COP 3223: C Programming
Spring 2009

Strings In C – Part 3

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop3223/spr2009/section1

School of Electrical Engineering and Computer Science
University of Central Florida

Another `strcat` Example

- The previous section of notes finished with an example that used the `strcat` function from the `<string.h>` library. We'll start this section of notes with another example that uses the `strcat` function.
- In the previous example, we simply entered two strings and concatenated them together using `strcat`. This example, is similar, expect that different strings are concatenated together depending on a value entered by the user.
- The big difference between the two programs is that in the previous program we simply ignored (did not use) the pointer to the string that was returned by `strcat`. In the next example, this pointer is used.
- Notice that in the `printGreeting` function, I used the value returned by `strcat` in two different ways. In the first case, I printed the returned string using a call to puts, and in the other cases used the implicit pointer as an argument to the `printf` function.

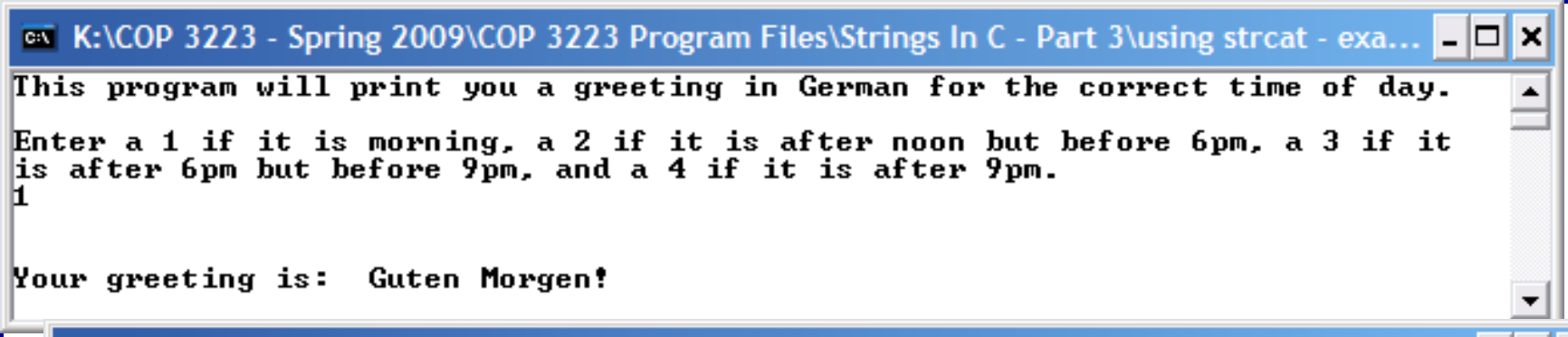
strcat example

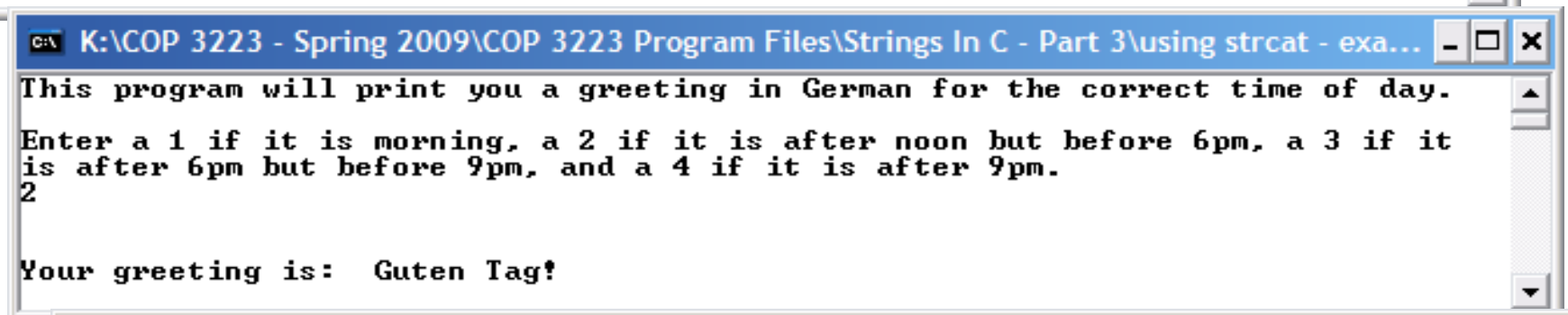
[*] using strcat - example 2.c | arrays of strings - music lyrics.c

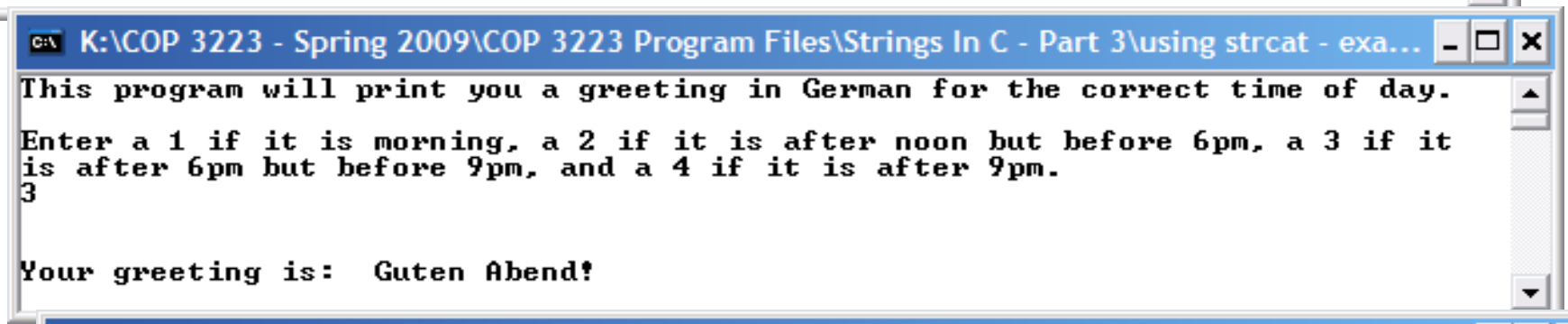
```
 7 void printGreeting(int designator)
 8 {
 9      char string1[] = "Guten";
10      char string2[] = "Gute";
11      char string3[] = " Morgen!";
12      char string4[] = " Tag!";
13      char string5[] = " Abend!";
14      char string6[] = " Nacht!";
15
16      switch (designator) {
17        case 1: strcat(string1, string3);
18                puts(string1);  break;
19        case 2: printf("%s", strcat(string1, string4));  break;
20        case 3: printf("%s", strcat(string1, string5));  break;
21        case 4: printf("%s", strcat(string2, string6));  break;
22     }//end switch stmt
23     return;
24 }//end setGreeting function
25
26 int main()
27 {
28    int tod;  //value for the time of day
29
30     printf("This program will print you a greeting in German for the correct time of d
31     printf("\nEnter a 1 if it is morning, a 2 if it is after noon but before 6pm, a 3
32     printf("is after 6pm but before 9pm, and a 4 if it is after 9pm.\n");
33     scanf("%d", &tod);
34     printf("\n\nYour greeting is:  ");
35     printGreeting(tod);
```

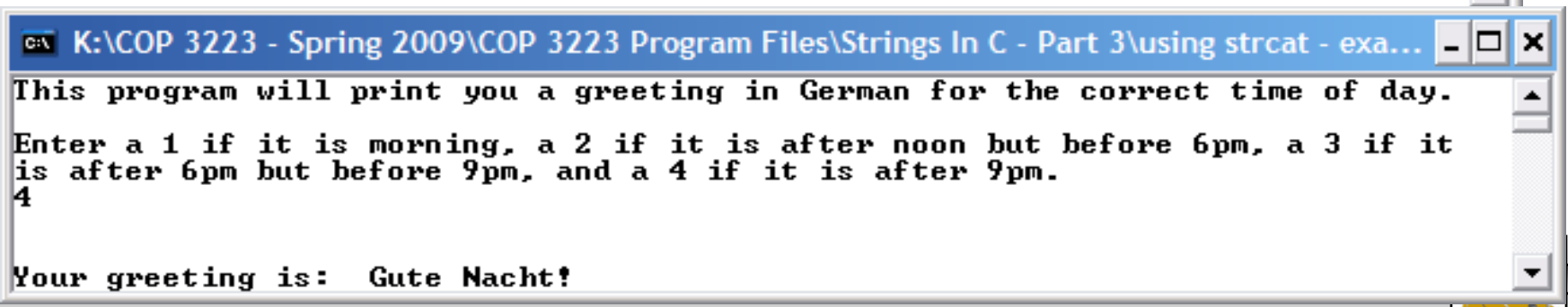
The value returned by `strcat` (a pointer) is used by either the `puts` or `printf` function to reference the string that is to be printed.

NOTE: To get everything to fit on 1 page I did not use good style with the switch statement.


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\using strcat - exa...
This program will print you a greeting in German for the correct time of day.
Enter a 1 if it is morning, a 2 if it is after noon but before 6pm, a 3 if it is after 6pm but before 9pm, and a 4 if it is after 9pm.
1
Your greeting is: Guten Morgen!


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\using strcat - exa...
This program will print you a greeting in German for the correct time of day.
Enter a 1 if it is morning, a 2 if it is after noon but before 6pm, a 3 if it is after 6pm but before 9pm, and a 4 if it is after 9pm.
2
Your greeting is: Guten Tag!


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\using strcat - exa...
This program will print you a greeting in German for the correct time of day.
Enter a 1 if it is morning, a 2 if it is after noon but before 6pm, a 3 if it is after 6pm but before 9pm, and a 4 if it is after 9pm.
3
Your greeting is: Guten Abend!


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\using strcat - exa...
This program will print you a greeting in German for the correct time of day.
Enter a 1 if it is morning, a 2 if it is after noon but before 6pm, a 3 if it is after 6pm but before 9pm, and a 4 if it is after 9pm.
4
Your greeting is: Gute Nacht!

Using the `strcmp` Function

- The `strcmp` function allows for the comparison of two strings. It is not destructive to either of the strings passed to it.
- The `strcmp` function returns:
  - A negative value if the first argument is less than the second argument.
  - A zero value if the two arguments are equal to each other.
  - A positive value if the first argument is greater than the second argument.
- As it compares the characters from the two strings, `strcmp` uses the numerical codes that represent the characters in the character set being using (commonly ASCII – see chart on last page of this set of notes).

Using the `strcmp` Function

- Notice in the ASCII character set that each of characters in the sequence A-Z, a-z, and 0-9 have consecutive codes.
- Also notice that upper-case letters have codes that are less in value than lower-case letters. A-Z have codes from 65-90 and a-z have codes from 97-122.
- Also, digits have codes lower than alpha characters, with codes between 48-57 representing digits 0-9.
- The space character has code 32, which is less than all other printing characters.
- The program who uses the `strcmp` function needs to be aware of the character set being used and the values therein.
- The program on the following page illustrates the `strcmp` function.

[*] Untitled2 [*] strcmp example.c

```
1  //Strings In C - Part 3 - example using the strcmp function
2  //March 22, 2009     Written by: Mark Llewellyn
3
4  #include <stdio.h>
5  #include <string.h>
6
7  void compareStrings(char *s1, char *s2)
8  {
9       int returnedValue;  //value returned from strcmp
10
11      returnedValue = strcmp(s1, s2);
12      if (returnedValue < 0) {
13          printf("%s  - is less than - %s\n", s1, s2);
14      }
15      else if (returnedValue == 0) {
16          printf("%s  - is equal to  - %s\n", s1, s2);
17      }
18      else if (returnedValue > 0) {
19           printf("%s  - is greater than  - %s\n", s1, s2);
20      }
21      printf("\n\n");
22      return;
23 }//end compareString function
24
```


using strcat - example 2.c | [*] strcmp example.c

```
23
24
25
26 int main()
27 {
28     char *string1 = "Hello there!";
29     char *string2 = "Hi there!";
30     char *string3 = "Hello there!";
31     char *string4 = "hello there!";
32     char *string5 = "Hello!";
33     char *string6 = "hello!";
34     char *string7 = "Hello              there!";
35     char *string8 = "hi hi hi";
36     char *string9 = "hi hi hi hi hi hi";
37
38     compareStrings(string1, string2);
39     compareStrings(string1, string3);
40     compareStrings(string1, string4);
41     compareStrings(string2, string3);
42     compareStrings(string1, string5);
43     compareStrings(string5, string6);
44     compareStrings(string6, string5);
45     compareStrings(string7, string1);
46     compareStrings(string8, string9);
47
48     printf("\n\n");
49     system("PAUSE");
50     return 0;
51 }//end main function
52
```


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strcmp example.exe

```
Hello there!  - is less than - Hi there!

Hello there!  - is equal to  - Hello there!

Hello there!  - is less than - hello there!

Hi there!  - is greater than  - Hello there!

Hello there!  - is less than - Hello!

Hello!  - is less than - hello!

hello!  - is greater than  - Hello!

Hello            there!  - is less than - Hello there!

hi hi hi  - is less than - hi hi hi hi hi hi


Press any key to continue . . .
```

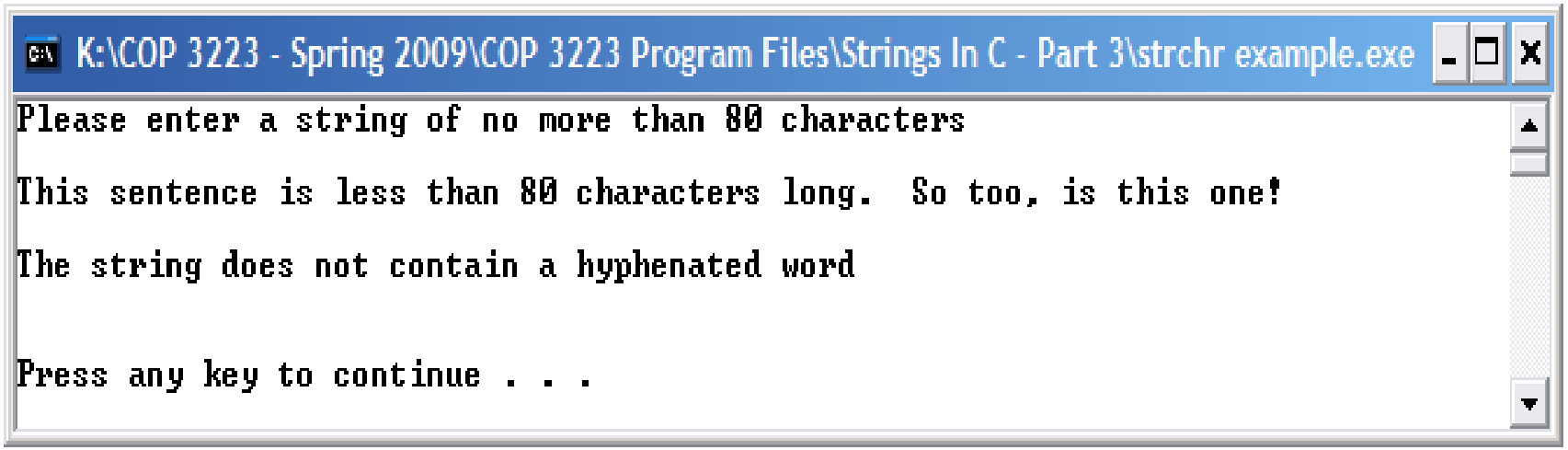

Using the `strchr` Function

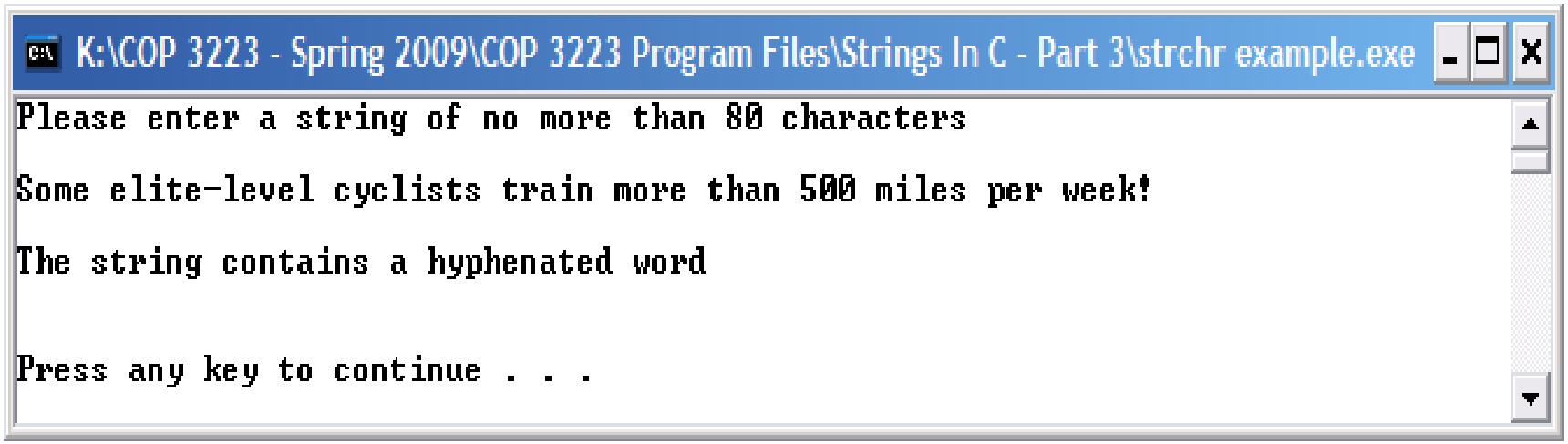
- The `strchr` function allows you to find a specific character (or characters) in a string.
- It requires a string and an character as parameters and is not destructive to the string parameter. If the character is found in the string, a pointer to the character is returned, otherwise a NULL pointer is returned.
- The following program illustrates the use of the `strchr` function by determining if the string passed to it contains a hyphenated word.

strchr example

strchr example.c

```
1  //Strings In C - Part 3 - an example using the strchr function to find a character
2  //in a string - this program looks for a hyphenated word in the string.
3  //March 22, 2009     Written by: Mark Llewellyn
4
5  #include <stdio.h>
6  #include <string.h>
7  #define MAX_LENGTH 81
8
9  void findHyphenatedWord(char sentence[MAX_LENGTH])
10 {
11      char *chPtr;  //pointer returned by strchr
12
13      chPtr = strchr(sentence, '-');
14      if ( chPtr != NULL ){
15         printf("The string contains a hyphenated word\n");
16      }//end if stmt
17      else {
18         printf("The string does not contain a hyphenated word\n");
19      }//end else stmt
20      return;
21 }//end findHyphenatedWord function
22
23 int main()
24 {
25     char aString[MAX_LENGTH]; //user entered string
26
27     printf("Please enter a string of no more than 80 characters\n\n");
28     gets(aString);
29     printf("\n");
30     findHyphenatedWord(aString);
```



K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strchr example.exe
Please enter a string of no more than 80 characters
This sentence is less than 80 characters long. So too, is this one!
The string does not contain a hyphenated word
Press any key to continue . . .


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strchr example.exe
Please enter a string of no more than 80 characters
Some elite-level cyclists train more than 500 miles per week!
The string contains a hyphenated word
Press any key to continue . . .

Using the `strstr` Function

- The `strstr` function allows you to find a specific substring of characters in a string.
- The `strstr` function is a fundamental function in the area of pattern recognition. For example, many word processors have the capability for doing global updates. You want to replace all occurrences of a certain word with another word, means that you need to find substrings inside strings.
- It requires two strings as parameters and is not destructive to either string. If the second string is found, in its entirety inside the first string, a pointer to the first character of the substring in the first string is returned, otherwise a NULL pointer is returned.
- The following program illustrates the use of the `strstr` function by determining if the string passed to it contains a certain word.

strstr example

```
4  #include <stdio.h>
5  #include <string.h>
6  #define MAX_LENGTH 81
7
8  void findSubStrings(char sentence[MAX_LENGTH], char substring[MAX_LENGTH])
9  {
10      char *chPtr;  //pointer returned by strstr
11
12      chPtr = strstr(sentence, substring);
13      if ( chPtr != NULL ){
14         printf("The string contains the substring.\n");
15      }//end if stmt
16      else {
17         printf("The string does not contain the substring.\n");
18      }//end else stmt
19      return;
20 }//end findSubStringsdWord function
21
22 int main()
23 {
24     char aString[MAX_LENGTH]; //user entered string
25     char aSubString[MAX_LENGTH]; //user entered substring
26
27     printf("Please enter a string of no more than 80 characters\n\n");
28     gets(aString);
29     printf("\n");
30     printf("Please enter a substring (<=80 chars) to find in the first string\n\n");
31     gets(aSubString);
32     printf("\n");
33     findSubStrings(aString, aSubString);
```


```
K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strstr e...
Please enter a string of no more than 80 characters

Olympic air rifle shooting requires extreme marksmanship!

Please enter a substring (<=80 chars) to find in the first string

mark

The string contains the substring.


Press any key to continue . . .
```

```
K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strstr ...
Please enter a string of no more than 80 characters

Today is Monday March 23, 2009.

Please enter a substring (<=80 chars) to find in the first string

Mark

The string does not contain the substring.


Press any key to continue . . .
```


Some More Of The Functions In <string.h>

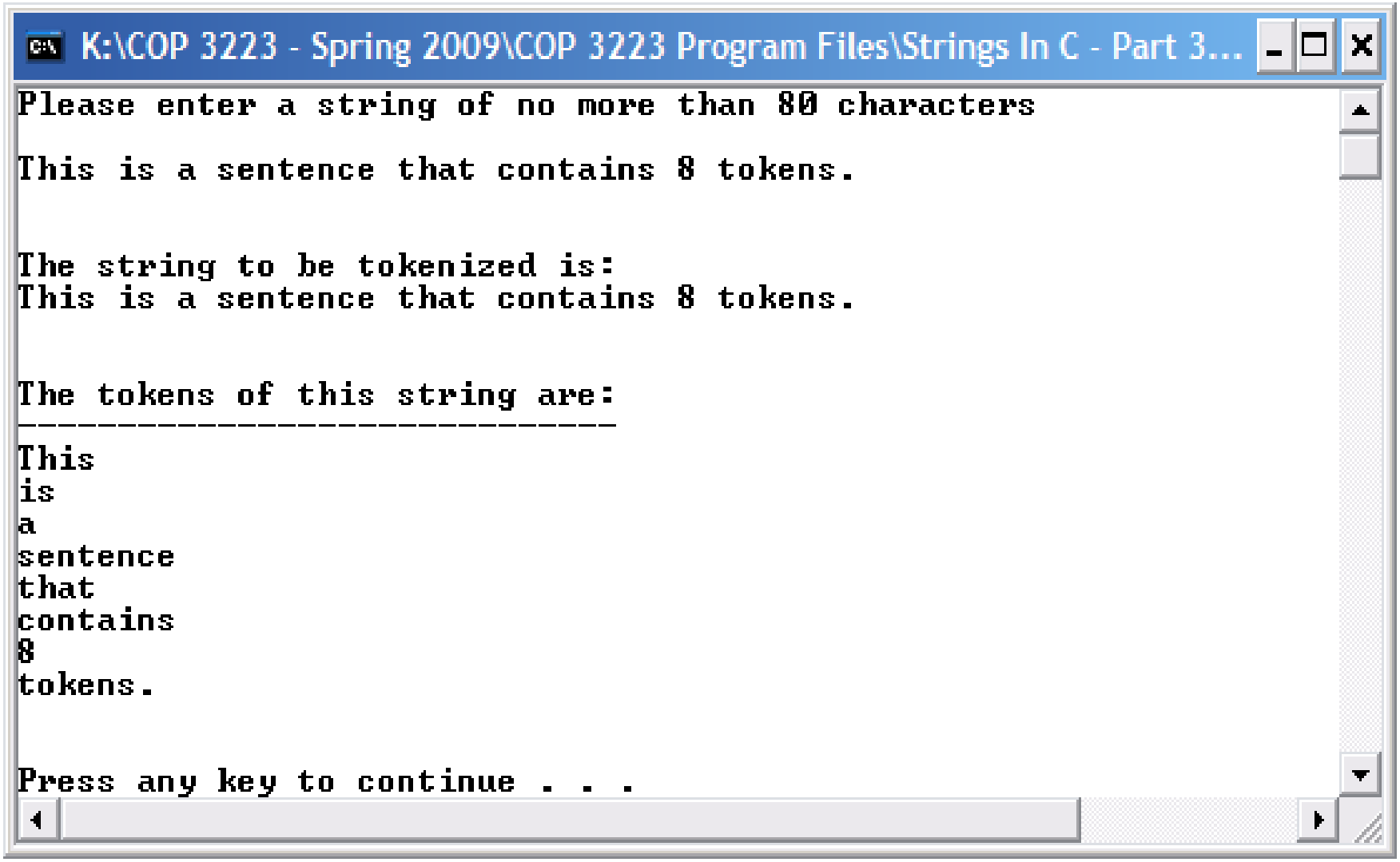
Function Prototype	Function Description
`char *strpbrk (const char *string1, const char *string2);`	Locates the first occurrence in string `s1` of any character in string `s2`. If a character from `s2` is found in `s1`, a pointer to that character in s1 is returned, otherwise, a NULL pointer is returned.
`char *strtok (char *s1, const char *s2);`	A sequence of calls to `strtok` breaks string `s1` into "tokens" - logical pieces such as words in a line of text – separated by characters contained in string `s2`. The first call contains `s1` as the first argument and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

- The examples on page 17 and 19 illustrate using these two functions.

strtok example

strtok example.c

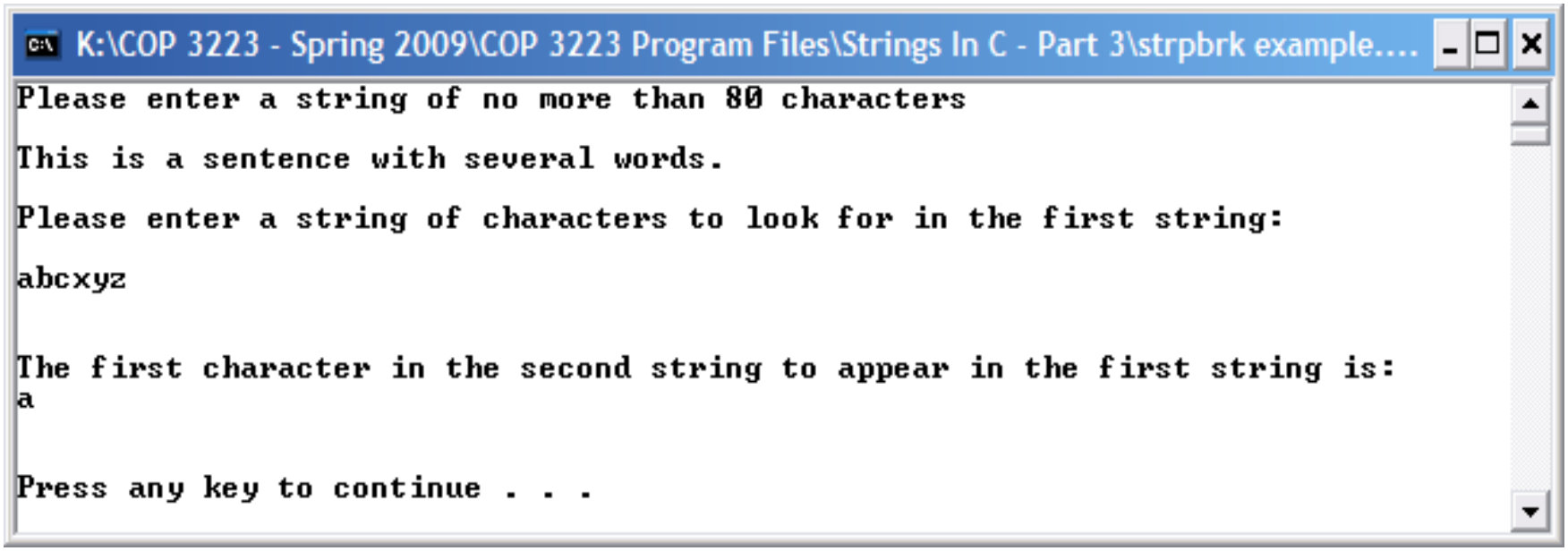
```
4  #include <stdio.h>
5  #include <string.h>
6  #define MAX_LENGTH 81
7
8  void getTokens(char sentence[MAX_LENGTH])
9  {
10      char *tokenPtr;  //pointer to the token returned by strtok
11
12      printf("The tokens of this string are:\n");
13      printf("------------------------------\n");
14      tokenPtr = strtok(sentence, " ");
15      while( tokenPtr != NULL) {
16         printf("%s\n", tokenPtr);
17         tokenPtr = strtok(NULL, " ");
18      }//end while stmt
19      return;
20 }//end getTokens function
21
22 int main()
23 {
24     char aString[MAX_LENGTH]; //user entered string
25
26     printf("Please enter a string of no more than 80 characters\n\n");
27     gets(aString);
28     printf("\n\nThe string to be tokenized is:\n");
29     puts(aString);
30     printf("\n\n");
31     getTokens(aString);
32
33     printf("\n\n");
```

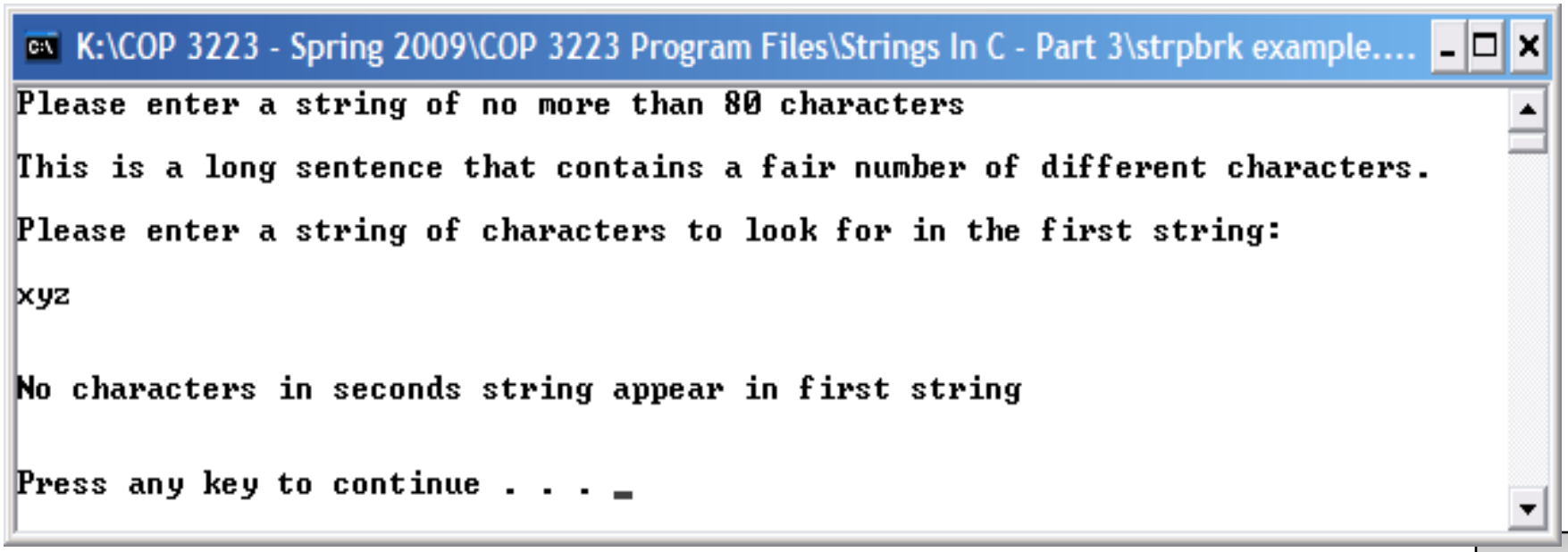


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3...
Please enter a string of no more than 80 characters
This is a sentence that contains 8 tokens.
The string to be tokenized is:
This is a sentence that contains 8 tokens.
The tokens of this string are:
This
is
a
sentence
that
contains
8
tokens.
Press any key to continue . . .

strpbrk example

strpbrk example.c

```c
#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 81

void findFirstChar(char sentence1[MAX_LENGTH], char sentence2[MAX_LENGTH])
{
     char *chPtr;  //pointer returned by strpbrk

     chPtr = strpbrk(sentence1, sentence2);
     if (chPtr != NULL) {
        printf("The first character in the second string to appear in the first string
        printf("%c\n", *chPtr);
     }//end if stmt
     else {
        printf("No characters in seconds string appear in first string\n");
     }//end else stmt
     return;
}//end findFirstChar function

int main()
{
    char aString[MAX_LENGTH]; //user entered string
    char bString[MAX_LENGTH]; //user entered search for string

    printf("Please enter a string of no more than 80 characters\n\n");
    gets(aString);
    printf("\nPlease enter a string of characters to look for in the first string:\n\n
    gets(bString);
    printf("\n\n");
```



K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strpbrk example....
Please enter a string of no more than 80 characters
This is a sentence with several words.
Please enter a string of characters to look for in the first string:
abcxyz
The first character in the second string to appear in the first string is:
a
Press any key to continue . . .


K:\COP 3223 - Spring 2009\COP 3223 Program Files\Strings In C - Part 3\strpbrk example....
Please enter a string of no more than 80 characters
This is a long sentence that contains a fair number of different characters.
Please enter a string of characters to look for in the first string:
xyz
No characters in seconds string appear in first string
Press any key to continue . . .

Practice Problems

1. Re-write the program on page 11, so that if a hyphenated word is found, the hyphen is replaced by an underscore.
2. Modify the program on page 14, so that it counts the number of times the substring appears in the string. The version shown, only finds the first occurrence, if any, in the string. Your version must find all occurrences of the substring and count them.

7 bit ASCII Table ($2^7 = 128$)

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f