Κεφάλαιο 9.1-9.2

Αλφαριθμητικές Σειρές Χαρακτήρων (Strings)

(Διάλεξη 19)

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

1) Strings στη C

- Ένα string είναι μία ακολουθία αλφαριθμητικών **χαρακτήρων**, σημείων στίξης κτλ.
- Π.χ. “Hello” “How are you?” “121212” “*Apple#123*%”

Σημερινή Διάλεξη

1. Εισαγωγικές Έννοιες σε Strings(Αρχικοποίηση, Ανάγνωση & Εκτύπωση)
2. Πίνακες από Strings
3. Συναρτήσεις Βιβλιοθήκης <string.h>

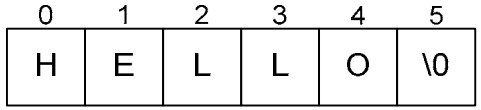
Το άλλο Μάθημα

Υλοποίηση Συναρτήσεων <string.h>

1) Strings στη C

- Έχουμε ήδη χρησιμοποιήσει σε διάφορες περιπτώσεις τα strings π.χ. printf(“Hello”);
- Μέχρι τώρα όμως, δεν αναφερόμασταν στα strings με την χρήση μεταβλητών.

- Στην C δεν υπάρχει ο τύπος String (όπως το float, int και char), αλλά αυτός υλοποιείτε ως **πίνακες χαρακτήρων**



- Εισάγουμε τώρα την έννοια των strings (αντί πιο πριν), διότι τώρα έχουμε κάποια εμπειρία με πίνακες.

1.1) Αρχικοποίηση string

Ορισμός String

Ένας πίνακας από χαρακτήρες που τελειώνει με τον χαρακτήρα **NULL '\0' ,**

π.χ. |Hello\0|, |Hi there\0|, |\0|

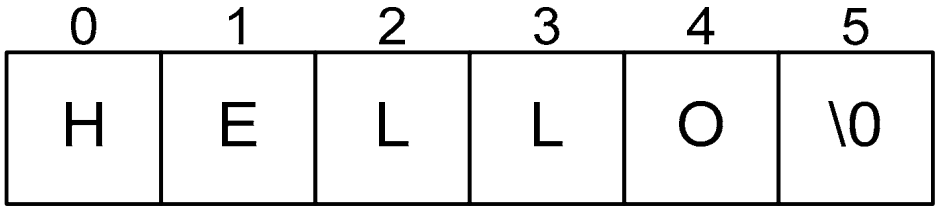
(Προσοχή το \0 δεν φαίνεται στην οθόνη)

Υπάρχουν διάφοροι τρόποι να ορίσουμε ένα String στην C, ανάλογα με το αν γνωρίζουμε το string προτού την μεταγλώττιση ή όχι.

Α) Αρχικοποίηση (γνωρίζοντας εκ'των πρότερων το String)

char msg[ ]="Hello";

Δημιουργεί αυτόματα το:



1.1) Αρχικοποίηση string

Συνίσταται αυτός ο ορισμός
char msg[ ]="Hello"

αλλά υπάρχουν και άλλοι τρόποι…

Σωστό
```
char msg[6];
msg[0] = 'H';
msg[1] = 'e';
msg[2] = 'l';
msg[3] = 'l';
msg[4] = 'o';
msg[5] = '\0';
```

Σωστό
char msg[ ]={'H','e','l','l','o','\0'};

Σωστό
char msg[6]={'H','e','l','l','o','\0'};

Σωστό
char msg[40]="Hello"

Λάθος
char msg[ ]={'H','e','l','l','o'};

Πίνακας χαρακτήρων που δεν τελειώνει σε \0. Επομένως δεν είναι String
Αν κάποιος εκτελέσει printf("%s", msg); Τότε στην οθόνη θα δείξει |Hello???|

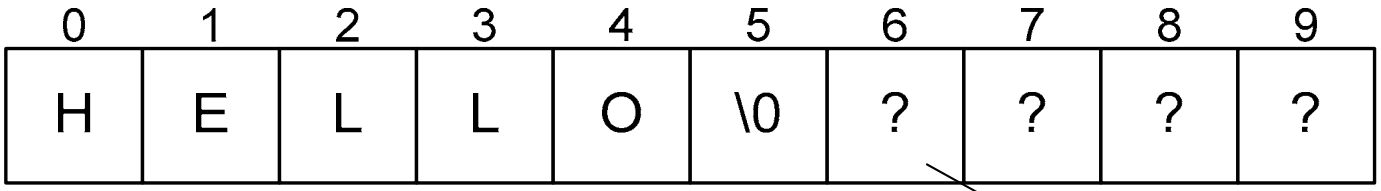
Λάθος → **Δεν δεσμεύουμε αρκετό χώρο**
char msg[5]={'H','e','l','l','o','\0'};

1.1) Αρχικοποίηση string

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char msg[10]=“Hello”;

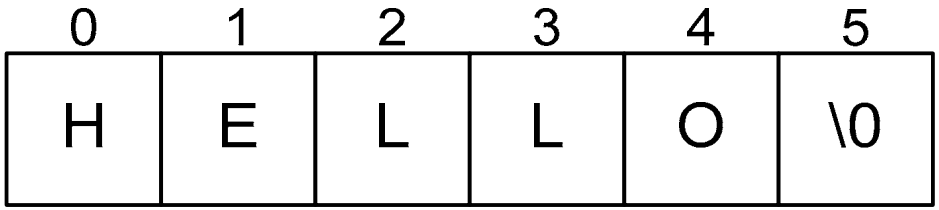
SIZE=10



Το ‘?’ είναι απροσδιόριστος χαρακτήρας

char msg[ ]=“Hello”;

SIZE=6



1.1) Η διάφορα char και string

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char c = 'H';

Ο χαρακτήρας Η βρίσκεται κάπου στην μνήμη

?	?	?	**H**	?	?	?	?	?	?

char c[ ]="H";

0	1	2	3	4	5
		H	\0		

Το String Η βρίσκεται κάπου στην μνήμη

char c="H"; **ΛΑΘΟΣ στην μεταγλώττιση**

1.1) Αρχικοποίηση string

Ερώτηση

- Τι γίνεται αν δεν ξέρουμε το String εκ' των πρότερων?
- Πόσο χώρο πρέπει να δεσμεύσουμε?

Απάντηση

- Αρκετό για να χωρέσει διάφορα δεδομένα εισόδου που πρόκειται να εισάγουμε.

  π.χ. αν πρόκειται να αποθηκεύσουμε κάποιο **όνομα** τότε 20 χαρακτήρες φαίνεται να αρκούν.

- Στην πραγματικότητα χρησιμοποιούμε **δυναμική δέσμευση μνήμης**, η οποία μας επιτρέπει να δεσμεύσουμε τον απαιτούμενο χώρο κατά την διάρκεια εκτέλεσης του προγράμματος (αλλά δεν θα μιλήσουμε για αυτό το θέμα σε αυτό μάθημα).

Προτού δούμε ένα παράδειγμα, ας δούμε πως διαβάζουμε strings από τον χρήστη και πως τα εκτυπώνουμε….

1.2) Ανάγνωση/Εκτύπωση String

scanf (“%s”, name)

ΠΡΟΣΟΧΗ: Δεν χρησιμοποιείτε το &, γιατί το name είναι πίνακας.

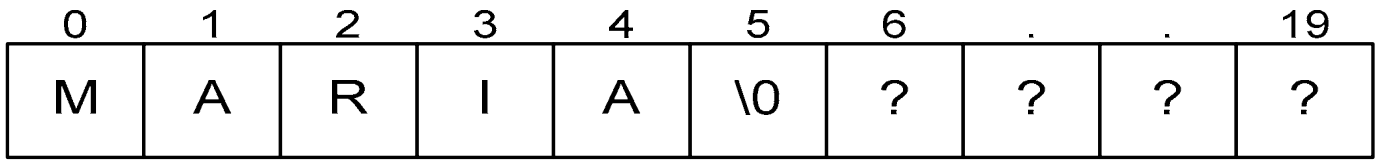
Θυμηθείτε ότι σε άλλους τύπους δεδομένων χρησιμοποιείτε π.χ. scanf(“%d”, &a);

printf (“%s”,name)

1.2) Ανάγνωση/Εκτύπωση string

```
#include <stdio.h>
int main()
{
   char name[20];
   scanf("%s", name);
   printf("%s", name);
   return 0;
}
```

Το printf γνωρίζει ότι η εκτύπωση πρέπει να γίνει μέχρι το \0

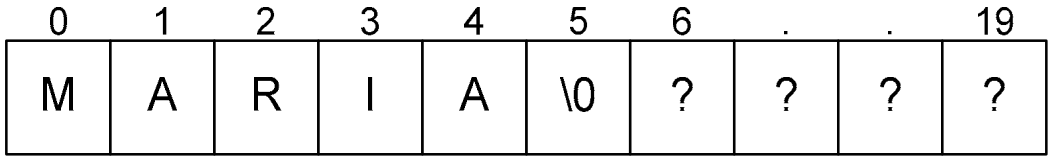


1.2) Ανάγνωση/Εκτύπωση string

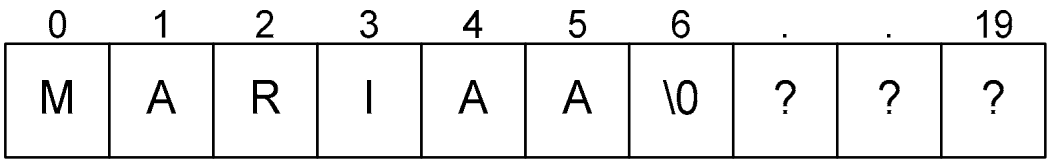
Πρόγραμμα που προσθέτει ένα «A» στην θέση 5 και τερματίζει το string

```
#include <stdio.h>
int main()
{
    char name[20];
    scanf("%s", name);
    name[5]='A';
    name[6]='\0';
    printf("%s", name);
    return 0;
}
```

Πρίν



Μετά



1.2) Ανάγνωση/Εκτύπωση string

Μπορούμε βέβαια να διαβάσουμε πολλαπλά πεδία ταυτόχρονα

```
int id;
char tname[20];
scanf("%s %d",tname,&id);
printf("You entered : %s and %d",tname, id);
```

- Αν δώσει κανείς σαν τιμή εισόδου

  MARIA 5

  Τότε θα εκτυπωθεί «You entered : MARIA and 5»

Κοινό Λάθος (Ανάθεση σε String)

Υποθέστε ότι έχουμε την πιο κάτω δήλωση
char name [50];

Δεν επιτρέπετε να αναθέσουμε ένα string από ένα άλλο (όπως στους υπόλοιπους τύπους)
Π.χ.

- name=“hydrogen” **ΛΑΘΟΣ (compile error)**
- name={‘h’,’y’,’d’,’r’,’o’,’\0’} **ΛΑΘΟΣ (compile error)**
- **name[0]=“h”; name[1]=“y”; …. ΟΡΘΟ**

Όμως, υπάρχει ευκολότερος τρόπος τον οποίο θα δούμε σε λίγο (με χρήση strcpy)

2) Πίνακες από Strings

- Είπαμε ότι το String είναι ένας μονοδιάστατος πίνακας από χαρακτήρες που τελειώνει σε \0.
- **Μπορούμε να έχουμε πίνακες από Strings?**

  ΝΑΙ π.χ. Λίστα με ονόματα, ημέρες, κτλ
- **Παραδείγματα**
  - char names[NUM_STUDENTS][NAME_LEN];
  - char weekdays[7][10]={“Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, “Sunday”};

	0	1	2	3	4	5	6			
0	M	o	n	d	a	y	\0	?	?	?
1	T	u	e	s	d	a	y	\0	?	?
6	S	u	n	d	a	y	\0	?	?	?

3) Συχνές λειτουργίες με string

Υπάρχουν κάποιες λειτουργίες που είναι πολύ συχνές πάνω σε Strings.

Παραδείγματα

- **Compare:** Σύγκριση δυο strings
- **Copy:** Αντιγραφή από ένα string σε άλλο
  - ολόκληρο ή μέρος του.
- **Concat:** συνέδεσε δυο strings μαζί
- **Substring**: πόσες γραμμές περιέχουν το abc

Η Βιβλιοθήκη <String.h> περιέχει συναρτήσεις για όλα τα πιο πάνω.

Για εξάσκηση θα υλοποιήσουμε κάποιες από τις συναρτήσεις μόνοι μας

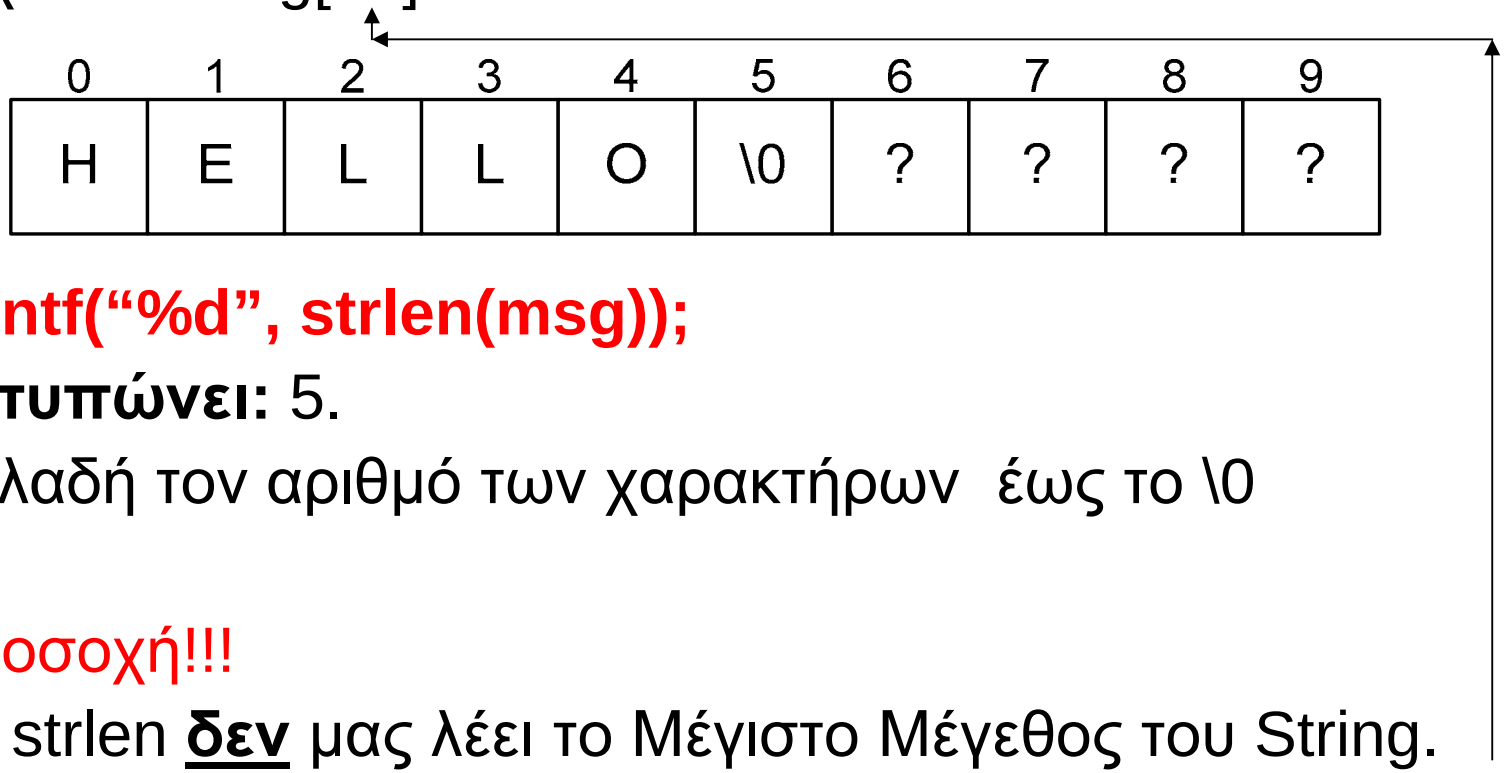
3) Η Βιβλιοθήκη string.h

- Το αρχείο επικεφαλίδα (header file), string.h παρέχει συναρτήσεις για χειρισμό strings
- Περιέχει Διάφορες Συναρτήσεις, πχ.
  - **strlen(s),** υπολογίζει το μέγεθος του string
  - **strcpy(s1,s2),** αντιγράφει το s2 στο s1
  - **strcat(s1,s2),** προσθέτει το s2 στο s1.
  - **strcmp(s1,s2),** συγκρίνει το s1 με s2 και επιστρέφει θετική τιμή εάν s1 μεγαλύτερο (αλφαβητικά) από το s2, μηδέν αν είναι ίσα, και αρνητική τιμή εάν s1 μικρότερο από s2.

    (Η σύγκριση γίνεται βάση του πίνακα ASCII)

3) Η συνάρτηση strlen()

- Η συνάρτηση strlen μετρά το μέγεθος ενός String.
  π.χ. char msg[10]=“HELLO”



printf(“%d”, strlen(msg));

Εκτυπώνει: 5.

Δηλαδή τον αριθμό των χαρακτήρων έως το \0

Προσοχή!!!

Το strlen **δεν** μας λέει το Μέγιστο Μέγεθος του String.

Αυτό είναι 10 χαρακτήρες και το ξέρουμε πριν το compile

3) Η συνάρτηση strlen()

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

int main()
{
   char x[10] = "123456" ;
   printf("%d", strlen(x));
}
```

Τυπώνει 6

3) Η συνάρτηση strlen()

- Ξέρουμε ότι η strlen είναι έτοιμη συνάρτηση.
- Για εξάσκηση, θα την υλοποιήσουμε μόνοι μας

ΑΣΚΗΣΗ

- Γράψετε την συνάρτηση
  int mystrlen(char c[])

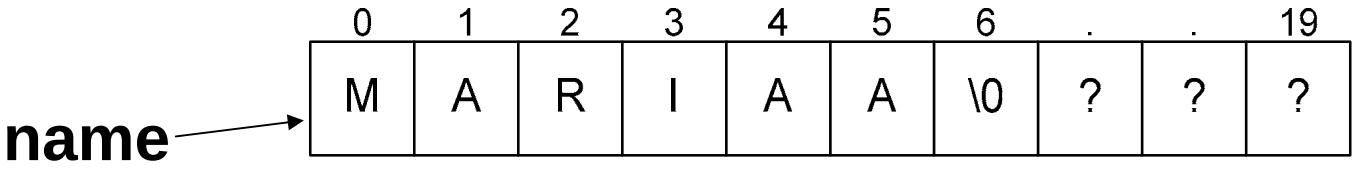
  που μέτρα το μέγεθος ενός string. Η συνάρτηση επιστρέφει το μήκος του string

- **Τι θα επιστρέψει?**
  - mystrlen(“abc”) **=> 3**
  - char x[10] = “123456” ; mystrlen(x); **=> 6**
  - mystrlen(“abc abc”) **=> 7**

3) Η συνάρτηση strlen()

Αλγόριθμος

- Διάβασε το string από την αρχή μέχρι να βρεις το \0.
- Σε κάθε βήμα αύξησε κάποιο μετρητή κατά 1.



Ερώτηση

- Γιατί δεν μπορούμε να πάμε κατευθείαν στο τέλος του String (για να βρούμε κατευθείαν το μέγεθος)?

3) Η συνάρτηση strlen()

```
#include <stdio.h>

int mystrlen (char s[])
{
   int i=0;
   while (s[i] != '\0')
         i++;
    return i;
}
```

```
int main()
{
      char x[10] = "123456" ;
      printf("%d", mystrlen(x));
}
```