ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Συμβολοσειρές, Δείκτες και Παραδείγματα (Κεφάλαιο 13, ΚΝΚ-2ED)

Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

http://www.cs.ucy.ac.cy/courses/EPL232

Dr. Andreas Aristidou a.aristidou@ieee.org

Περιεχόμενο Διάλεξης 7

- **Συμβολοσειρές (Strings)**
  - Σταθερά (literal), Μεταβλητή (variable), Αρχικοποίηση, Ανάγνωση / Εκτύπωση
  - Παραδείγματα: Σάρωση, Μέτρηση Κενών Χαρακ.
- **Η Βιβλιοθήκη <String.h>**
  - Συναρτήσεις: strlen, strcpy, strcat, strcmp
  - Υλοποίηση Συναρτήσεων
- **Πίνακες Δεικτών (και Συμβολοσειρών)**
  - Ορίσματα Προγράμματος *argv[], argc
  - Παραδείγματα Επεξεργασίας

Dr. Andreas Aristidou a.aristidou@ieee.org

String Literal (or Constant)

Σταθερά Συμβολοσειράς

- Strings είναι πίνακες χαρακτήρων στους οποίους ένας ειδικός χαρακτήρας — ο χαρακτήρας null — σηματοδοτεί το τέλος.
- Η βιβλιοθήκη C παρέχει μια συλλογή από συναρτήσεις για χρήση σε συμβολοσειρές.
- **Σταθερά Συμβολοσειράς (String Literal):** ακολουθία χαρακτήρων που εσωκλείεται από **δίπλα εισαγωγικά (")** και τερματίζεται αυτόματα από τον χαρακτήρα **NUL ('\0'):**

```
"Candy\nIs dandy\nBut liquor\n...  --Ogden Nash\n"
```

```
Candy
Is dandy
But liquor
Is quicker.
  --Ogden Nash
```

  - Το NUL ('\0') δεν δίνεται στον ορισμό της σταθεράς
  - Σεβόμενοι τον κανόνα 80 στηλών κατά την εκτύπωση …

```
printf("When you come to a fork in the road, take it. \
--Yogi Berra");
// ή printf("When you come to a fork in the road, take it.  "
                                "--Yogi Berra");
```

Ο χαρακτήρας \ μπορεί να χρησιμοποιηθεί για τη ένωση δύο ή περισσότερων γραμμών ενός προγράμματος σε μία γραμμή.

Όταν δύο ή περισσότερες συμβολοσειρές είναι συνεχόμενες, τότε θα τους ενωθούν σε μια.

Dr. Andreas Aristidou a.aristidou@ieee.org

String Literal (or Constant)

Σταθερά Συμβολοσειράς

- Όταν ένας μεταγλωττιστής C συναντά ένα αλφαριθμητικό συμβολοσειράς μήκους `n` σε ένα πρόγραμμα, θέτει στην άκρη `n + 1` byte μνήμης για τη συμβολοσειρά.
- Αυτή η μνήμη θα περιέχει τους χαρακτήρες στη συμβολοσειρά, συν έναν επιπλέον χαρακτήρα—τον ***null character***—για να επισημάνει το τέλος της συμβολοσειράς.
- Ο null character είναι ένα byte του οποίου τα bit είναι όλα μηδέν, επομένως αντιπροσωπεύεται από την ακολουθία διαφυγής `\0`.

Dr. Andreas Aristidou a.aristidou@ieee.org

Συμβολοσειρά (Ο Χαρακτήρας NUL \0)

ASCII Code Chart

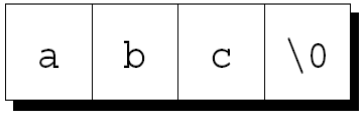
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Dr. Andreas Aristidou a.aristidou@ieee.org

String Literal (or Constant)

Σταθερά Συμβολοσειράς

- Δεδομένου ότι μια συμβολοσειρά αποθηκεύεται ως πίνακας, το πρόγραμμα μεταγλώττισης το αντιμετωπίζει ως δείκτη τύπου `char *`
- **`char *p; p = "abc"; /*** ΣΩΣΤΟ ***/`**
  - Το p δείχνει πάνω στον πρώτο χαρακτήρα της σταθεράς "abc".
  - Ορθός ορισμός, εφόσον το `"abc"` είναι ήδη στην μνήμη



- Οι συμβολοσειρές μπορούν να έχουν δείκτη:

```
char ch;
ch = "abc"[1];
```

  Η νέα τιμή του `ch` θα είναι το γράμμα `b`.

- Μια συνάρτηση που μετατρέπει έναν αριθμό μεταξύ 0 και 15 στο ισοδύναμο δεκαεξαδικό ψηφίο:

```
char digit_to_hex_char(int digit)
{
        return "0123456789ABCDEF"[digit];
}
```

String Literal (or Constant)

Σταθερά Συμβολοσειράς

- Η προσπάθεια τροποποίησης μιας συμβολοσειράς προκαλεί απροσδιόριστη συμπεριφορά:
  - **`char *p = "abc";`**
  - **`*p = 'd'; /*** ΛΑΘΟΣ – Δεν επιτρέπεται να μεταβάλλεται μια σταθερά (το “abc”) ***/`**

- `char *p; p[1]= 'a'; /*** ΛΑΘΟΣ – p δεν δεσμεύτηκε χώρος***/`

Dr. Andreas Aristidou a.aristidou@ieee.org

Σταθερά Συμβολοσειράς & Σταθερά Χαρακτήρα

- Μια **σταθερά συμβολοσειράς (string literal)** η οποία περιέχει ένα χαρακτήρα **ΔΕΝ είναι το ίδιο** με μια **σταθερά χαρακτήρα (character constant).**
  - `"a"` αναπαριστάται από ένα δείκτη σε char (τερματίζεται με NUL καταλαμβάνοντας 2 bytes) .
  - `'a'` αναπαριστάται από ένα χαρακτήρα (και καταλαμβάνοντας 1 byte)
- Επομένως:

```
printf("\n");    /*** ΣΩΣΤΟ ***/
printf('\n');    /*** ΛΑΘΟΣ ***/
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Μεταβλητές Συμβολοσειράς (String Variables)

- Οποιοσδήποτε μονοδιάστατος πίνακας χαρακτήρων μπορεί να χρησιμοποιηθεί για την αποθήκευση μιας συμβολοσειράς.
- Μια συμβολοσειρά πρέπει να τερματιστεί από έναν μηδενικό χαρακτήρα.
- Δυσκολίες με αυτή την προσέγγιση:
  - Μπορεί να είναι δύσκολο να διαπιστωθεί αν ένας πίνακας χαρακτήρων χρησιμοποιείται ως συμβολοσειρά.
  - Οι συναρτήσεις χειρισμού συμβολοσειρών πρέπει να είναι προσεκτικοί για να αντιμετωπίσουν σωστά τον μηδενικό χαρακτήρα.
  - Η εύρεση του μήκους μιας συμβολοσειράς απαιτεί αναζήτηση για τον μηδενικό χαρακτήρα.
- **Μεταβλητή Συμβολοσειράς:** Μια μεταβλητή που μπορεί να αναπαραστήσει μια συμβολοσειρά.
  - Για string μήκους N, είναι ένας πίνακας N+1 θέσεων.

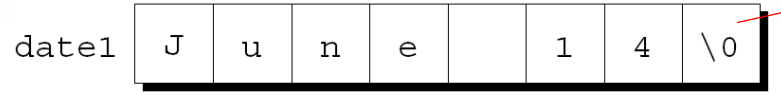
```
#define STR_LEN 80
char str[STR_LEN+1];
```

ΣΗΜΑΝΤΙΚΟ: Η προσθήκη 1 στο μήκος επιτρέπει χώρο για τον μηδενικό χαρακτήρα στο τέλος της συμβολοσειράς

Μεταβλητές Συμβολοσειράς (String Variables)

- **Μεταβλητή Συμβολοσειράς**

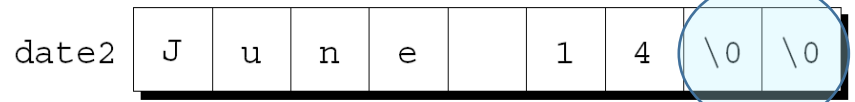
```
char date1[8] = "June 14";
```



date1 **τερματίζεται με NUL '\0'**

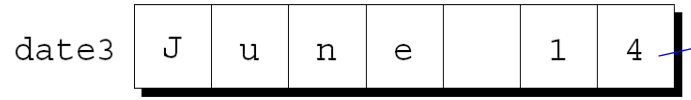
- Επιπλέον NUL προστίθενται από τον μεταγλωττιστή εάν υπάρχει χώρος (για μεταβλητές μόνο)

```
char date2[9] = "June 14";
```



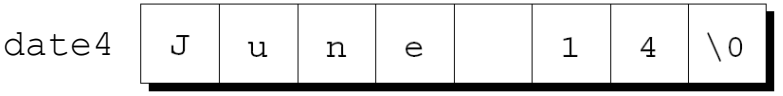
Δεν υπάρχει χώρος για τον μηδενικό χαρακτήρα, οπότε ο μεταγλωττιστής δεν κάνει προσπάθεια να τον αποθηκεύσει

```
char date3[7] = "June 14";
```



- Η δήλωση μιας μεταβλητής συμβολοσειράς μπορεί να παραλείψει το μήκος της, οπότε το πρόγραμμα μεταγλώττισης το υπολογίζει

```
char date4[] = "June 14";
```



Πίνακες χαρακτήρων vs Δείκτες χαρακτήρων

- Η δήλωση

  ```
  char date[] = "June 14";
  ```

  δηλώνει την `date` σαν ένας *πίνακα (array),*
- Η παρόμοια δήλωση

  ```
  char *date = "June 14";
  ```

  δηλώνει την `date` σαν ένα *δείκτη (pointer).*
- Χάρη στη στενή σχέση μεταξύ πινάκων και δεικτών, και οι δυο πιο πάνω δηλώσεις μπορούν να χρησιμοποιηθούν ως συμβολοσειρά.

Dr. Andreas Aristidou a.aristidou@ieee.org

Πίνακες χαρακτήρων vs Δείκτες χαρακτήρων

- Ωστόσο, υπάρχουν σημαντικές διαφορές μεταξύ των δύο εκδόσεων της `date`.
  - Στην έκδοση με τον πίνακα, οι χαρακτήρες που είναι αποθηκευμένοι στο `date` μπορεί να τροποποιηθούν. Στην έκδοση ωστόσο με τον δείκτη, `date` δείχνει σε μια συμβολοσειρά και **δεν** πρέπει να τροποποιηθεί.
  - Στην έκδοση με τον πίνακα, η `date` είναι ένα όνομα πίνακα. Στην έκδοση με τον δείκτη, η `date` είναι μια μεταβλητή που μπορεί να δείχνει σε άλλες συμβολοσειρές.

Πίνακες χαρακτήρων vs Δείκτες χαρακτήρων

- Η δήλωση

  ```
  char *p;
  ```

  δεν εκχωρεί χώρο για μια συμβολοσειρά.
- Πριν μπορέσουμε να χρησιμοποιήσουμε το `p` ως συμβολοσειρά, πρέπει να δείχνει σε έναν πίνακα χαρακτήρων.
- Μια δυνατότητα είναι να κάνετε το `p` να δείχνει σε μια μεταβλητή συμβολοσειρά:

  ```
  char str[STR_LEN+1], *p;

  p = str;
  ```
- Μια άλλη δυνατότητα είναι να κάνετε το `p` να δείχνει σε μια συμβολοσειρά που έχει εκχωρηθεί δυναμικά.

Πίνακες χαρακτήρων vs Δείκτες χαρακτήρων

- Η χρήση μιας μη αρχικοποιημένης μεταβλητής δείκτη ως συμβολοσειράς είναι ένα σοβαρό σφάλμα.
- Π.χ. για να δημιουργήσω το "abc":

```
char *p;

p[0] = 'a';     /*** WRONG ***/
p[1] = 'b';     /*** WRONG ***/
p[2] = 'c';     /*** WRONG ***/
p[3] = '\0';    /*** WRONG ***/
```

- Δεδομένου πως η p δεν έχει αρχικοποιηθεί, αυτό προκαλεί απροσδιόριστη συμπεριφορά.

Μεταβλητές Συμβολοσειράς (Αρχικοποίηση)

literal ← Συνιστώνται αυτοί οι ορισμοί (declarations) → variable

const char *msg="Hello"; ή const char msg[]="Hello";

αλλά υπάρχουν και άλλοι τρόποι…

Σωστό αλλά άκομψο

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

Σωστό αλλά άκομψο

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

Σωστό αλλά άκομψο

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

Σωστό αλλά σπάταλο

```
char msg[40]="Hello";
```

Λάθος (ξεχνάμε το \0)

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

Λάθος (το \0 δεν προστίθεται έλλειψη χώρου)

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

Λάθος (δεν δεσμεύεται χώρος)
ΕΠΙΚΙΝΔΥΝΟ (συχνό λάθος)

```
char *p; p[0] = 'a'; /*** ΛΑΘΟΣ ***/
```

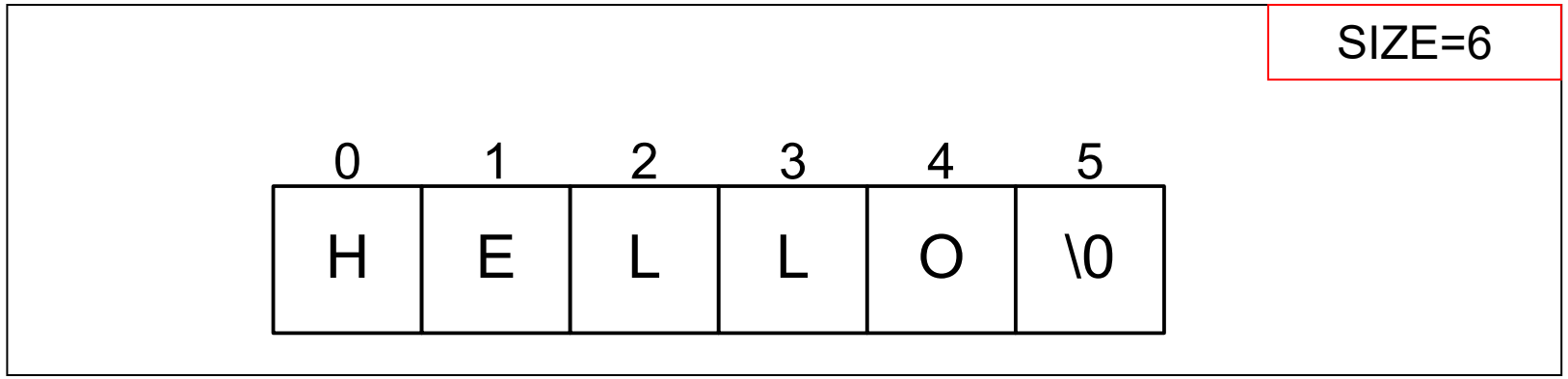

Dr. Andreas Aristidou a.aristidou@ieee.org

Μεταβλητές Συμβολοσειράς (Αρχικοποίηση)

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char msg[10]="Hello";

char msg[ ]="Hello";



Μεταβλητές Συμβολοσειράς (Αρχικοποίηση)

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char c = 'H';

Ο χαρακτήρας H βρίσκεται κάπου στην μνήμη

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

char c[ ]="H";

0	1	2	3	4	5
		H	\0		

Το String H βρίσκεται κάπου στην μνήμη

char c="H";

ΛΑΘΟΣ στην μεταγλώττιση

Ανάγνωση/Εκτύπωση String

- Δεδομένου ότι μια συμβολοσειρά αποθηκεύεται ως πίνακας, το πρόγραμμα μεταγλώττισης το αντιμετωπίζει ως δείκτη τύπου `char *`.
- Τόσο η `printf` όσο και η `scanf` αναμένει μια τιμή τύπου `char *` ως το πρώτο τους στοιχείο.
- Η ακόλουθη κλήση της `printf` μεταβιβάζει τη διεύθυνση του `"abc"` (ένα δείκτη στο σημείο όπου είναι αποθηκευμένο το γράμμα `a` στη μνήμη):

```
printf("abc");
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Εκτύπωση String

- Η εκτύπωση μιας συμβολοσειράς είναι εύκολη χρησιμοποιώντας είτε το `printf` ή το `puts`.
- Το `%s` επιτρέπει στην `printf` να εκτυπώσει μια συμβολοσειρά:

```
char str[] = "Are we having fun yet?";
printf("%s\n", str);
```

Η έξοδος θα είναι

```
Are we having fun yet?
```

- Η `printf` εκτυπώνει ένα-ένα τους χαρακτήρες μιας συμβολοσειράς μέχρι να συναντήσει έναν μηδενικό χαρακτήρα.
- Για να εκτυπώσετε μέρος μιας συμβολοσειράς, χρησιμοποιήστε την `%.`*p*`s`, όπου *p* είναι ο αριθμός των χαρακτήρων που θα εμφανιστούν.
- Η δήλωση

```
printf("%.6s\n", str);
```

θα εκτυπώσει

```
Are we
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Εκτύπωση String

- Το `%`*m*`s` θα εμφανίσει μια συμβολοσειρά σε ένα πεδίο μεγέθους *m*.
- Εάν η συμβολοσειρά έχει λιγότερους από *m* χαρακτήρες, θα ευθυγραμμιστεί στα δεξιά μέσα στο πεδίο.
  - Αριστερή στοίχιση θα επιτευχθεί χρησιμοποιώντας το -*m*.
  - Τα *m* και *p* μπορούν να χρησιμοποιηθούν σε συνδυασμό.
  - Το `%`*m*`.`*p*`s` εκτυπώνει τους πρώτους *p* χαρακτήρες μιας συμβολοσειράς σε ένα πεδίο μεγέθους *m*.
- Η `printf` δεν είναι η μόνη λειτουργία που μπορεί να γράψει συμβολοσειρές.
- Η βιβλιοθήκη C παρέχει επίσης την `puts`:

```
puts(str);
```

- Αφού εκτυπώσει μια συμβολοσειρά, η `puts` γράφει έναν χαρακτήρα νέας γραμμής.

Dr. Andreas Aristidou a.aristidou@ieee.org

Ανάγνωση String

- Η ανάγνωση μιας συμβολοσειράς είναι λίγο δυσκολότερη, επειδή η είσοδος μπορεί να είναι μεγαλύτερη από τη μεταβλητή συμβολοσειράς στην οποία αποθηκεύεται.
- Για να διαβάσετε μια συμβολοσειρά σε ένα μόνο βήμα, μπορούμε να χρησιμοποιήσουμε είτε την `scanf` ή την `gets`.
- Το `%s` επιτρέπει στην `scanf` να διαβάσει μια συμβολοσειρά σε έναν πίνακα χαρακτήρων:

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

  - Το **`str`** αντιμετωπίζεται ως δείκτης, επομένως δεν χρειάζεται να τοποθετήσετε το `&` πριν το `str`.

Dr. Andreas Aristidou a.aristidou@ieee.org

Ανάγνωση String

- Όταν καλείται η `scanf`, παρακάμπτει το κενό διάστημα, στη συνέχεια διαβάζει τους χαρακτήρες και τους αποθηκεύει στο `str` μέχρι να συναντήσει ξανά ένα χαρακτήρα κενού διαστήματος.
- Η `scanf` αποθηκεύει πάντα έναν μηδενικό χαρακτήρα στο τέλος της συμβολοσειράς.
- **Για εισαγωγή συμβολοσειράς με κενά χρησιμοποιείται η** `gets`
  - Μην παραλείψετε το κενό διάστημα πριν ξεκινήσετε την ανάγνωση εισόδου.
  - Διαβάζει μέχρι να βρει ένα χαρακτήρα νέας γραμμής.
  - Απορρίπτει τον χαρακτήρα νέας γραμμής αντί να τον αποθηκεύει και χρησιμοποιεί τον κενό χαρακτήρας.

Ανάγνωση String

- Εξετάστε το ακόλουθο τμήμα προγράμματος:

```
char sentence[SENT_LEN+1];
printf("Enter a sentence:\n");
scanf("%s", sentence);
```

- Ας υποθέσουμε ότι μετά την προτροπή

```
Enter a sentence:
```

ο χρήστης γράφει

```
  To C, or not to C: that is the question.
```

- Η `scanf` θα αποθηκεύσει τη συμβολοσειρά `"To"` στη `sentence`.
- Ας υποθέσουμε ότι αντικαταστήσαμε την `scanf` με `gets`:

```
gets(sentence);
```

- Όταν ο χρήστης γράψει ό,τι και πιο πάνω, η `gets` θα αποθηκεύσει την συμβολοσειρά

```
"  To C, or not to C: that is the question."
```

στην `sentence`.

Dr. Andreas Aristidou a.aristidou@ieee.org

Ανάγνωση String

- Καθώς διαβάζουν χαρακτήρες σε έναν πίνακα, οι `scanf` και `gets` δεν έχει τρόπο να το ανιχνεύσουν όταν είναι πλήρης.
- Κατά συνέπεια, μπορούν να αποθηκεύουν χαρακτήρες μετά το τέλος του πίνακα, προκαλώντας απροσδιόριστη συμπεριφορά.
- Η `scanf` μπορούν να γίνει ασφαλής με τη χρήση του `%`*n*`s` αντί του `%s`.
  - *n* είναι ένας ακέραιος που υποδεικνύει το μέγιστο αριθμό χαρακτήρων που θα αποθηκευτούν.
- Η `gets` είναι μη ασφαλές; η `fgets` είναι μια πολύ καλύτερη επιλογή.

Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

Σάρωση Συμβολοσειρών

- Γράψετε μια ασφαλή συνάρτηση σάρωσης εισόδου στη γλώσσα C, η οποία να διαβάζει το περιεχόμενο της εισόδου σε μεταβλητή πίνακα `str` με τις εξής συνθήκες:
  - (1) Δεν διαγράφει τους white-space χαρακτήρες
  - (2) Σταματά μόλις διαβάσει τον πρώτο χαρακτήρα γραμμής `\n` (που δεν ανήκει στο str) ή εάν έχει διαβάσει `n` χαρακτήρες
  - (3) Αγνοεί οποιουσδήποτε χαρακτήρες μετά το `\n`
  - (4) Επιστρέφει τον αριθμό των χαρακτήρων που αναγνώστηκαν
- Πρότυπο Συνάρτησης:

```
int read_line(char str[], int n);
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

Σάρωση Συμβολοσειρών

```
int read_line(char str[], int n)     {

  int ch, i = 0;

  while ((ch = getchar()) != '\n') {
    if (i < n) {
      str[i] = ch;
        i++;
      }
  }
  str[i] = '\0';    /* terminates string */
  return i;         /* # of chars stored */
}
```

Τυπικές λειτουργίες, όπως η `scanf` και `gets` θέτει αυτόματα έναν μηδενικό χαρακτήρα στο τέλος μιας συμβολοσειράς εισόδου. Αν γράφουμε τη δική μας λειτουργία εισόδου, πρέπει να αναλάβουμε εμείς αυτή την ευθύνη.

- Το `ch` έχει τύπο `int` παρά `char` εφόσον `getchar` επιστρέφει ακέραια τιμή `int`.

Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

Μέτρηση Κενών σε Συμβολοσειρά

Γράψετε μια συνάρτηση στη γλώσσα C, η οποία μετρά και επιστρέφει τον αριθμό των spaces σε μια συμβολοσειρά εισόδου str

Πρότυπο Συνάρτησης:

int count_spaces(const **char s[]**)

```
int count_spaces(const char s[])
{
  int count = 0, i;

  for (i = 0; s[i] != '\0'; i++){
    if (s[i] == ' ') {
      count++;
    }
  }
  return count;
}
```

Θα μπορούσαν να αντιστραφούν

```
int count_spaces(const char *s){
  int count = 0;

  for (; *s != '\0'; s++) {
    if (*s == ' ') {
      count++;
    }
  }
  return count;
}
```

Λύση με αριθμητική δεικτών

a.aristidou@ieee.org

Παράδειγμα

Μέτρηση Κενών σε Συμβολοσειρά

- Ερωτήσεις που τίθενται από την `count_spaces`:
  - *Είναι προτιμότερο να χρησιμοποιείτε λειτουργίες πίνακα ή δείκτη για να έχετε πρόσβαση στους χαρακτήρες σε μια συμβολοσειρά;*
    - Μπορούμε να χρησιμοποιήσουμε και τα δύο. Παραδοσιακά, οι προγραμματιστές C τείνουν προς τη χρήση λειτουργιών δείκτη.
  - *Πρέπει να δηλωθεί μια παράμετρος συμβολοσειράς ως πίνακας ή ως δείκτης;*
    - Δεν υπάρχει διαφορά μεταξύ των δύο.
  - *Η μορφή της παραμέτρου (`s[]` or `*s`) επηρεάζει το τι μπορεί να δοθεί ως είσοδος;*
    - Όχι.

Dr. Andreas Aristidou a.aristidou@ieee.org

C String Library

Η Βιβλιοθήκη <string.h>

- Εφόσον τα Strings είναι πίνακες χαρακτήρων στη C, αυτά **ΔΕΝ μπορούν** να **αντιγραφούν**, **συγκριθούν**, κτλ με **απλούς τελεστές** (π.χ., =, ==, κτλ) που μάθαμε για τους άλλους τύπους.
- **Συνηθισμένο λάθος:**

```
char str1[10], str2[10];
str1 = "abc";  /*** ΛΑΘΟΣ μεταγλώττισης ***/
str2 = str1;   /*** ΛΑΘΟΣ μεταγλώττισης ***/
if (str1 == str2) …    /*** ΟΧΙ αναμενόμενο αποτέλεσμα, συγκρίνει τις διευθύνσεις των str1,str2 επιστρέφοντας FALSE 0) ***/
```

```
typedef unsigned long int ADDR;
printf("%lx %lx", (ADDR)str1, (ADDR)str2);
7ffffa0f 7ffffa0e
```

- Ορθό (πρόκειται για ορισμό – declaration - όχι ανάθεση)

```
char str1[10] = "abc";      // OK
```

- Ορθό: `if (str1[0] == str2[0]) // OK`

Dr. Andreas Aristidou a.aristidou@ieee.org

C String Library

Η Βιβλιοθήκη <string.h>

- Η Βιβλιοθήκη <string.h> περιέχει ένα μεγάλο σύνολο λειτουργιών επεξεργασίας συμβολοσειρών.

  #include <string.h>(strcpy,strcat,strlen,strcmp,..)

- **Αντιγραφή (Copy)** του string s2 στο s1 (return: διεύθυνση του s1)

```
char *strcpy(char *s1, const char *s2);
```

  - Ασφαλής Αντιγραφή του `str2` στο `str1` (εάν s2 μεγαλύτερο s1):

```
strncpy(str1, str2, sizeof(str1) - 1); str1[sizeof(str1)-1] = '\0';
```

- **Εύρεση Μήκους (Length)** συμβολοσειράς (δεν μετρά το '\0')

```
size_t strlen(const char *s);  // sizeof περιλαμβάνει '\0' και πέρα (π.χ., εάν s[10] = "abc")
```

  - `size_t` είναι ένα `typedef` όνομα της C για unsigned integer τύπους.

- **Επικόλληση (Append) S2 στο S1:**

```
char *strcat(char *s1, const char *s2);
ΠΡΟΣΟΧΗ: char str1[6] = "abc"; strcat(str1, "def");   /*** ΕΠΙΚΙΝΔΥΝΟ ***/
ΑΣΦΑΛΗΣ: strncat(str1, str2, sizeof(str1) - strlen(str1) - 1);
```

Εάν str1[100] δίνει 100

δίνει 3

Διαθέσιμος χώρος στο str1

Η Συνάρτηση `strcpy`

- Για εξάσκηση, θα υλοποιήσουμε τώρα κάποιες από τις συναρτήσεις της `string.h`.
  - Το βιβλίο περιέχει διαφορετικές εκδόσεις των υλοποιήσεων για κάθε μια από τις συναρτήσεις.
- Η συνάρτηση `char *strcpy(char *s1, const char *s2)` αντιγράφει τη συμβολοσειρά που δείχνει το `s2` στον πίνακα που δείχνει η `s1`.
- π.χ.

```
strcpy(str2, "abcd");
  /* str2 now contains "abcd" */
```

- Μια κλήση που αντιγράφει τα περιεχόμενα του `str2` στο `str1`:

```
strcpy(str1, str2);
  /* str1 now contains "abcd" */
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Η Συνάρτηση `strcpy`

- Κατά την κλήση της `strcpy(str1, str2)`, η `strcpy` δεν έχει τρόπο να ελέγξει ότι η συμβολοσειρά `str2` θα χωρέσει στον πίνακα που δείχνει η `str1`.
  - Εάν δεν υπάρχει χώρος, εμφανίζεται απροσδιόριστη συμπεριφορά.
- Η χρήση της συνάρτηση `strncpy` είναι πιο ασφαλής, αλλά πιο αργή.
- Η `strncpy` έχει μια επιπλέον είσοδο που περιορίζει τον αριθμό των χαρακτήρων που θα αντιγραφούν.
- π.χ.:

```
strncpy(str1, str2, sizeof(str1));
```

- Μια πιο σωστή κλήση της `strncpy` είναι η ακόλουθη:

```
strncpy(str1, str2, sizeof(str1) - 1);
str1[sizeof(str1)-1] = '\0';
```

- Η δεύτερη δήλωση εγγυάται ότι η `str1` τερματίζει πάντα με **NUL**.

Dr. Andreas Aristidou a.aristidou@ieee.org

Υλοποίηση Συναρτήσεων

Η Συνάρτηση `strlen`

- Η συνάρτηση `strlen(s1)` μετρά το μήκος μιας συμβολοσειράς (χωρίς το NUL)

```
int len;

len = strlen("abc");  /* len is now 3 */
len = strlen("");     /* len is now 0 */
strcpy(str1, "abc");
len = strlen(str1);   /* len is now 3 */
```

Έκδοση Α

Ίδιο με (*s != 0) ή (*s) // 1 TRUE

```
size_t mystrlen(const char *s) {
    size_t n;
    for (n = 0; *s != '\0'; s++)
        n++;
    return n;
}
```

Έκδοση Β

```
size_t mystrlen(const char *s) {
    size_t n = 0;
    while (*s != '\0') {
        s++; n++;
    }
    return n;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

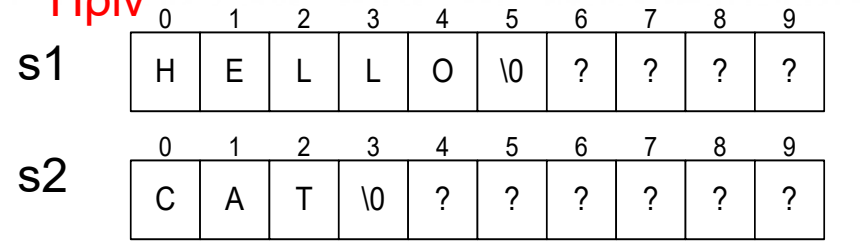
Η Συνάρτηση `strcat`

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

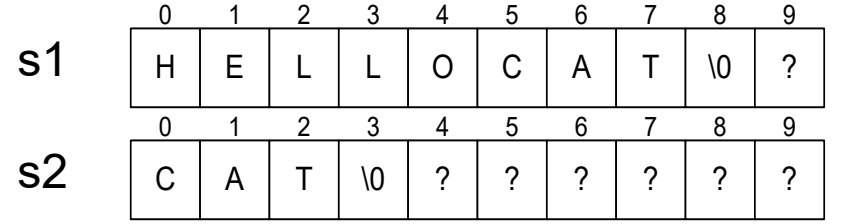
π.χ.,

```
int main() {
        char ma[10]="HELLO";
        char mb[10]="CAT";
        strcat(ma,mb);
        return 0;
    }
```

Πρίν



Μετά



Αλγόριθμος

- Βρές τον χαρακτήρα NUL στο τέλος του `s1` και κάνε τον δείκτη `p` να δείχνει στο σημείο αυτό.
- Αντίγραψε ένα-ένα τους χαρακτήρες από το `s2` ξεκινώντας από το σημείο που δείχνει το `p`.
- Δεν μελετούμε ακόμη το θέμα της υπερχείλισης του s1.

Dr. Andreas Aristidou a.aristidou@ieee.org

Η Συνάρτηση `strcat`

- Η χρήση της συνάρτηση `strncat` είναι πιο ασφαλής, αλλά πιο αργή.
- Μια κλήση της `strncat`:

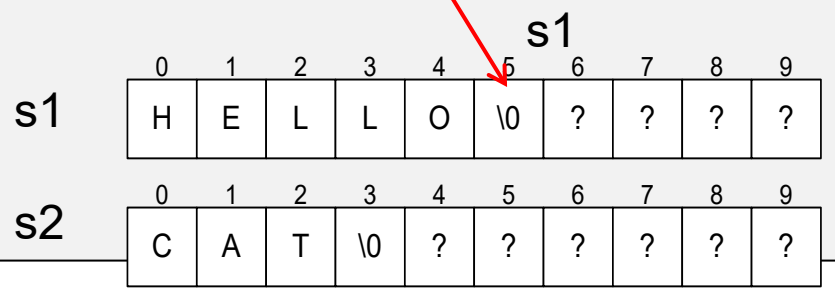
```
strncat(str1, str2, sizeof(str1) - strlen(str1) - 1);
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Υλοποίηση Συναρτήσεων

Η Συνάρτηση `strcat`

```
void mystrcat(char *s1, const char *s2) {
   if ((s1==NULL)||(s2==NULL)) return;
   while (*s1 != '\0') { // εύρεση NUL
     s1++;
   }
   while (*s2 != '\0') { // Αντιγραφή s2 -> s1
     *s1 = *s2;
     s1++; s2++;
   }
   *s1 = '\0';
  return;
}
```



A) Για λόγους ευκολίας : printf("%s\n", strcat(s1, s2));
B) Θα μπορούσαμε να δεσμεύουμε νέο s1 (με χώρο s1+s2) εάν θέλαμε για να αποφύγουμε το πρόβλημα της υπερχείλισης του s1. Κάτι τέτοιο θα δούμε στη Διάλεξη 9 (με malloc).

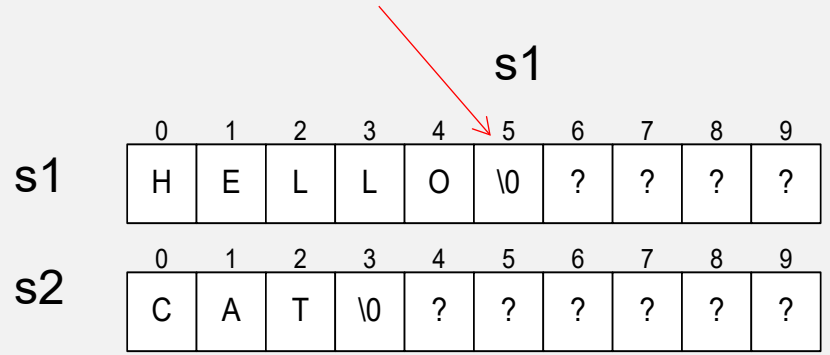
Υλοποίηση Συναρτήσεων

Η Συνάρτηση `strcat`

- Μια πιο συμπαγής έκδοση της `strcat`:

```
void strcat(char *s1, const char *s2)
{
   if ((s1==NULL)||(s2==NULL)) return;

  while (*s1 != '\0') {
    s1++;
  }
   while (*s1 = *s2) { // ανάθεση, όχι σύγκριση
    s1++; s2++;
   }
  return;
}
```

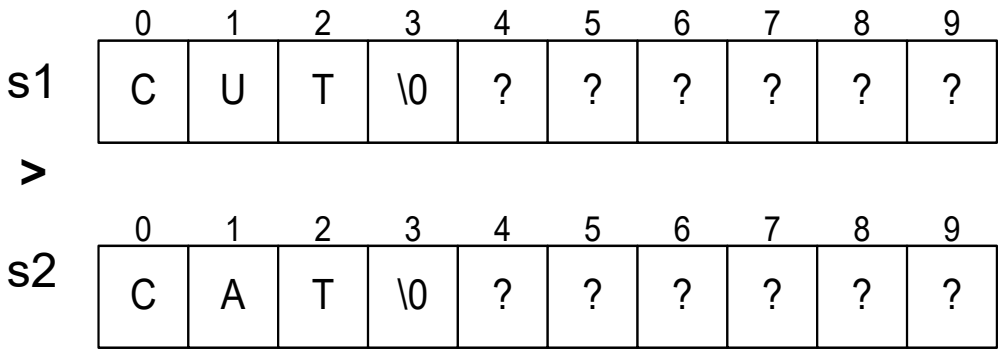


Όταν το `s2` φτάσει το `'\0'` τότε γίνεται `*s1=0`, δηλ., `(*s1=0)` γίνεται 0 (`FALSE`) και διακόπτεται το loop

Όλοι οι χαρακτήρες εκτός από τον null χαρακτήρα επιστρέφουν True.

Η Συνάρτηση `strcmp`

- Γράψετε συνάρτηση που συγκρίνει τα **αλφαριθμητικά** s1 και s2 και επιστρέφει
  - **1:** αν το s1 είναι μεγαλύτερο λεξικογραφικά
  - **0:** αν είναι ίσα (ή το ένα είναι υπο-συμβολοσειρά του άλλου) και
  - **-1:** αν το s2 είναι μεγαλύτερο λεξικογραφικά



Επιστρέφει 1

Η Συνάρτηση `strcmp`

- Η `strcmp` θεωρεί την `s1` να είναι μικρότερη από την `s2` εάν πληρείται μία από τις ακόλουθες προϋποθέσεις:
  - Οι πρώτοι *i* χαρακτήρες του `s1` και `s2` ταιριάζουν, αλλά ο (*i*+1)ος χαρακτήρας του `s1` είναι μικρότερος από τον (*i*+1)ο χαρακτήρα του `s2`.
  - Όλοι οι χαρακτήρες του `s1` ταιριάζουν με του `s2`, αλλά ο `s1` πίνακας είναι μικρότερος από τον `s2`.
  - Η αξία του κάθε χαρακτήρα φαίνεται στο πίνακα ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

A–Z, a–z, and 0–9 have consecutive codes.
All upper-case letters are less than all lower-case letters.
Digits are less than letters.
Spaces are less than all printing characters.

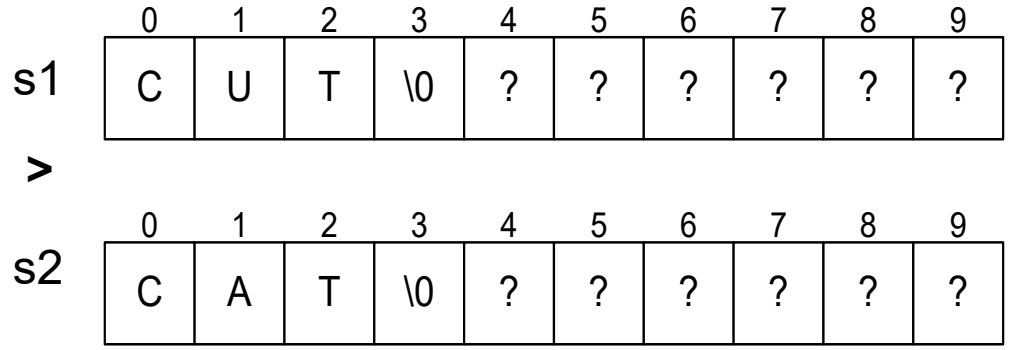
Dr. Andreas Aristidou a.aristidou@ieee.org

Υλοποίηση Συναρτήσεων

Η Συνάρτηση `strcmp`

- **Εκδοχή A (P.J. Plauger, Standard C Library):**

```
int mystrcmp(const char *s1, const char *s2) {
    int i;
    for(i=0; s1[i] == s2[i]; i++) {
        if(s1[i] == 0) // Εάν φτάσαμε τον '\0' χαρακτήρα
            return 0;
   } // χαρακτήρας του s1 δεν είναι ίσος με τον αντίστοιχο του s2
    return (s1[i] < s2[i]) ? -1 : 1;
}
```



Η Συνάρτηση `strcmp`

- **Β) Εκδοχή Β (μόνο δείκτες)**

```
int mystrcmp(const char *s1, const char *s2) {
   if ((s1==NULL) && (s2!=NULL)) return -1;
   else if ((s1!=NULL) && (s2==NULL)) return 1;
   else if ((s1==NULL) && (s2==NULL)) return 0;

   for(; *s1 == *s2; ++s1, ++s2)
        if(*s1 == 0)
            return 0;

     return (*s1 < *s2) ? -1 : 1;
}
```

Arrays of Strings

Πίνακες Συμβολοσειρών

- Υπάρχουν πολλοί τρόποι να αποθηκεύσουμε ένα πίνακα από strings.
- Μια προσέγγιση είναι να χρησιμοποιήσουμε ένα **2-d πίνακα χαρακτήρων** (ένα string ανά γραμμή):

```
char planets[][8] = {
    "Mercury", "Venus", "Earth",
    "Mars", "Jupiter", "Saturn",
    "Uranus", "Neptune", "Pluto"
};
```

Σπάταλη Προσέγγιση!

	0	1	2	3	4	5	6	7
0	M	e	r	c	u	r	y	\0
1	V	e	n	u	s	\0	\0	\0
2	E	a	r	t	h	\0	\0	\0
3	M	a	r	s	\0	\0	\0	\0
4	J	u	p	i	t	e	r	\0
5	S	a	t	u	r	n	\0	\0
6	U	r	a	n	u	s	\0	\0
7	N	e	p	t	u	n	e	\0
8	P	l	u	t	o	\0	\0	\0

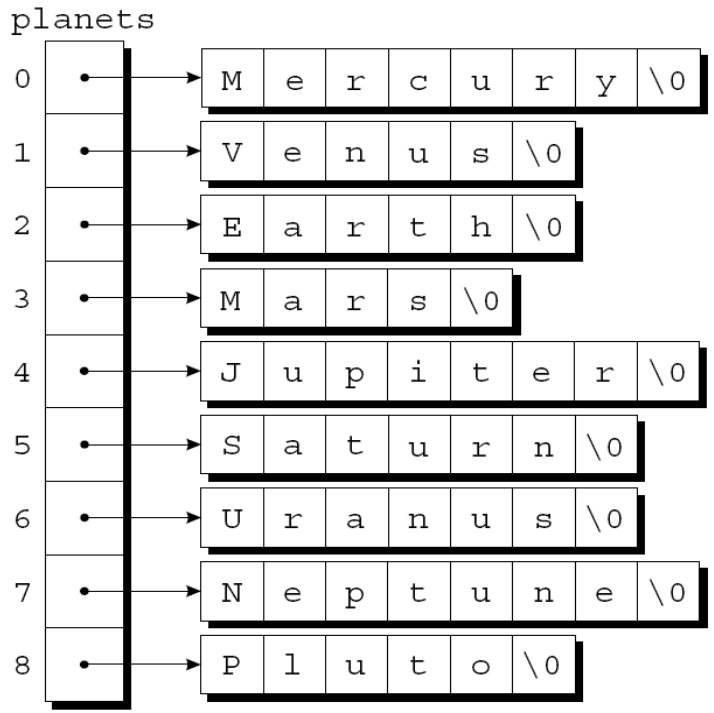
Dr. Andreas Aristidou a.aristidou@ieee.org

Arrays of Pointers

Πίνακες Δεικτών

- Μια καλύτερη προσέγγιση είναι η δημιουργία ενός ***ακανόνιστου πίνακα (ragged array),*** των οποίων οι γραμμές έχουν διαφορετικά μεγέθη.
- Κάτι τέτοιο μπορεί να επιτευχθεί με ένα **πίνακα δεικτών (array of pointers) :**

```
char *planets[] = {
    "Mercury", "Venus", "Earth",
    "Mars", "Jupiter", "Saturn",
    "Uranus", "Neptune","Pluto"
};
```



Αποδοτική Προσέγγιση!

Πίνακες Δεικτών

- Π.χ.
- Ένας βρόγχος που αναζητά στην `planets` συμβολοσειρές που ξεκινούν με το γράμμα M:

```
for (i = 0; i < 9; i++)
  if (planets[i][0] == 'M')
    printf("%s begins with M\n", planets[i]);
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Πολυδιάστατοι Πίνακες vs. Πίνακες Δεικτών

- Έστω

```
int a[10][20]
int *b[10];
```

- **Ποια η διαφορά ανάμεσα στις δυο δηλώσεις;**
  - ο `a` είναι πραγματικά δισδιάστατος πίνακας: κατά τον ορισμό του δεσμεύθηκαν `200` συνεχόμενες θέσεις.
  - Κατά τον ορισμό του `b` κατανέμεται χώρος για `10` δείκτες. Απόδοση αρχικών τιμών πρέπει να **γίνει ρητά** είτε **στατικά** ή με κώδικα (malloc() που θα δούμε αργότερα)
- Πλεονέκτημα ενός **πίνακα με δείκτες** είναι ότι κάθε δείκτης μπορεί να δείχνει σε γραμμή με διαφορετικό μήκος

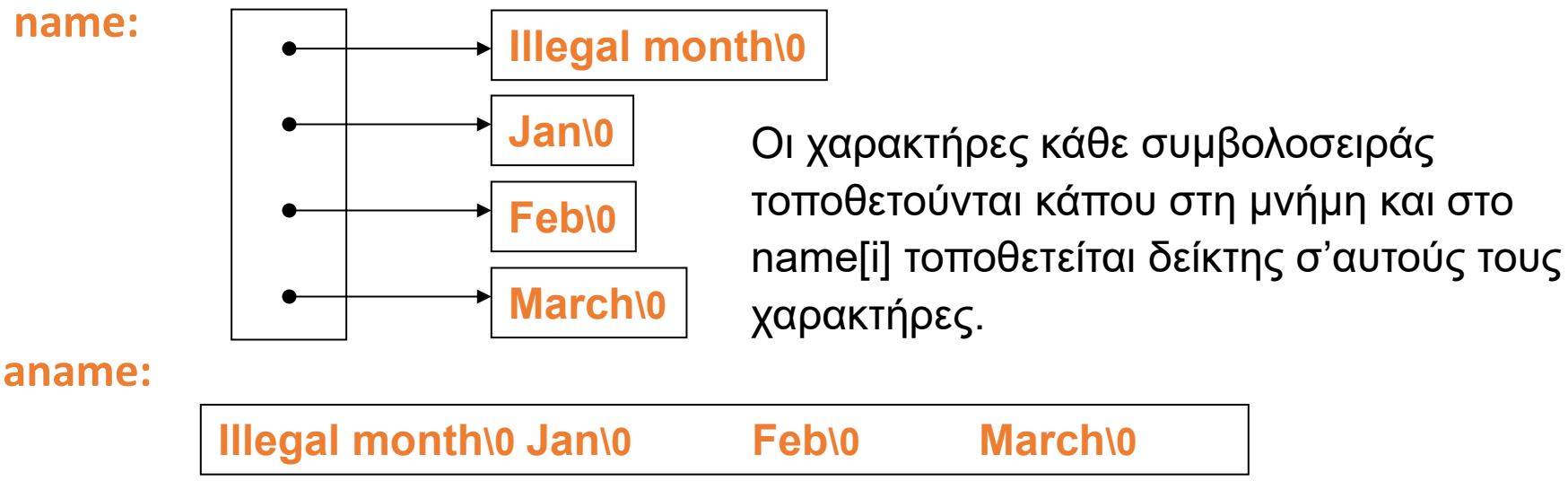
Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

Έστω

```
char  *name[]       =  {"Illegal month", "Jan", "Feb", "March"}
char  aname[] [15] =  {"Illegal month", "Jan", "Feb", "March"}
```

Γραφικά στο επίπεδο της μνήμης έχουμε



Οι χαρακτήρες κάθε συμβολοσειράς τοποθετούνται κάπου στη μνήμη και στο name[i] τοποθετείται δείκτης σ'αυτούς τους χαρακτήρες.

Command-Line Arguments

Ορίσματα Προγράμματος

- Μέχρι τώρα ορίζαμε τη συνάρτηση **main** με παράμετρο void θέλοντας να δείξουμε ότι η συνάρτηση **main** δε δέχεται ορίσματα.

```
int main(void) {
    . . .
}
```

- Αυτό όμως δε σημαίνει ότι δεν μπορούμε να περάσουμε ορίσματα. Τα ορίσματα όμως που μπορεί να πάρει η **main** είναι καθορισμένα και είναι τα εξής:

```
int main(int argc, char *argv[]) {
    . . .
}
```

- Τα ορίσματα περνούνται στο πρόγραμμα από τη γραμμή εντολών τη στιγμή που αρχίζει η εκτέλεσή του ➔ **ορίσματα στη γραμμή εκτέλεσης**.

Dr. Andreas Aristidou a.aristidou@ieee.org

Command-Line Arguments

Ορίσματα Προγράμματος

- Το πρώτο όρισμα **argc**, το οποίο είναι τύπου ακέραιος, είναι ο αριθμός των ορισμάτων της γραμμής διαταγών, με τα οποία έχει κληθεί το πρόγραμμα
  - συμπεριλαμβανομένου και του ονόματος του εκτελέσιμου αρχείου).

- Το δεύτερο όρισμα **argv** είναι δείκτης για έναν πίνακα συμβολοσειρών ο οποίος περιέχει τα ορίσματα. Κατά σύμβαση **(πίνακας δεικτών)**
  - **argv[0]** είναι το όνομα με το οποίο κλήθηκε το πρόγραμμα.
  - **argv[1], …, argv[argc - 1],** είναι τα υπόλοιπα ορίσματα με την σειρά που δόθηκαν στη γραμμή εντολής.
  - **argv[argc]** περιέχει το μηδενικό δείκτη (κενή συμβολοσειρά: “”).

Dr. Andreas Aristidou a.aristidou@ieee.org

Command-Line Arguments

Ορίσματα Προγράμματος

- Παράδειγμα: Έστω ένα πρόγραμμα C το οποίο έχει τη μορφή:

```
int main(int argc, char *argv[]) {
        ……
}
```

- Θεωρείστε επίσης ότι το εκτελέσιμο αρχείο του παραπάνω προγράμματος έχει ονομαστεί **prog**. Τότε κατά την κλήση του **prog** υπό τη μορφή:

$ prog opt1 opt2 opt3

έχουμε την εξής ανάθεση τιμών στα ορίσματα της **main**:

argc = 4

argv[0] = “prog”
argv[1] = “opt1”
argv[2] = “opt2”
argv[3] = “opt3”
argv[4] = NULL

Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

- Ζητούμενο: πρόγραμμα που κατά την κλήση του με `n` ορίσματα (συμπεριλαμβανομένου και του εκτελέσιμου αρχείου) αντηχεί τα `n-1` τελευταία στην οθόνη. Για παράδειγμα αν το πρόγραμμα αυτό λέγεται **`echo`**, τότε μία κλήση της μορφής:

```
$ echo Hello world!
```

θα εμφάνιζε στην έξοδο

```
$ Hello world!
```

- Οι τυπικές παράμετροι της συνάρτησης main θα έχουν τιμές:

```
argc  =  3          argv[0]  =  "echo"
                    argv[1]  =  "Hello"
                    argv[2]  =  "world!"
                    argv[3]  =  ""
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Παράδειγμα

```
#include <stdio.h>

int main (int argc, char *argv[]){

   int i;

   for(i = 1; i < argc; i++)
      printf("%s%s", argv[i],
                  (i < argc - 1) ? " " : "");
   printf("\n");
   return 0;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org