αλφαριθμητικό από έναν πίνακα σε έναν άλλο. Δέχεται δύο ορίσματα που είναι τα ονόματα (οι δείκτες) στα αλφαριθμητικά. *Το όνομα του πίνακα προορισμού πρέπει να είναι το πρώτο όρισμα, ενώ το δεύτερο προσδιορίζει τον πίνακα πηγής*. Στο παρακάτω τμήμα κώδικα

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

η πρόταση strcpy(name1,name2); αντιγράφει το περιεχόμενο του πίνακα name2 στον πίνακα name1. Έτσι στην οθόνη θα εμφανισθεί το ef.

Αντιγραφή αριθμού χαρακτήρων: strncpy(destination,source,number)

```
#include <stdio.h>
#include <string.h>          /* για την strncpy()*/

void main()  {
   char msg[20];

   strncpy(msg,"Hello world!", 4);
   printf("my string:%s\n",msg);

}
```

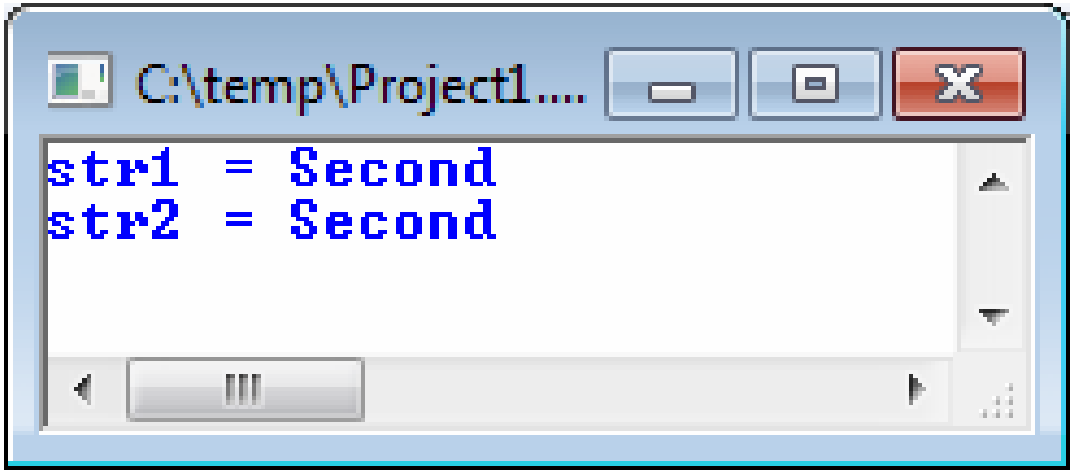
Αντιγράφει μόνο τους πρώτους 4 chars

Αποτέλεσμα:

> my string: Hell

Παράδειγμα:

```
#include <stdio.h>
#include <string.h>
main()  {
  int i;
  char str1[80]="This is the first string", str2[ ]="Second";
  if ((strlen(str2)+1)>sizeof(str1))
      printf("Error!!! str2 exceeds str1's dimension\n\n");
  else  {
    for (i=0;i<=strlen(str2);i++)
        str1[i] = str2[i];
  }
  printf("str1 = %s\n",str1);
  printf("str2 = %s\n",str2);
}
```



Η συνάρτηση συνένωσης αλφαριθμητικών (strcat)

Η συνάρτηση δέχεται δύο ορίσματα που είναι τα ονόματα (οι δείκτες) στα αλφαριθμητικά, 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 name1[12] = “abcd”;
char name2[12] = “ef”;
strcat(name1,name2);
printf( “%sn”, name1 );
```

προσθέτει στο τέλος του name1 το περιεχόμενο του πίνακα name2. Έτσι στην οθόνη θα εμφανισθεί το abcdef.

***Είναι ευθύνη του προγραμματιστή να έχει ο πίνακας* name1 *αρκετά στοιχεία ώστε να «χωρούν» και τα στοιχεία του* name2.**

Η συνάρτηση: strncat(string1,string2,number)

```
#include <stdio.h>
#include <string.h>   /* για την strncat()*/
  void main() {
     char msg1[81],msg2[81];

     strcpy(msg1,"Hello you!");
     strcpy(msg2,"Hello me!");
        strncat(msg1,msg2,4);
        printf("%s\n", msg1);

  }
```

Η msg1 είναι ΤΟΣΟ είσοδος ΟΣΟ ΚΑΙ έξοδος στην strncat()!

Αποτέλεσμα:

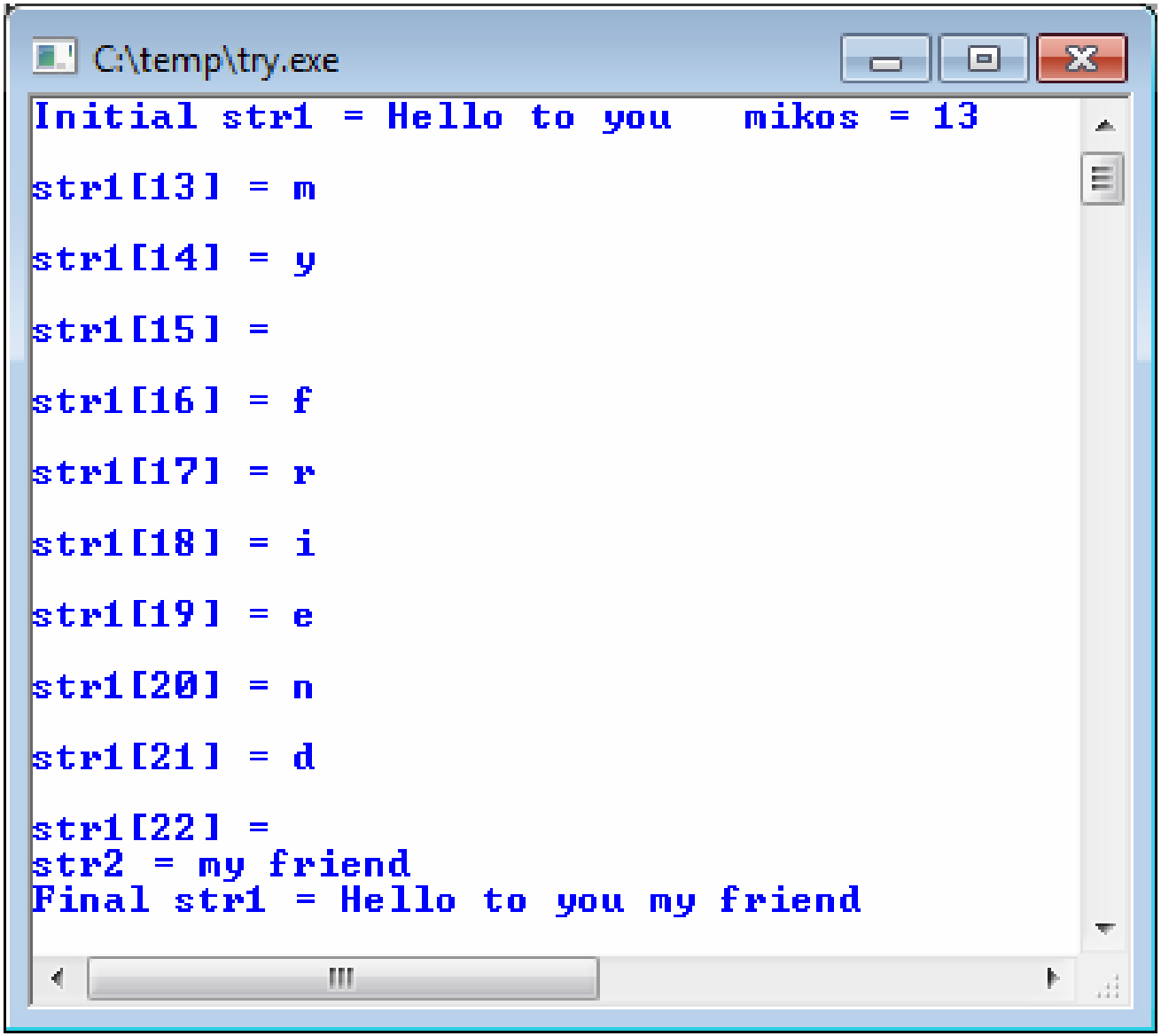
> Hello you!Hell

>

Παράδειγμα:

```
#include <stdio.h>
#include <string.h>
main()  {
  int i,mikos;
  char str1[80]="Hello to you ", str2[]="my friend";
  mikos = strlen(str1);
  printf("Initial str1 = %s  mikos = %d\n",str1,mikos);
  if ((strlen(str2)+1+strlen(str1))>sizeof(str1))
     printf("Error!!! total length exceeds str1's dimension\n\n");
  else  {
    for (i=0;i<=strlen(str2);i++)  {
      str1[mikos+i] = str2[i];
      printf("\nstr1[%d] = %c\n",mikos+i,str1[mikos+i]); }  // τέλος της for
  }  // τέλος της if-else
  printf("str2 = %s\n",str2);
  printf("Final str1 = %s\n",str1);  }  // τέλος της main
```

Αποτελέσματα:



Η συνάρτηση σύγκρισης αλφαριθμητικών int strcmp(a,b)

```
#include <stdio.h>
#include <string.h>        /* για την strcmp() */
void main() {
    char msg1[81] = {“Hello to you!”};
    char msg2[81] = {“Hello to me!”};
    int diff;
    diff = strcmp(msg1,msg2);
    if(diff==0) printf(“same!\n”);
    else printf(“different!\n”);
}
```

Αποτέλεσμα:

> different!

>

Η συνάρτηση: int strncmp(a,b,number)

```
#include <stdio.h>
#include <string.h>
void main() {
    char msg1[81] = {"Hello to you!"};
    char msg2[81] = {"Hello to me!"};
    int diff;
    diff = strncmp(msg1,msg2,6);
    if(0==diff) printf("same!\n");
    else printf("different!\n");
}
```

Συγκρίνει μόνο τους πρώτους 6 chars

Αποτέλεσμα:

```
> same!
>
```