

EEE243 – Applied Computer Programming

Strings

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER
ENGINEERING



GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA



Outline

- Initialization
- Delimiter
- Logical vs. Physical storage
- Useful functions

General Strings Treatment

- Most languages such as Java provide a specific data type for strings
- We understand that a string represents a named “thing” in the environment
- In C we have two ways to store strings
 - Arrays
 - Pointers

General Strings Treatment

- A string is a *logical* unit of storage derived from the *physical* storage *char* type
- What is interesting in strings is that they can and do vary in length
 - Last name ... not all the same length
 - Car brand
 - Color...

Initializing Strings

- Strings can be initialized the same way as other arrays: with size, without size or with a pointer

- `char str[9] = "Good Day";`
- `char str[9] = {'G','o','o','d',' ','D','
 'a','y','\0'};`
- `char month[] = "January";`

month →

J	a	n	u	a	r	y	\0
---	---	---	---	---	---	---	----

"Good Day" →

G	o	o	d		D	a	y	\0
---	---	---	---	--	---	---	---	----

str →

G	o	o	d		D	a	y	\0
---	---	---	---	--	---	---	---	----

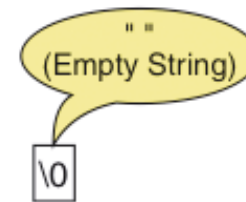
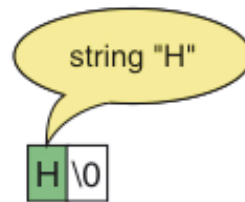
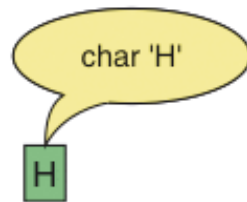
Delimiter Character

- In both variable size strings that we saw, we used the delimiter character `\0`
- `\0` is used as a logical backstop in C to tell our functions that we have reached the end of the string.
- C designers could have chosen any character in the set.
 - ASCII provides for 128 characters
- But the chosen character could not have been used inside a string
 - It can only be used to signify logical end of string

ASCII TABLE

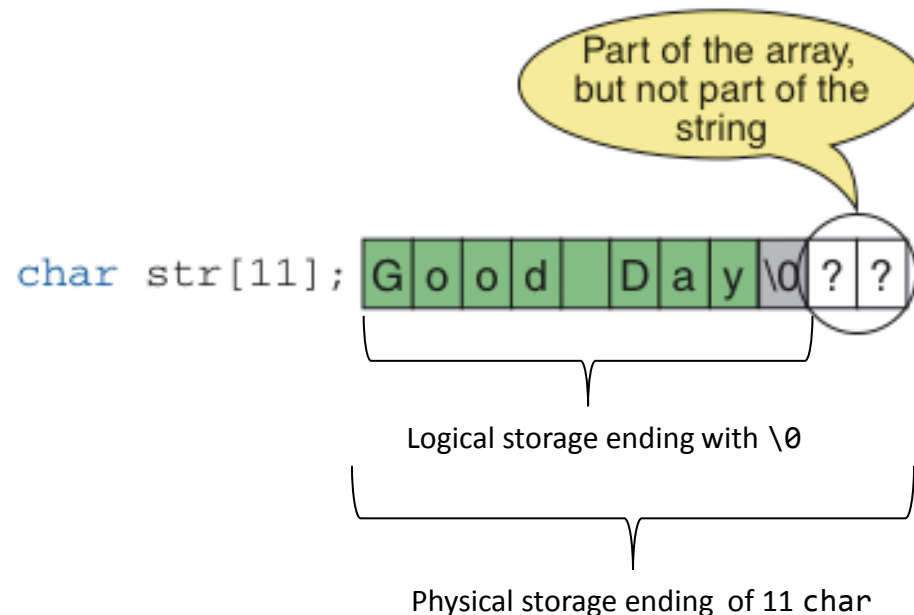
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Effect of the Delimiter Character



Logical and Physical Storage

- We say that a string is a logical derived type
 - There can be a difference between the logical storage of the string (delimited by `\0`) and the physical storage reserved for it



Variable Length Strings - Arrays

Variable length –

- Array of characters with delimiter

```
char last_name[30]; //uninitialized array
// array is physical storage
last_name[0] = 'P';
last_name[1] = 'h';
//...The rest of my name
last_name[7] = 's';
last_name[8] = '\0'; //“null terminator”
// signals the end of the string
// characters to \0 are logical storage
```

Alternately...

Variable length –

- Array of characters with delimiter

```
char last_name[30] = "Phillips";
```

```
//fills first part of character array, null  
terminator '\0' automatically added
```

Some useful functions for strings

From `stdio.h`

`printf`
`scanf`
`gets`
`fgets`
`puts`
`fputs`

From `string.h`

`strcpy`
`strcat`
`strncat`
`strlen`

printf() of a String

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

```
#include <stdlib.h>
```

```
int main(){
```

```
    char str[] = "Hello World!";
```

```
    printf("Message: %s\n", str);
```

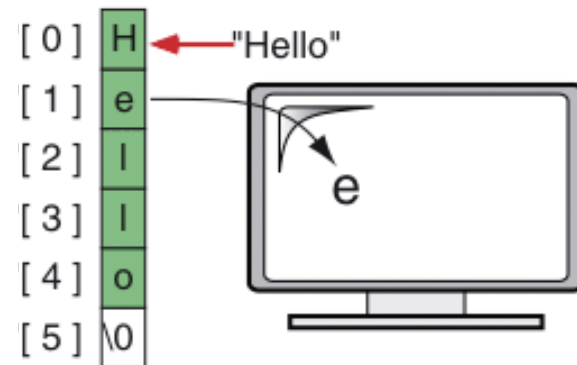
```
    return EXIT_SUCCESS;
```

```
}
```

printf() one char at a time

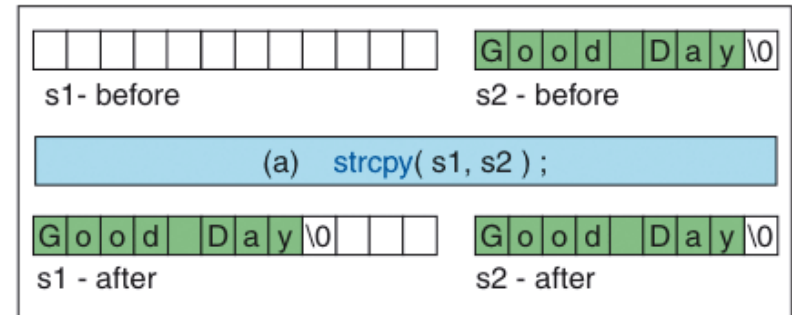
```
#include <stdio.h>

int main() {
    char hello[] = "Hello";
    printf("%c", hello[1]);
    return 0;
}
```

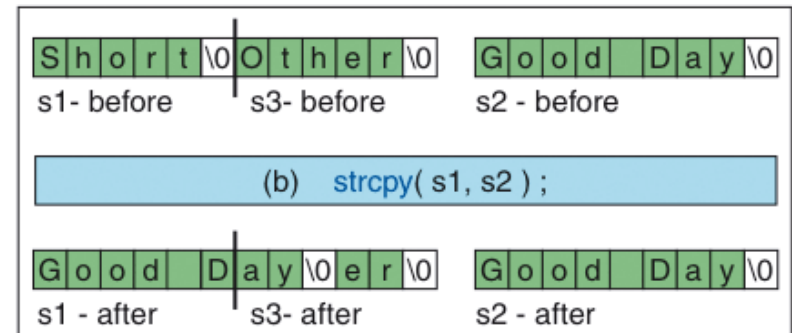


Copying strings with strcpy()

- Use `strcpy()` to copy one string into another
- Beware of string lengths! Use `strncpy()` to copy only the beginning of a string
 - `strncpy(dest, src, n);`



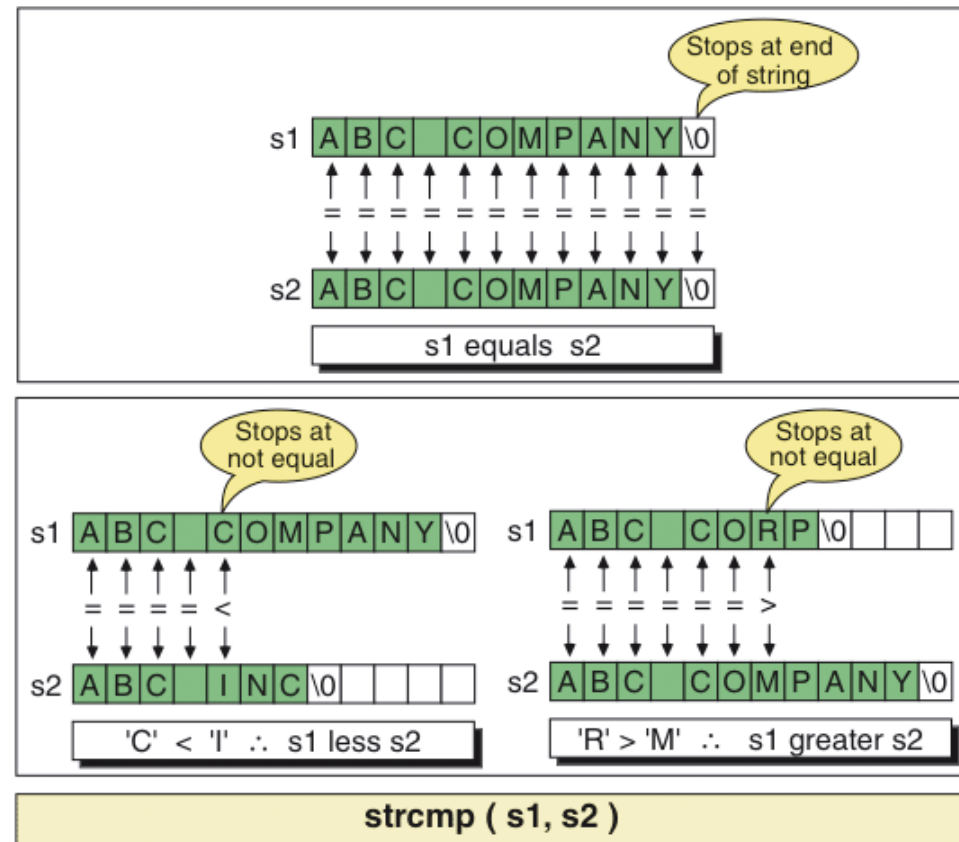
Copying Strings



Copying Long Strings

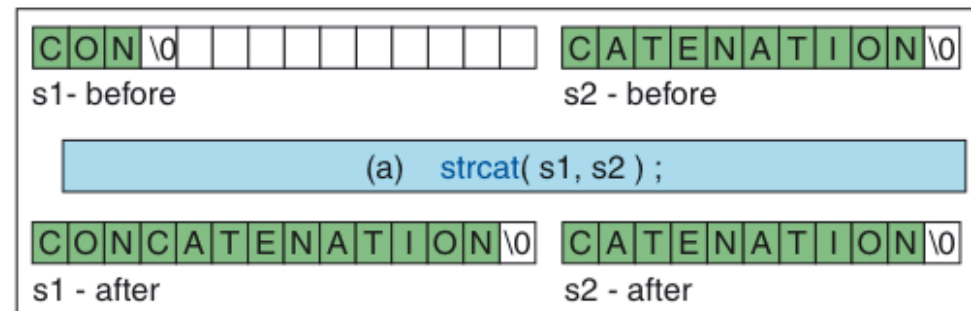
Comparing strings with strcmp()

- `strcmp()` compares two strings, one char at a time.
 - returns 0 if `s1 == s2` which is false in logical terms
 - returns -1 if `s1 < s2`
 - returns 1 if `s1 > s2`

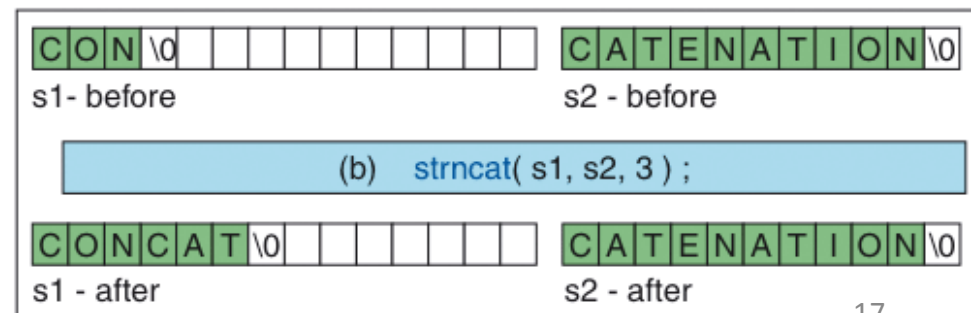


Combining strings with strcat()

- We can concatenate strings with `strcat()`
- We can specify the number of char to concatenate with `strncat()`



String Concatenate



String N Concatenate

Finding the length of a string with `strlen()`

- `strlen()` returns the number of characters in a string before the first '`\0`' (logical storage)
- `sizeof()` will return the number of characters allocated to the array, if the array is declared statically in the same context (physical storage)
- `sizeof` and `strlen` will NOT be the same!!

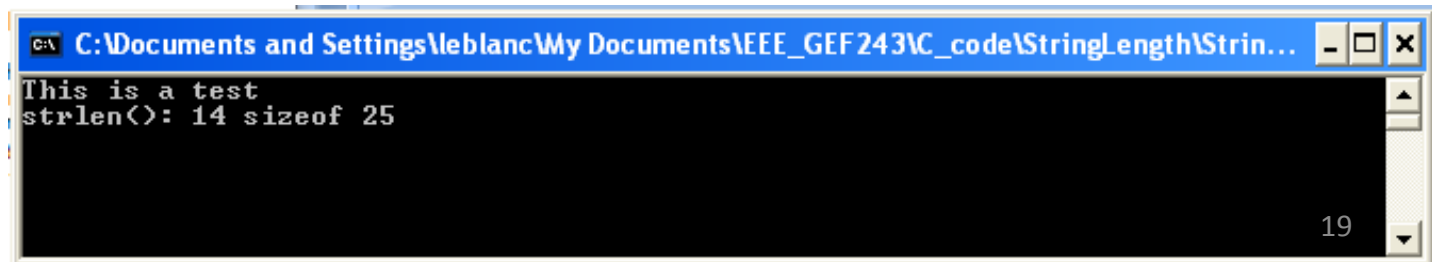
Length of a String

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

int main(int argc, char *argv[])
{
    char test[25] = "This is a test";

    printf("%s\n",test);
    printf("strlen(): %d sizeof  

    %d\n",strlen(test),sizeof(test));
    return 0;
}
```

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Documents and Settings\leblanc\My Documents\EEE_GEF243\C_code\StringLength\Strin...'. The command prompt displays the output of the program: 'This is a test' on the first line and 'strlen(): 14 sizeof 25' on the second line. The window has standard Windows controls (minimize, maximize, close) in the top right corner.

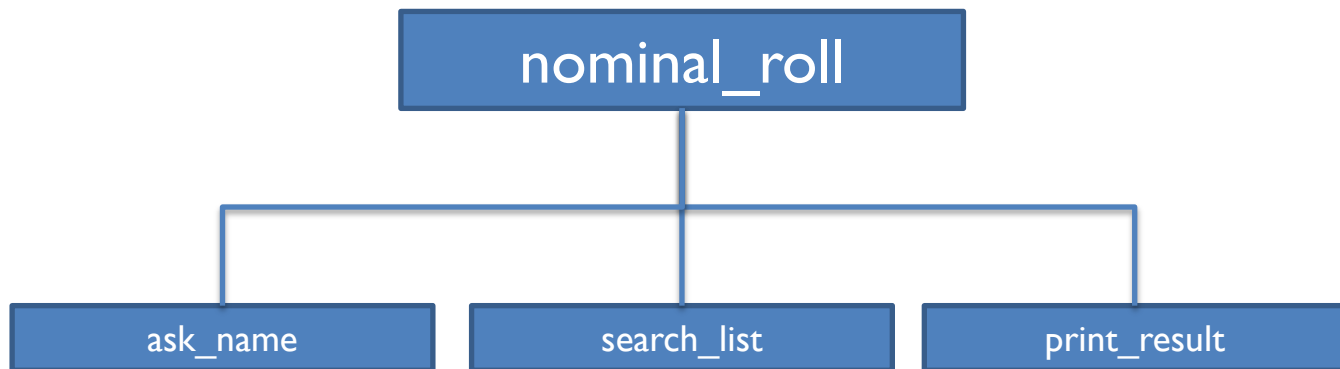
```
C:\Documents and Settings\leblanc\My Documents\EEE_GEF243\C_code\StringLength\Strin...
This is a test
strlen(): 14 sizeof 25
```

Pay Attention

- **None of the string functions check to see if you have enough memory allocated!**
- You can cause a **Kernel Overwrite** on the 3pi if you write to memory using an uninitialized or too short an string!
- In Eclipse/minGW you might get a run-time error if you write beyond the end of the char array!

Exercise

- Write a program that looks through a list of names to see if it contains a name entered by the user.
- Your list of names should have the following names in it: Bob, Joe, Jane, Gill, John
- The following slide has the structure diagram for your program
- Make sure you spend some time thinking about your solution before coding



Questions?