Διάλεξη 2: Επανάληψη Προγραμματισμού – Συμβολοσειρές (strings)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Εισαγωγικές Έννοιες σε Strings
- Πίνακες από Strings
- Συναρτήσεις Βιβλιοθήκης <string.h>
- Υλοποίηση Συναρτήσεων Βιβλιοθήκης

Διδάσκων: Παναγιώτης Ανδρέου

Συμβολοσειρές (Strings)

Μία συμβολοσειρά (string) είναι μία ακολουθία αλφαριθμητικών **χαρακτήρων**, σημείων στίξης, π.χ. “Hello” “121212” “*Apple#123*%”

- 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 υλοποιείται ως πίνακες χαρακτήρων
- Επαναλαμβάνουμε τα strings γιατί θα μας δώσει την δυνατότητα να δουλέψουμε με την απλούστερη δομή δεδομένων: τον πίνακα, αλλά και για λόγους επανάληψης

Ορισμός String: Ένας πίνακας από χαρακτήρες που τελειώνει με τον χαρακτήρα NULL ‘\0’ , π.χ. |Hello\0|

(Προσοχή το \0 δεν φαίνεται στην οθόνη)

Αρχικοποίηση string

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

A. <u>Αρχικοποίηση (γνωρίζοντας εκ'των πρότερων το String)</u>

`char msg[]="Hello";` Δημιουργεί το:

H	E	L	L	O	\0

Άλλοι τρόποι:

Σωστό
```
char msg[6];
msg[0]='H';
msg[1]='e';
…
msg[5] = '\0';
```

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

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

Σωστό (αλλά σπάταλο!)
```
char msg[40]="Hello";
```

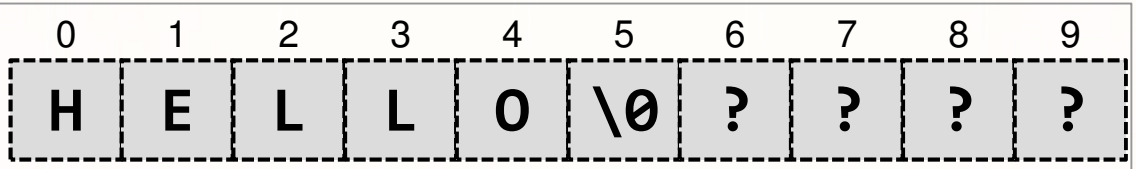
!!! Λάθος !!!
```
char msg[ ]
={'H','e','l','l','o'};
```
Πίνακας χαρακτήρων που δεν τελειώνει σε \0. Επομένως δεν είναι String. Αν κάποιος εκτελέσει printf("%s", msg); Τότε στην οθόνη θα δείξει |Hello???| π.χ, msg=Hello╠╠╠╠╠╠╠╠╠C╠╠╠

!!! Λάθος !!!
```
char msg[5]=
{'H','e','l','l','o','\0'};
```
Δεν δεσμεύουμε αρκετό χώρο και δημιουργείται buffer overflow

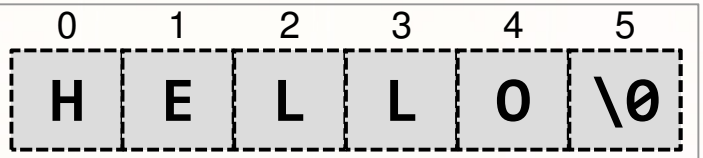
Αρχικοποίηση string

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

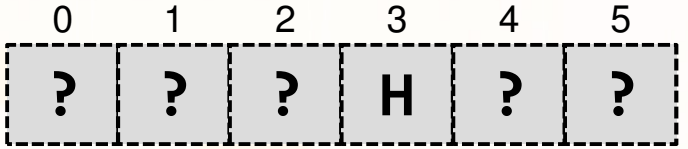
- **char msg[10]="Hello";**
  - Size (sizeof(msg))= 10
  - String Size (strlen(msg)) = 5



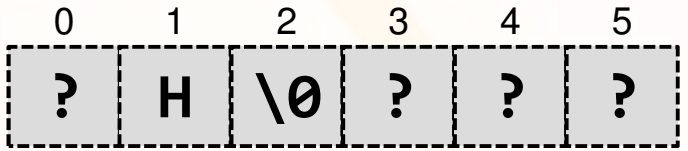
- **char msg[ ]="Hello";**
  - Size = 6
  - String Size = 5



- **char c = 'H';**
  - Ο **χαρακτήρας 'H'** βρίσκεται κάπου στη μνήμη



- **char c[ ]="H";**
  - Το **string "H"** βρίσκεται κάπου στη μνήμη



- **char c="H";**

!!! Λάθος στην μεταγλώττιση!!!

Αρχικοποίηση string

Β. <u>Αρχικοποίηση (ΜΗ γνωρίζοντας εκ των προτέρων το String)</u>

- **<u>Ερώτηση:</u>** Πόσο χώρο πρέπει να δεσμεύσουμε;
- **<u>Απάντηση:</u>** Αρκετό για να χωρέσει διάφορα δεδομένα εισόδου που πρόκειται να εισάγουμε.

  π.χ., αν πρόκειται να αποθηκεύσουμε κάποιο όνομα τότε συνήθως 20-30 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 (θα μιλήσουμε για αυτό το θέμα στην συνέχεια του μαθήματος)!

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

Ανάγνωση/Εκτύπωση string

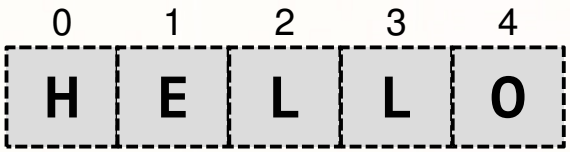
- Η μέθοδος **`printf`** τυπώνει στην πρότυπη έξοδο (συνήθως η οθόνη)
- Παραδείγματα:
  - `int x=5; printf(“%d”, x); //Διαβάζω ένα ακέραιο αριθμό`
  - `char name[20]=“Hello”; printf(“%s”, name);`
- Η μέθοδος **`scanf`** διαβάζει από την πρότυπη είσοδο (συνήθως το πληκτρολόγιο) διάφορους τύπου δεδομένων (π.χ., ακέραιους, συμβολοσειρές, κ.ο.κ.
- Παραδείγματα:
  - `int x; scanf(“%d”, &x); //Διαβάζω ένα ακέραιο αριθμό`
  - `char name[20]; scanf(“%s”, &name); // Διαβάζω μία συμβολοσειρά.` Το **&** δεν χρειάζεται γιατί η μεταβλητή name είναι πίνακας.

Ανάγνωση/Εκτύπωση string

- Ερώτηση 1: Τι θα γίνει σε αυτή την περίπτωση;
  - Ο χρήστης δίνει σαν είσοδο “Hello”
  - **!!! Λάθος στην εκτέλεση. Δεν υπάρχει χώρος για να αποθηκευτεί το ‘\0’ !!!**
  - Αποφυγή λάθους χρησιμοποιώντας τον διαμορφωτή `scanf("%4s", name);`

Ερώτηση 1

```
char name[5];
scanf("%s", name);
```



- Ερώτηση 2: Τι θα γίνει σε αυτή την περίπτωση;
  - Ο χρήστης δίνει σαν είσοδο “`This is a test`”
  - **!!! Λογικό Λάθος. Το <space> δηλώνει ότι το string έχει τελειώσει. Θα τυπωθεί μόνο η λέξη “This” !!!**

Ερώτηση 2

```
char name[20];
scanf("%s", name);
printf("%s\n", name);
```

- Ερώτηση 3: Τι θα γίνει με το υπόλοιπο string;
  - **Το “is a test” παραμένει στη μνήμη της εισόδου!**
  - Αποφυγή λάθους χρησιμοποιώντας την μέθοδο “fflush(stdin);” μετά από κάθε διάβασμα.
  - Για εισαγωγή συμβολοσειράς με κενά χρησιμοποιείται η **`fgets(name, sizeof(name), stdin);`**

Ερώτηση 3

```
char name[20];
scanf("%s", name);
printf("%s\n", name);
scanf("%s", name);
printf("%s\n", name);
```

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	7	8	9
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	?	?	?

Συχνές Λειτουργίες με Strings

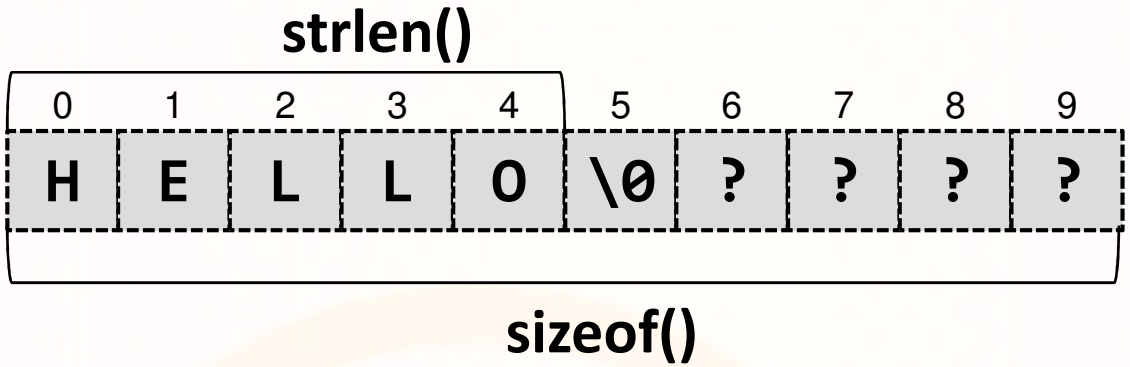
- Το αρχείο επικεφαλίδα (header file), <string.h> παρέχει συναρτήσεις για χειρισμό strings
- Υπάρχουν κάποιες συναρτήσεις που είναι πολύ συχνές με Strings:
  - Μήκος (strlen): Επιστρέφει το μήκος ενός string
  - Αντιγραφή (strcpy): Αντιγράφει ένα string σε κάποιο άλλο
  - Συνένωση (strcat): Τοποθετεί ένα string συμβολοσειράς στο τέλος κάποιου άλλου
  - Σύγκριση (strcmp): συγκρίνει δύο strings
- Υπάρχουν επίσης κάποιες συναρτήσεις που δεν είναι πολύ συχνές
  - Αναζήτηση (strstr): Επιστρέφει την αφετηρία ενός string που βρίσκεται εντός ενός άλλου
  - Αναζήτηση (strchr): Επιστρέφει την αφετηρία ενός char που βρίσκεται εντός ενός string
  - Κατακερματισμός (strtok): Επιστρέφει ένα δείκτη στο τελευταίο τμήμα (token) που έχει βρεθεί σε ένα string βάση κάποιων οριοθετών (delimiters)
- Για εξάσκηση, θα υλοποιήσουμε μόνοι μας κάποιες συναρτήσεις

Η συνάρτηση <strlen>

- Επιστρέφει το μήκος ενός string
- Υπογραφή: size_t strlen(const char * str)

  Δεν επιτρέπεται η αλλαγή του str

  Παράμετρος εξόδου: size_t = unsigned long int
- Παράδειγμα: char msg[10]="Hello";



- **ΠΡΟΣΟΧΗ**: Η συνάρτηση strlen δεν μας λέει το μέγιστο μέγεθος του string. Αυτό είναι 10 χαρακτήρες και το ξέρουμε πριν το compile ή με printf("%d", **sizeof(**msg**)**);
- Ερώτηση 1: Τι θα γίνει σε αυτή την περίπτωση;

```
printf("%s\n", strlen(""));
```

Άσκηση: Υλοποίηση συνάρτησης <strlen>

- Υπογραφή: size_t **mystrlen**(const char * str)
- Αλγόριθμος:
  - Αρχικοποίησε κάποιο μετρητή <c> σε 0.
  - Διάβασε ένα-ένα χαρακτήρα από το string <str> μέχρι να βρεις τον ειδικό χαρακτήρα “\0”.
    - Σε κάθε βήμα αύξησε το μετρητή <c> κατά 1.
  - Επέστρεψε το <c> σαν αποτέλεσμα

```
int mystrlen(
 const char * str){
   int c=0;
   while(str[c]!='\0'){
       c++;
   }
   return c;
}
```

```
int mystrlen(
 const char * str){
   int c=0;
   while(str[c++]!='\0');
   // OR while(str[c++]);

   return --c;
}
```

Η συνάρτηση <strcpy>

- Αντιγράφει ένα string σε κάποιο άλλο
- Υπογραφή: char * strcpy (char * destination, const char * source);

  Παράμετρος εξόδου: char * = destination
- Παράδειγμα:

```
char s1[]="Hello";
char s2[7];
strcpy(s2,s1);
printf("%s\n", s2);
```

ΠΡΙΝ

	0	1	2	3	4	5	6
s1	H	E	L	L	O	\0	?
s2	?	?	?	?	?	?	?

ΜΕΤΑ

	0	1	2	3	4	5	6
s2	H	E	L	L	O	\0	?

Άσκηση: Υλοποίηση συνάρτησης <strcpy>

- Υπογραφή: void **mystrcpy** (char* dest, const char* source);
- Αλγόριθμος:
  - Αρχικοποίησε κάποιο μετρητή <i> σε 0.
  - Διάβαζε τους χαρακτήρες και κάθε φορά αντέγραψε τον χαρακτήρα που βρίσκεται στη θέση [i] από το string <source> στη θέση [i] στο string <dest> μέχρι να βρεις τον ειδικό χαρακτήρα “\0”.
    - Σε κάθε βήμα το αύξησε το μετρητή <i> κατά 1.

```
void mystrcpy(
 char * dest,
 const char * source ){
   int i=0;
   while(dest[i]!='\0'){
       dest[i]=source[i];
       i++;
   }
}
```

```
void mystrcpy(
 char * dest,
 const char * source ){
   int i=0;
   while(dest[i]=source[i++]);
}
```

Η συνάρτηση <strcat>

- Αντιγράφει ένα string σε κάποιο άλλο
- Υπογραφή: char * strcat (char * destination, const char * source);

  Παράμετρος εξόδου: char * = destination
- Παράδειγμα:

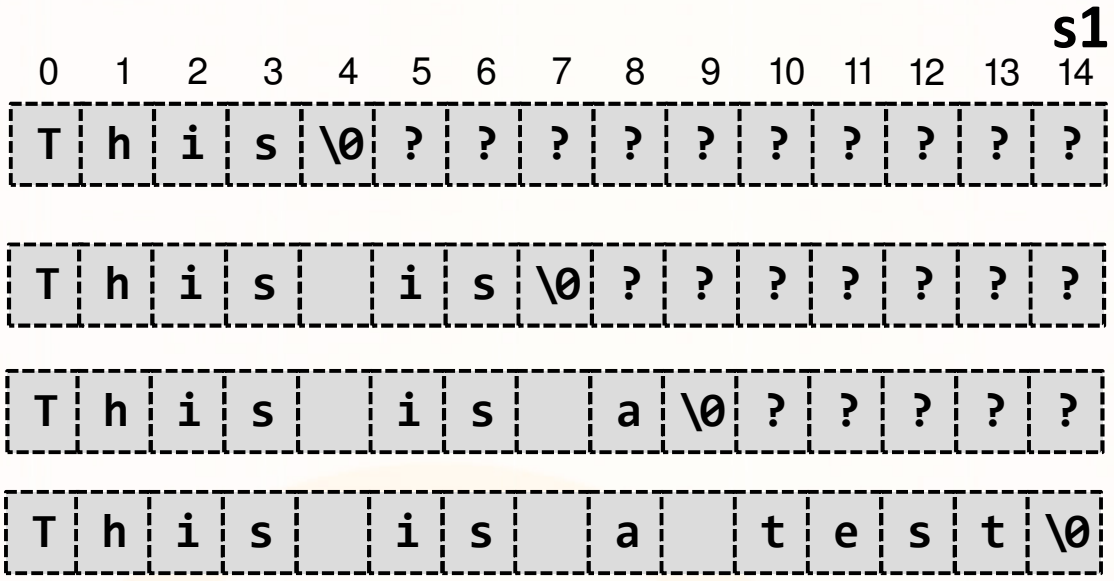
```
char s1[15]="This";

strcat(s1, " is");

strcat(s1, " a");

strcat(s1, " test");

printf("%s\n", s1);
```



Άσκηση: Υλοποίηση συνάρτησης <strcat>

- Υπογραφή: void **mystrcat** (char* dest, const char* source);
- Αλγόριθμος:
  - Αρχικοποίησε κάποιο μετρητή <i> σε 0.
  - Βρες τη θέση <k> όπου dest[k]='\0'
  - Αντέγραψε κάθε χαρακτήρα που βρίσκεται στη θέση [i] από το string <source> στη θέση [i+k] στο string <dest> μέχρι να βρεις τον ειδικό χαρακτήρα '\0'.
    - Σε κάθε βήμα αύξησε το μετρητή <i> κατά 1.

```
void mystrcat(
 char * dest,
 const char * source ){
   int i=0, k=0;
   while(dest[k]!='\0'){k++;}
   while(dest[i]!='\0'){
        dest[i+k]=source[i];
        i++;   }
}
```

```
void mystrcat(
 char * dest,
 const char * source ){
   int i=0, k=0;
   while(dest[k++])
   k--;
   while((dest[i+k]=source[i++]);
}
```

Η συνάρτηση <strcmp>

- Συγκρίνει ένα string με κάποιο άλλο
- Υπογραφή: int * strcmp (const char * str1, const char * str2);

  Παράμετρος εξόδου: int
  - **0**: τα strings str1 και str2 είναι ίδια
  - **>0**: Ο πρώτος χαρακτήρας που δεν είναι ίδιος στο string str1 είναι μεγαλύτερος από το χαρακτήρα με ίδια θέση στο str2
  - **<0**: αντίστροφα με το πιο πάνω

Η σύγκριση γίνεται βάση του πίνακα ASCII

- Παραδείγματα:

```
strcmp("Apple", "Android");      → 1
strcmp("Android", "Apple");      → -1
strcmp("Android", "Android");    → 0
strcmp("Android", "android");    → -1
```