Επανάληψη
Αλφαριθμητικές Σειρές Χαρακτήρων (Strings)

(Διάλεξη 2)

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: Χρήση της πιο απλής δομής δεδομένων: πίνακας

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

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 υλοποιείται ως **πίνακες χαρακτήρων**

0	1	2	3	4	5
H	E	L	L	O	\0

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

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|

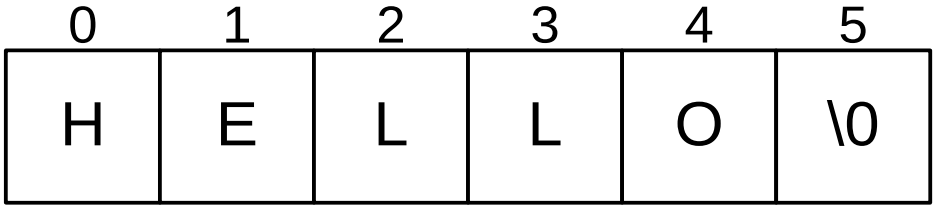
(Προσοχή το \0 δεν φαίνεται στην οθόνη)

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 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???|

Δεν δεσμεύουμε αρκετό χώρο και δημιουργείται buffer overflow

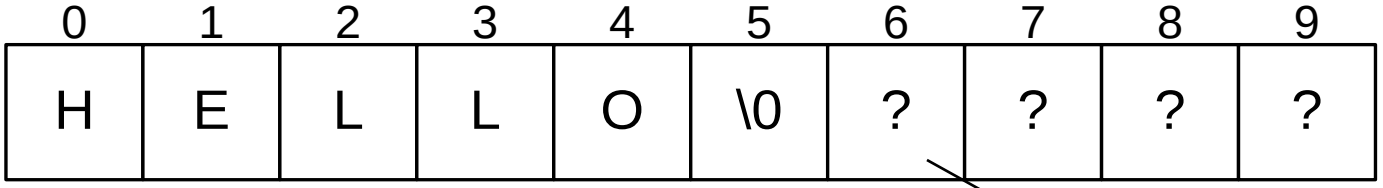
Λάθος
char msg[5]={'H','e','l','l','o','\0'};

1.1) Αρχικοποίηση string

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char msg[10]="Hello";

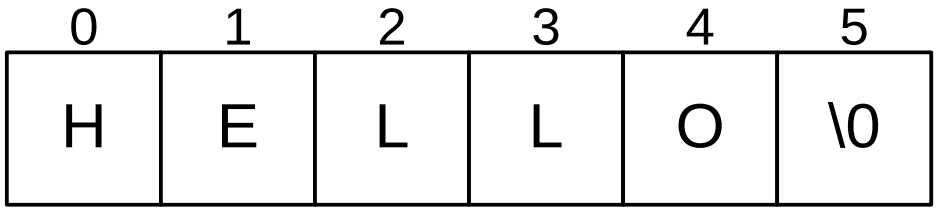
SIZE=10



Το '?' είναι απροσδιόριστος χαρακτήρας

char msg[ ]="Hello";

SIZE=6

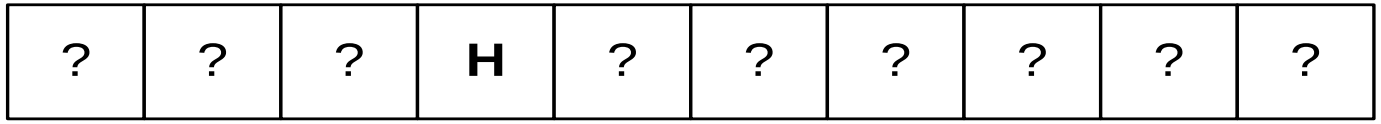


1.1) Η διάφορα char και string

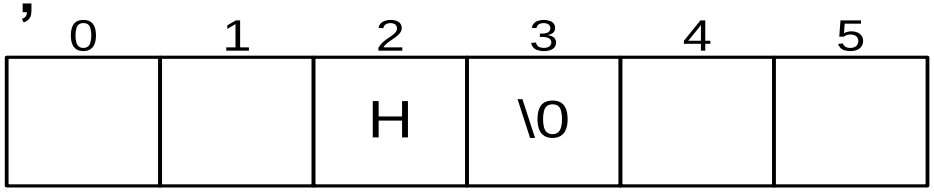
Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char c = 'H';

Ο χαρακτήρας Η βρίσκεται κάπου στην μνήμη



char c[ ]="H";



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 (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….

1.2) Ανάγνωση/Εκτύπωση String

scanf (“%s”, name)

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

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

- Για εισαγωγή συμβολοσειράς με κενά χρησιμοποιείται η
  - fgets(name, sizeof(name), stdin); ή
  - #define MAX "50“

    scanf("%" MAX "[^\n]", name);

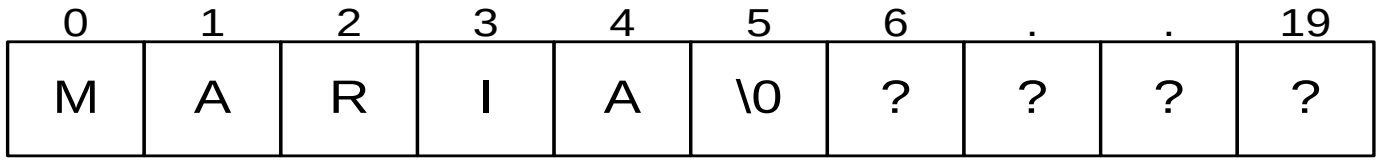
printf (“%s”,name)

- Για εκτύπωση συμβολοσειρών

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

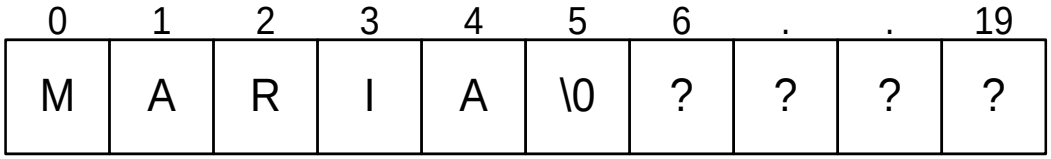


1.2) Ανάγνωση/Εκτύπωση string

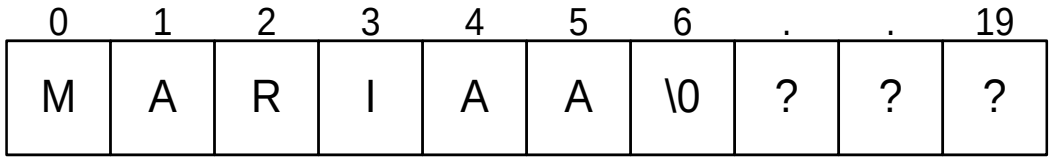
Πρόγραμμα που προσθέτει ένα «A» στην θέση 5 και τερματίζει το string

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

Πρίν



Μετά



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)**
  - **error: incompatible types in assignment**
- 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:** σύνδεση δυο strings
- **Substring**: εύρεση συμβολοσειράς σε μια μεγαλύτερη ακολουθία

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

Για εξάσκηση θα υλοποιήσουμε κάποιες από τις συναρτήσεις μόνοι μας

3) Η συνάρτηση strcmp()

Οι συγκρίσεις γίνονται βάση του πίνακα ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Spa	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DE

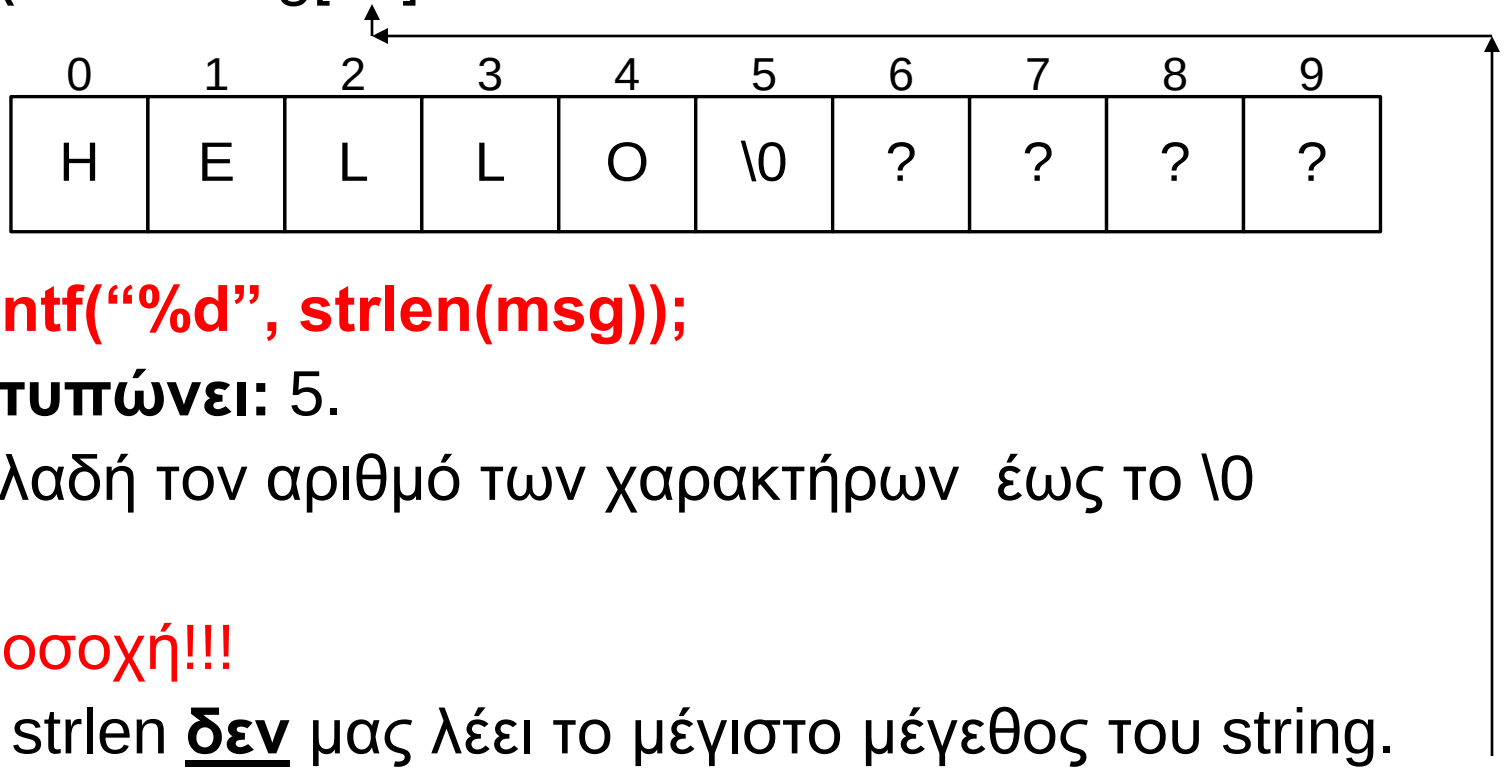
3) Η Βιβλιοθήκη string.h

- Το αρχείο επικεφαλίδα (header file), string.h παρέχει συναρτήσεις για χειρισμό strings
- Περιέχει Διάφορες Συναρτήσεις, π.χ.,
  - **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 ή με «printf("%ld", **sizeof(msg)**);»

3) Η συνάρτηση strlen()

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

int main()
{
   char x[10] = "123456" ;
   printf("%d", strlen(x));
   return 0;
}
```

Τυπώνει 6

3) Η συνάρτηση strlen()

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

ΑΣΚΗΣΗ

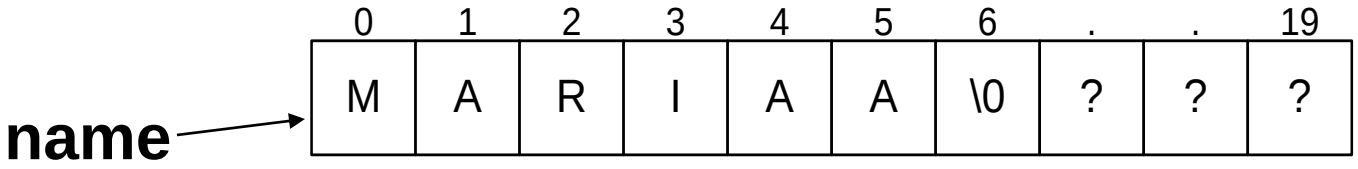
- Γράψετε την συνάρτηση
  int mystrlen(char c[])
  που μέτρα το μέγεθος ενός string. Η συνάρτηση επιστρέφει το μήκος του string

- **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));
}
```

1) Η συνάρτηση strcpy()

- Η συνάρτηση strcpy(ma, mb) αντιγράφει το mb στο ma

  π.χ.

Πρίν

	0	1	2	3	4	5	6	7	8	9
ma	?	?	?	?	?	?	?	?	?	?
	0	1	2	3	4	5	6	7	8	9
mb	H	E	L	L	O	\0	?	?	?	?

Μετά

	0	1	2	3	4	5	6	7	8	9
ma	H	E	L	L	O	\0	?	?	?	?
	0	1	2	3	4	5	6	7	8	9
mb	H	E	L	L	O	\0	?	?	?	?

```
int main()
{
    char ma[10];
    char mb[10]="HELLO";
    strcpy(ma,mb);
    printf("ma=%s and mb=%s", ma, mb);
}
```

1) Υλοποίηση της strcpy()

- Υλοποιήστε την συνάρτηση

  mystrcpy(char to[], char from[])

  η οποία αντιγράφει το String b στο String a

Αλγόριθμος

Για κάθε στοιχείο from[i] αντίγραψε το from[i] στην θέση to[i], μέχρι να φτάσεις στο \0.

1) Υλοποίηση της strcpy()

```
#include <stdio.h>
/* Το string 'from' αντιγράφεται στο 'to' */
void mystrcpy(char to[ ], char from[ ]) {
   int i=0;
   while (from[i] != '\0') {
            to[i] = from[i];
            i++;
    }
    to[i]='\0';
}

int main() {
            char ma[10];  char mb[10]="HELLO";
            mystrcpy(ma,mb);
            printf("ma=%s and mb=%s", ma, mb);
}
```

Πρίν

	0	1	2	3	4	5	6	7	8	9
to	?	?	?	?	?	?	?	?	?	?
	0	1	2	3	4	5	6	7	8	9
from	H	E	L	L	O	\0	?	?	?	?

Μετά

	0	1	2	3	4	5	6	7	8	9
to	H	E	L	L	O	\0	?	?	?	?
	0	1	2	3	4	5	6	7	8	9
from	H	E	L	L	O	\0	?	?	?	?

2) Η συνάρτηση strcat()

- Η συνάρτηση strcat(s1, s2) αντιγράφει το s2 στο τέλος του s1

  π.χ.

Πρίν

	0	1	2	3	4	5	6	7	8	9
s1	H	E	L	L	O	\0	?	?	?	?
s2	C	A	T	\0	?	?	?	?	?	?

Μετά

	0	1	2	3	4	5	6	7	8	9
s1	H	E	L	L	O	C	A	T	\0	?
s2	C	A	T	\0	?	?	?	?	?	?

```
int main()
{
    char ma[10]="HELLO";
    char mb[10]="CAT";
    strcat(ma,mb);
    printf("ma=%s and mb=%s", ma, mb);
}
```

2) Υλοποίηση strcat()

- Υλοποιήστε την συνάρτηση

  void mystrcat(char s1[], char s2[])

  η οποία προσθέτει το string s2 στο τέλος του s1

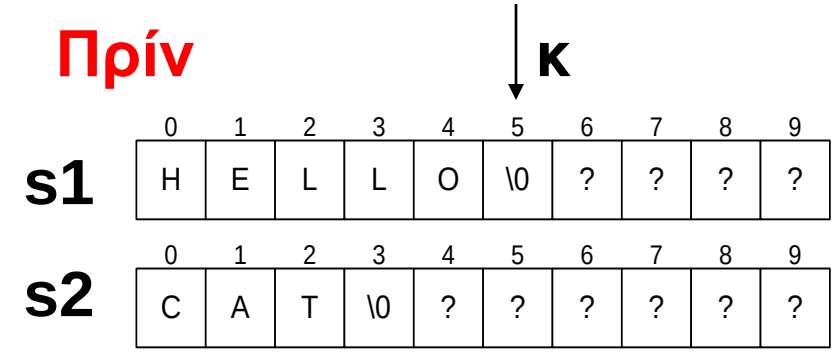
Αλγόριθμος

- Βρες το \0 στο s1 στην θέση K.
- Αντίγραψε κάθε s2[i], στην αντίστοιχη θέση s1[i+K].
- Πρόσθεσε \0 στο τέλος του s1.

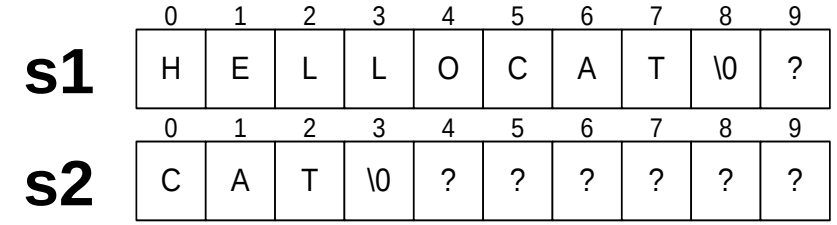
2) Υλοποίηση strcat() – έκδοση 1

```
void mystrcat(char s1[], char s2[])
{
    int i=0, k=0;
    // Εύρεση \0 στο S1
    while (s1[k] != '\0') {
          k++;
    }
    // Αντιγραφή Στοιχείων
     while (s2[i] != '\0') {
        s1[k+i]=s2[i];
        i++;
     }
    s1[k+i]='\0';          // Προσθήκη NULL στο τέλος του s1
}
```

Πρίν



Μετά

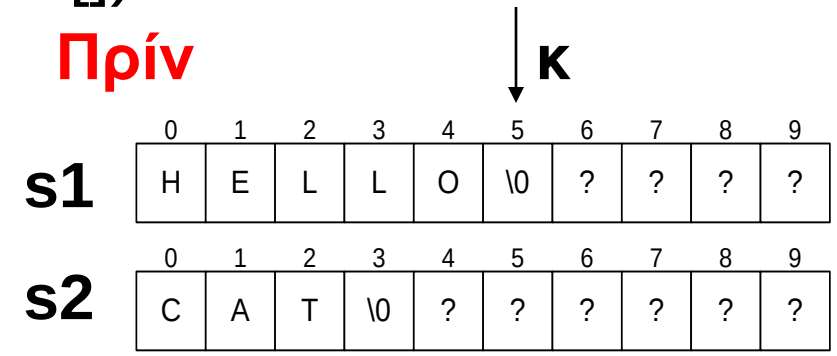


2) Υλοποίηση strcat() - έκδοση 2

```
void mystrcat(char s1[], char s2[])
{
   int i=0, k=0;
   // Εύρεση \0 στο S1
   k = strlen(s1);  // χρήση strlen

   // Αντιγραφή Στοιχείων
    while (s2[i] != '\0') {
       s1[k+i]=s2[i];
       i++;
    }
   s1[k+i]='\0';// προσθήκη NULL στο τέλος του s1
}
```

Πρίν



Μετά