

Συναρτήσεις αλφαριθμητικών `#include <string.h>`

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

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`.

Η χρήση της `strcpy()` αποτελεί τον τρόπο ανάθεσης 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");`

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

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

int main() {
    char msg[20];

    strncpy(msg, "Hello world!", 4);
    printf("my string:%s\n", msg);
    return 0;
}
```

Αντιγράφει μόνο τους
πρώτους 4 chars

Αποτέλεσμα:

> my string: Hell

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() */
int main() {
    char msg1[81],msg2[81];

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

    return 0; }
```

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

Αποτέλεσμα:

> Hello you!Hell

>

Παρατήρηση

Στις συναρτήσεις `strcpy()` και `strcat()` ελλοχεύει ο κίνδυνος να ξεπερασθούν τα όρια του πίνακα χαρακτήρων προορισμού, με αποτέλεσμα να δημιουργηθούν απροσδιόριστες καταστάσεις. Εάν π.χ. έχει ορισθεί ο πίνακας χαρακτήρων `char array[4]`, ο οποίος καταλαμβάνει τις θέσεις **1000** έως και **1003** στον χάρτη μνήμης του σχήματος της επόμενης διαφάνειας, και επιχειρηθεί να τοποθετηθεί σε αυτό το αλφαριθμητικό: `"character"` θα τεθεί θέμα απροσδιοριστίας. Συγκεκριμένα, στις θέσεις **1004-1009**, τις οποίες δε 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.



Παρατήρηση

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.

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

Η συνάρτηση σύγκρισης αλφαριθμητικών *int strcmp(a,b)*

```
#include <stdio.h>
#include <string.h>      /* για την strcmp() */
int 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");
    return 0; }
```

Αποτέλεσμα:

> different!

>

Η συνάρτηση: ***int strncmp(a,b,number)***

```
#include <stdio.h>
#include <string.h>
int 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");
    return 0; }
```

Συγκρίνει μόνο τους
πρώτους 6 **chars**

Αποτέλεσμα:

> same!

>

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.

Παράδειγμα μετατροπών

<pre>#include <stdio.h> #include <stdlib.h> int main() { int val1; long int val2; double val3; char str1[20]="123456",str2[20]="abcd"; char str3[20]=" 1234bn";</pre>	<pre>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);</pre>
---	--



Παράδειγμα μετατροπών

```
printf("\n\natoi example:\n");  
val2 = atoi(str1);  
printf( "Str1 = \"%s\"", Long integer ", str1 );  
printf( "value = %ld\n", val2 );  
val2 = atoi(str2);  
printf( "Str2 = \"%s\"", Long integer ", str2 );  
printf( "value = %ld\n", 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

Παράδειγμα ανάπτυξης προγράμματος (χωρίς συναρτήσεις)

Να γραφεί πρόγραμμα που να δέχεται ως είσοδο κείμενο, να απαριθμεί τις εμφανίσεις των ψηφίων 0-9, τα λευκά διαστήματα και τους υπόλοιπους χαρακτήρες και στη συνέχεια να τυπώνει τα αποτελέσματα. Το πρόγραμμα ολοκληρώνεται όταν δοθεί ο χαρακτήρας τελεία '.'.

Λύση:

Το παραπάνω πρόβλημα μπορεί να μορφοποιηθεί σε δομημένα Ελληνικά ως εξής:

Για κάθε χαρακτήρα του κειμένου που είναι διάφορος της τελείας '.'

έλεγε τον τύπο του χαρακτήρα

αν είναι ένας από τους ' ', '\t', '\n' 3

αύξησε τον απαριθμητή των διαστημάτων

αν είναι ψηφίο

αύξησε τον απαριθμητή που αντιστοιχεί στο ψηφίο

σε κάθε άλλη περίπτωση

αύξησε τον απαριθμητή των υπόλοιπων χαρακτήρων

Αναπαράσταση δεδομένων:

- Όσον αφορά τις μεταβλητές απαιτείται:
 - 1) Ένας απαριθμητής για τα διαστήματα, τον οποίο ονομάζουμε `n_white`.
 - 2) Ένας απαριθμητής για τους λοιπούς χαρακτήρες, ο `n_other`, και δέκα απαριθμητές για τα ψηφία.

Για την τελευταία περίπτωση μπορούμε να δηλώσουμε δέκα ανεξάρτητες μεταβλητές αλλά είναι προτιμότερο να επιλεχθεί ένας πίνακας δέκα θέσεων, όπως

```
int n_digit[10];
```

ο οποίος οδηγεί σε πιο συμπαγή και δομημένο κώδικα.

Αναπαράσταση δεδομένων:

- Για την εκτύπωση των αριθμών των εμφανίσεων των δέκα ψηφίων, στην περίπτωση του πίνακα απαριθμητών, ο αντίστοιχος κώδικας θα έχει τη μορφή

```
for (i=0;i<10;i++)  
    printf("Το ψηφίο %d εμφανίσθηκε \t %d \t  
φορές\n",i,n_digit[i]);
```

Αναλογιστείτε τον κώδικα για την περίπτωση 10 ανεξάρτητων μεταβλητών!!!

Αναπαράσταση διεργασιών:

- Για τη διεργασία **πάρε χαρακτήρα** χρησιμοποιείται η συνάρτηση **getchar()**, η οποία επιστρέφει τον χαρακτήρα που διαβάζει από την κύρια είσοδο. Αυτός ο χαρακτήρας πρέπει να αποθηκευθεί σε μία μεταβλητή τύπου χαρακτήρα για περαιτέρω επεξεργασία.

Τα παραπάνω οδηγούν στη δήλωση

```
char ch;
```

και στην έκφραση

```
ch=getchar();
```

η τιμή της οποίας είναι η τιμή του αριστερού τελεστέου της έκφρασης.

Αναπαράσταση διεργασιών:

- Για την ανίχνευση του τέλους του αρχείου χρησιμοποιούμε το συσχετιστικό τελεστή **!=** οδηγούμεστε στην έκφραση

`(ch=getchar())!='.'`

Η έκφραση γίνεται ψευδής όταν αναγνωσθεί **'.'**. Μπορεί επομένως να χρησιμοποιηθεί ως έκφραση μίας πρότασης **while**, που θα οδηγεί στην επανάληψη του συνόλου των ενεργειών που το πρόγραμμα πρέπει να εκτελεί για κάθε χαρακτήρα.

Διαμόρφωση της ροής ελέγχου:

Με βάση τα προηγούμενα, η περιγραφή διαμορφώνεται ως εξής:

```
while ((ch=getchar())!='.')
```

```
{
```

έλεγε τον τύπο του χαρακτήρα

αν είναι ένας από τους ' ', '\t', '\n'

αύξησε τον απαριθμητή των διαστημάτων

αν είναι ψηφίο

αύξησε τον απαριθμητή που αντιστοιχεί στο ψηφίο

σε κάθε άλλη περίπτωση

αύξησε τον απαριθμητή των υπόλοιπων χαρακτήρων

```
}
```

Το σώμα της **while** αποτελεί κλασική περίπτωση επιλογής από αμοιβαία αποκλειόμενες ενέργειες, γεγονός που οδηγεί στη χρήση της πρότασης **switch**. Η έκφραση ανάλογα με την τιμή της οποίας θα γίνει η επιλογή της κατάλληλης ενέργειας είναι η απλή έκφραση **ch**.

A) εάν είναι ένας από τους ' ', '\t', '\n'
αύξησε τον απαριθμητή `n_white`

ή εκφρασμένη στη C

```
case ' ':  
case '\t':  
case '\n':  
    n_white++;  
    break;
```

Κοινή έξοδος για τις τρεις `case`

B) εάν ο χαρακτήρας είναι ψηφίο
αύξησε τον απαριθμητή που αντιστοιχεί στο ψηφίο

μία πρώτη μορφή του κώδικα είναι η παρακάτω:

```
case '0':  
    n_digit[0]++; break;  
.....  
case '9':  
    n_digit[9]++; break;
```

- Ο κώδικας αυτός δεν εκμεταλλεύεται τη δήλωση των απαριθμητών των ψηφίων ως πίνακα χαρακτήρων. Για το λόγο αυτό, θα προσπαθήσουμε να ενοποιήσουμε τα **case** ώστε να έχουν μία παραμετρική πρόταση, που σε κάθε περίπτωση θα οδηγεί στην αύξηση του κατάλληλου απαριθμητή.

- Θεωρούμε την έκφραση

ch-'0'

και εξετάζουμε την τιμή της για τιμές της **ch** από το σύνολο **{'0', '1', ..., '9'}**. Είναι προφανές ότι, εάν το **ch** είναι **'0'**, η έκφραση έχει τιμή **0** οπότε και η πρόταση

n_digit[ch-'0']++;

έχει ως αποτέλεσμα την αύξηση του απαριθμητή που αντιστοιχεί στο ψηφίο **'0'**.

- Αντίστοιχα, η παραπάνω πρόταση για **ch='8'** θα αυξήσει τον απαριθμητή που αντιστοιχεί στο **'0'**. Κατά αυτόν τον τρόπο οδηγούμαστε στον ακόλουθο συμπαγή κώδικα:

case '0':

case '1':

...

...

case '9':

n_digit[ch-'0']++;

break;

Κοινή έξοδος για τις δέκα **case**

Γ) Για τα υπόλοιπα στοιχεία έχουμε την κλασική περίπτωση χρήσης της εντολής default, οπότε προκύπτει:

default:

n_other++;

break;

Το τελευταίο σημείο που πρέπει να προσεχθεί είναι η αρχικοποίηση των μεταβλητών.

Ενδεικτικός κώδικας:

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

```
int main() {
```

```
    char ch;
```

```
    int i,n_white=0,n_digit[10],n_other=0;
```

```
    for(i=0;i<10;i++) n_digit[i]=0;
```

```
    printf("Start writing:");
```

```
    while ((ch=getchar())!= '.') {
```

```
        switch(ch) {
```

```
            case ' ':
```

```
            case '\t':
```

```
            case '\n':
```

```
                n_white++;
```

```
                break;
```



```
case '0':    case '1':  
case '2':    case '3':  
case '4':    case '5':  
case '6':    case '7':  
case '8':    case '9':  
    n_digit[ch-'0']++;  
    break;  
default:  
    n_other++;  
    break; } //τέλος της switch  
} //τέλος της while  
printf("white characters=%d\n",n_white);  
for(i=0;i<10;i++) printf("Digit %d appeared %d times\n",i,n_digit[i]);  
printf("chars=%d \n",n_other);  
return 0; }
```



Αποτελέσματα:

Start writing: Year: 2020-2021

Month: 12

Week: 50

Day: 16

Time: 14:56:38

.

white characters=10

Digit 0 appeared 4 times

Digit 1 appeared 4 times

Digit 2 appeared 5 times

Digit 3 appeared 1 times

Digit 4 appeared 1 times

Digit 5 appeared 2 times

Digit 6 appeared 2 times

Digit 7 appeared 0 times

Digit 8 appeared 1 times

Digit 9 appeared 0 times

chars=28

0 \t, 5 \n, 5
κενά

Αρθρωτός σχεδιασμός

Στόχος του Διαδικαστικού Προγραμματισμού είναι ο αρθρωτός σχεδιασμός, δηλαδή ο μερισμός του συνολικού προβλήματος σε υποπροβλήματα και η επίλυση του καθενός εξ αυτών με αυτόνομες μονάδες κώδικα. Οι συναρτήσεις περιέχουν τα εργαλεία για την υλοποίηση του στόχου αυτού, επιδιώκοντας ταυτόχρονα την επαναχρησιμοποίηση υφιστάμενου κώδικα. Τα δύο παραδείγματα που ακολουθούν χρησιμοποιούνται ως πρωτόλεια υποδείγματα αρθρωτού σχεδιασμού. Απώτερος στόχος της συγκεκριμένης πρακτικής είναι να καταστεί η συνάρτηση **main()** συντονιστική συνάρτηση, η οποία θα κατανέμει το προς εκτέλεση έργο στα υπόλοιπα τμήματα κώδικα.

Παράδειγμα αρθρωτού σχεδιασμού

Να γραφεί πρόγραμμα, το οποίο θα διαβάζει χαρακτήρες από το πληκτρολόγιο, θα τους εμφανίζει στην οθόνη και θα τυπώνει το πλήθος των προτάσεων, το πλήθος των λέξεων και το πλήθος των χαρακτήρων του κειμένου. Η ανάγνωση χαρακτήρων θα περατώνεται όταν δοθεί ο χαρακτήρας του δολαρίου '\$'.

- Μία πρόταση ολοκληρώνεται όταν αναγνωσθεί ένας εκ των χαρακτήρων '.' ή ';' ή '!'.
ή ' ' ή ' '.
- Μία λέξη ολοκληρώνεται όταν αναγνωσθεί ένας εκ των χαρακτήρων '.' ή ';' ή '!'
ή ' ' ή ' '.

Σημείωση: Θεωρείται ότι δεν υπάρχουν διαδοχικές εμφανίσεις των χαρακτήρων
'.' ή ';' ή '!' ή ' ' ή ' '.

Παράδειγμα αρθρωτού σχεδιασμού

Με βάση τις προδιαγραφές, το πρόβλημα μπορεί να μερισθεί στα ακόλουθα υποπροβλήματα:

- Ανάγνωση χαρακτήρων μέσω επαναληπτικής πρότασης.
- Έλεγχος της φύσης του χαρακτήρα και συνακόλουθος χαρακτηρισμός του ως δηλωτικός του **τέλους λέξης** και του **τέλους πρότασης**.
- Εμφάνιση των αποτελεσμάτων στην οθόνη.

Η ανάγνωση των χαρακτήρων θα γίνεται μέσα στη **main()**. Επιπλέον από την κύρια συνάρτηση θα καλούνται οι ακόλουθες συναρτήσεις:

(α) **int endofSentence(char ch)**, η οποία θα δέχεται ως όρισμα ένα χαρακτήρα και θα επιστρέφει **1** αν ο χαρακτήρας είναι δηλωτικός του τέλους πρότασης, αλλιώς θα επιστρέφει **0**.

(β) **int endofWord(char ch)**, η οποία θα δέχεται ως όρισμα ένα χαρακτήρα και θα επιστρέφει **1** αν ο χαρακτήρας είναι δηλωτικός του τέλους λέξης, αλλιώς θα επιστρέφει **0**.

(γ) **void displayResults(int s, int w, int c)**, η οποία θα δέχεται τα πλήθη των προτάσεων, λέξεων και χαρακτήρων και θα τα εμφανίζει στην οθόνη.



Παράδειγμα αρθρωτού σχεδιασμού

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

```
int endofSentence(char ch);
```

```
int endofWord(char ch);
```

```
void displayResults(int s, int w, int c);
```

```
int main() {
```

```
    int s=0,w=0,c=0;
```

```
    char ch;
```

```
    while ((ch=getchar())!='$')
```

```
    {
```

```
        putchar(ch);
```

```
        s=s+endofSentence(ch);
```

```
        w=w+endofWord(ch);
```

```
        c++;
```

```
    }
```

```
    displayResults(s,w,c);
```

```
    return 0;
```

```
}
```



Παράδειγμα αρθρωτού σχεδιασμού

```
int endofSentence(char ch)
```

```
{
```

```
    if ((ch=='.') || (ch==';') || (ch=='!')) return(1);
```

```
    else return(0);
```

```
}
```

```
int endofWord(char ch)
```

```
{
```

```
    if ((ch=='.') || (ch==';') || (ch=='!') || (ch==' ') || (ch==',')) return(1);
```

```
    else return(0);
```

```
}
```



Παράδειγμα αρθρωτού σχεδιασμού

```
void displayResults(int s, int w, int c)
{
    printf( "\n\nNumber of sentences:%d\nNumber of words:%d\n",s,w );
    printf( "Number of characters:%d",c );
}
```

Type a number of characters!

Press USD to finish.

\$

Number of sentences: 2

Number of words:9

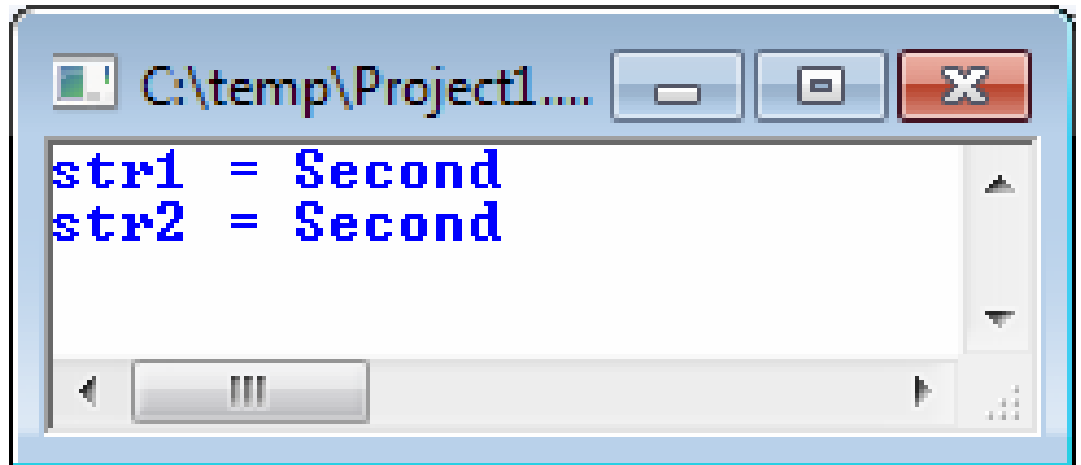
Number of characters:50

***(α) Επιπρόσθετα παραδείγματα πινάκων και
διαχείρισης αλφαριθμητικών***

(β) Σύνθετα παραδείγματα

1.

```
#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);
}
```



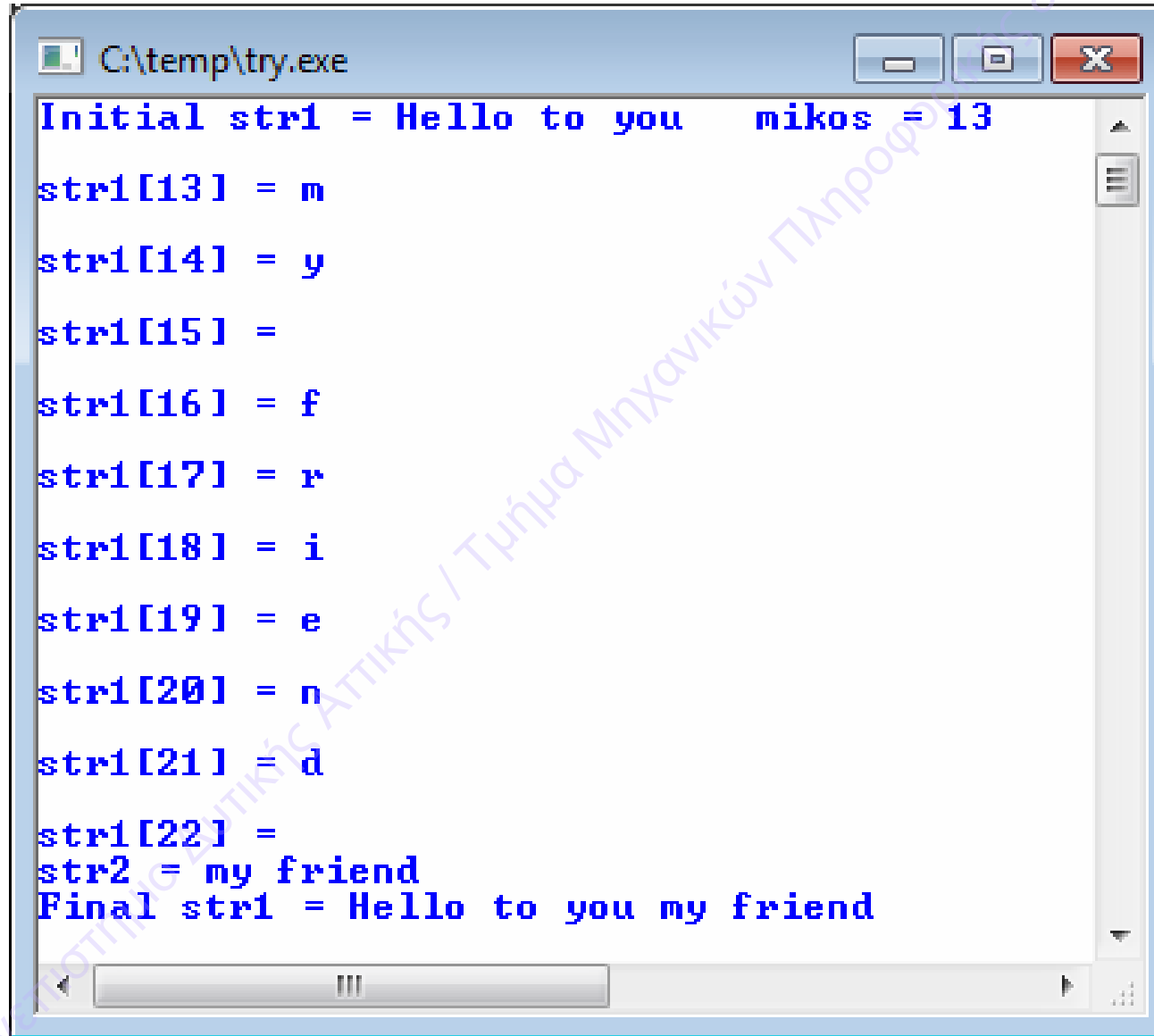
```
str1 = Second
str2 = Second
```


2.

```
#include <stdio.h>
#include <string.h>
int 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); return 0; } // τέλος της main
```



Αποτελέσματα:



```
Initial str1 = Hello to you   mikos = 13
str1[13] = m
str1[14] = y
str1[15] = 
str1[16] = f
str1[17] = r
str1[18] = i
str1[19] = e
str1[20] = n
str1[21] = d
str1[22] = 
str2 = my friend
Final str1 = Hello to you my friend
```

3.

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

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

```
#include <math.h>
```

```
#define N 6
```

```
#define lower 1.05
```

```
#define upper 50.8
```

```
#define MY_ZERO 0.000001
```

```
int main() {
```

```
    float array[N],maxim;
```

```
    int i,count=0;
```

```
    for (i=0;i<N;i++) {
```

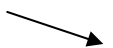
```
        printf( "\nGive number %d: ",i+1 );
```

```
        scanf( "%f",&array[i] );
```

```
}
```



```
maxim=array[0];
printf( "\n" );
for (i=0;i<N;i++)    // i=0 για να μπουν όλα σε ένα βρόχο,
                    // μόνο για το μέγιστο θα ήταν i=1
{
    if (array[i]>MY_ZERO) printf( "array[%d]>0: %.4f\n",i,array[i] );
    if (array[i]>maxim) maxim=array[i];
    if ((array[i]>=lower) && (array[i]<=upper)) count++;
}
printf( "Maximum=%.4f\n",maxim );
printf( "Numbers within [lower,upper]: %d\n",count );
return 0;
}
```



Αποτελέσματα:

Give number 1: 23.22

Give number 2: -12

Give number 3: 0

Give number 4: 45.68

Give number 5: 5.2

Give number 6: -1.54

array[0]>0: 23.2200

array[3]>0: 45.6800

array[4]>0: 5.2000

Maximum=45.6800

Numbers within [lower,upper]: 3

4.

Το πρόγραμμα **example_strlen.c** επιτελεί τα ακόλουθα:

- (α) Δέχεται από το πληκτρολόγιο δύο αλφαριθμητικά (μέγιστου μήκους 12) και τα αποθηκεύει.
- (β) 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) Αντιγράφει σε νέο πίνακα χαρακτήρων κατάλληλου μήκους τους χαρακτήρες του πρώτου αλφαριθμητικού που βρίσκονται σε άρτια θέση και τυπώνει το νέο πίνακα χαρακτήρων στην οθόνη ως αλφαριθμητικό.

5.

Να γραφεί πρόγραμμα στη γλώσσα C, το οποίο θα λαμβάνει από το πληκτρολόγιο τις τιμές ενός πίνακα πραγματικών αριθμών `arr[14][14]`. Στη συνέχεια θα υπολογίζει:

- Σε κάθε γραμμή το στοιχείο με τη μέγιστη απόλυτη τιμή. Για κάθε γραμμή θα απεικονισθούν στην οθόνη η απόλυτη τιμή του μέγιστου στοιχείου και η στήλη στην οποία βρίσκεται.
- Το άθροισμα των στοιχείων της κύριας διαγωνίου (ίχνος του πίνακα).
- Τον πίνακα που θα προκύψει εάν αντιμετωπιστούν η δεύτερη με την τρίτη στήλη του `arr` και, στη συνέχεια, η πρώτη με την τρίτη γραμμή.

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

```
#include <math.h>
```

```
#define N 3 //αντί για 14 για λόγους απλότητας και δυνατότητας απεικόνισης
```

```
int main()
```

```
{
```

```
float arr[N][N],maxim,trace=0.0,temp;
```

```
int i,j,col;
```



```
for (i=0;i<N;i++)  
    for (j=0;j<N;j++) {  
        printf("\narr[%d][%d]: ",i+1,j+1);  
        scanf("%f",&arr[i][j]);  
    }  
for (i=0;i<N;i++) {  
    trace=trace+arr[i][i];  
    maxim=fabs(arr[i][0]);  
    col=1;  
    for (j=1;j<N;j++)  
        if (fabs(arr[i][j])>maxim) {  
            maxim=fabs(arr[i][j]);  
            col=j+1;  
        }  
    printf("\nLine %d: column %d, size=%f",i+1,col,maxim);  
}  
printf("\n\nTrace(arr)=%f\n",trace);  
for (j=0;j<N;j++) {  
    temp=arr[j][1];  
    arr[j][1]=arr[j][2];  
    arr[j][2]=temp;  
}
```

αντιμετάθεση της δεύτερης με
την τρίτη στήλης


```
for (j=0;j<N;j++) {  
    temp=arr[0][j];  
    arr[0][j]=arr[2][j];  
    arr[2][j]=temp;  
}  
return 0;  
}
```

**αντιμετάθεση της πρώτης με την
τρίτη γραμμή**

arr[1][1]: 23.1

arr[1][2]: -32

arr[1][3]: 5.7

arr[2][1]: 1.15

arr[2][2]: -0.02

arr[2][3]: 0.17

arr[3][1]: 3.41

arr[3][2]: 4.2

arr[3][3]: -1

Line 1: column 2, size=32.000000

Line 2: column 1, size=1.150000

Line 3: column 2, size=4.200000

Trace(arr)=22.080000

Actual array:

23.1000	-32.0000	5.7000
1.1500	-0.0200	0.1700
3.4100	4.2000	-1.0000

Final array:

3.4100	-1.0000	4.2000
1.1500	0.1700	-0.0200
23.1000	5.7000	-32.0000

6.

Να γραφεί πρόγραμμα στη γλώσσα C, το οποίο θα επιτελεί τα ακόλουθα:

- Θα δέχεται από το πληκτρολόγιο τα ονοματεπώνυμα δύο φοιτητών, έως 40 χαρακτήρες το καθένα και δοσμένα με κεφαλαία λατινικά γράμματα, και θα τα αποθηκεύει σε δισδιάστατο πίνακα χαρακτήρων `all[2][41]`.
- Θα διαχωρίζει τα μικρά ονόματα από τα επώνυμα και θα τα αποθηκεύει σε δύο ξεχωριστούς δισδιάστατους πίνακες χαρακτήρων, `nm[2][16]` για τα μικρά ονόματα και `sr[2][26]` για τα επώνυμα.
- Θα ταξινομεί αλφαβητικά τα επώνυμα και θα αναδιατάσσει τόσο τα μικρά ονόματα όσο και τα επώνυμα σε δύο νέους πίνακες `nm_new[2][16]` και `sr_new[2][26]`. Θα πρέπει να ληφθεί μέριμνα ώστε εάν το ένα επώνυμο είναι υποσύνολο του άλλου (π.χ. ΧΑΤΖΙΣΑΒΒΑΣ και ΧΑΤΖΙΣ) ταξινομείται ως πρώτο το συντομότερο εξ αυτών.
- Θα εμφανίζει τους πίνακες `nm`, `sr`, `nm_new`, `sr_new` στην οθόνη.

Δίνεται ότι:

- Τα ονοματεπώνυμα δόθηκαν σωστά, με κεφαλαία λατινικά γράμματα, και δεν απαιτείται έλεγχος γι' αυτό.
- Όταν ο χρήστης πληκτρολογεί ένα ονοματεπώνυμο διαχωρίζει το όνομα από το επώνυμο με απλό κενό.



- Κάθε μικρό όνομα και επώνυμο είναι απλό, δεν περιέχει διπλά ονόματα, τίτλους ευγενείας ή άλλου είδους προσφωνήσεις.

Επαλήθευση: Εάν ο χρήστης δώσει “JOAN JAMESON” και “ANDREW DOE”, οδηγούμαστε στους ακόλουθους πίνακες:

`all[0] = “JOAN JAMESON”`

`nm[0] = “JOAN”`

`sr[0] = “JAMESON”`

`nm_new[0] = “ANDREW”`

`sr_new[0] = “DOE”`

`all[1] = “ANDREW DOE”`

`nm[1] = “ANDREW”`

`sr[1] = “DOE”`

`nm_new[1] = “JOAN”`

`sr_new[1] = “JAMESON”`

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

```
#include <conio.h>
```

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

```
int main()
```

```
{
```

```
char all[2][41],nm[2][16],nm_new[2][16],sr[2][26],sr_new[2][26];
```

```
int i,j,mikos,count=0;
```

```
printf("Give first name: ");
```

```
gets(all[0]);
```

//gets για να αναγνωρισθούν τα κενά



```
printf("\nGive second name: ");
gets(all[1]);
for (i=0;i<2;i++) {
    j=0;
    while (all[i][j]!=' ') {
        nm[i][j]=all[i][j];
        j++;
    }
    nm[i][j]='\0';
    j=0;
    do {
        sr[i][j]=all[i][strlen(nm[i])+j]; //Ετσι δε λαμβάνεται υπόψη το ενδιάμεσο κενό
        j++;
    } while (all[i][j]!='\0');
    printf("\nnm[%d]=%s\tsr[%d]=%s",i,nm[i],i,sr[i]);
}
mikos=(strlen(sr[0])<strlen(sr[1]))?strlen(sr[0]):strlen(sr[1]);
i=0;
```



```
do
{
    if (sr[0][i]<sr[1][i]) count++;
    i++;
} while ((i<mikos) && (!count));
if ((count) || (strlen(sr[0])>strlen(sr[1])))
{
    strcpy(sr_new[0],sr[1]);
    strcpy(sr_new[1],sr[0]);
    strcpy(nm_new[0],nm[1]);
    strcpy(nm_new[1],nm[0]);
}
else for (i=0;i<2;i++) {
    strcpy(sr_new[i],sr[i]);
    strcpy(nm_new[i],nm[i]);
}
for (i=0;i<2;i++)
    printf("\nnm_new[%d]=%s\tsr_new[%d]=%s",i,nm_new[i],i,sr_new[i]);
return 0;
} // τέλος της main
```



```
Give first name: JOAN JAMESON
Give second name: ANDREW DOE

nm[0]=JOAN      sr[0]= JAMESON
nm[1]=ANDREW    sr[1]= DOE
nm_new[0]=ANDREW sr_new[0]= DOE
nm_new[1]=JOAN  sr_new[1]= JAMESON_
```

```
Give first name: JOHN XATZISAVVAS
Give second name: JOHN XATZIS

nm[0]=JOHN      sr[0]= XATZISAVVAS
nm[1]=JOHN      sr[1]= XATZIS
nm_new[0]=JOHN  sr_new[0]= XATZIS
nm_new[1]=JOHN  sr_new[1]= XATZISAVVAS_
```

```
Give first name: JOHN KARAGIANNIS
Give second name: JOHN KARABLASSOPOYLOS

nm[0]=JOHN      sr[0]= KARAGIANNIS
nm[1]=JOHN      sr[1]= KARABLASSOPOYLOS
nm_new[0]=JOHN  sr_new[0]= KARABLASSOPOYLOS
nm_new[1]=JOHN  sr_new[1]= KARAGIANNIS_
```