CMPE 252
C PROGRAMMING

SPRING 2021

WEEK 7-8

STRINGS
CHAPTER 8

Problem Solving & Program Design in C

Eighth Edition
Global Edition

Jeri R. Hanly & Elliot B. Koffman

Chapter Objectives

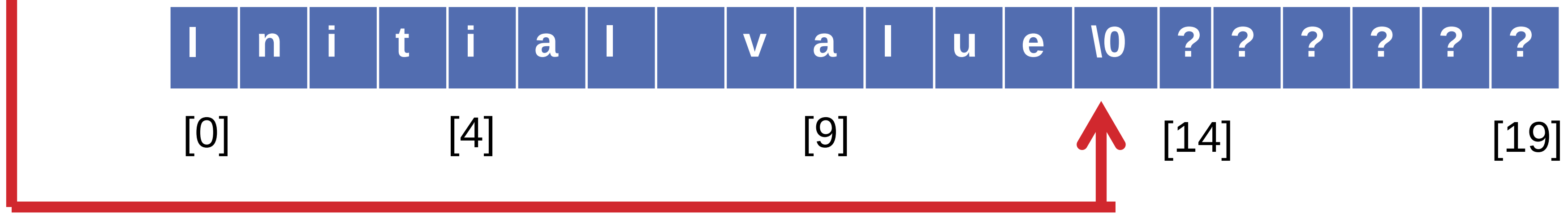
- To understand how a string constant is stored in an array of characters
- To learn about the placeholder %s and how it is used in printf and scanf operations
- To learn some of the operations that can be performed on strings such as copying strings extracting substrings, and joining strings using functions from the library string

Chapter Objectives

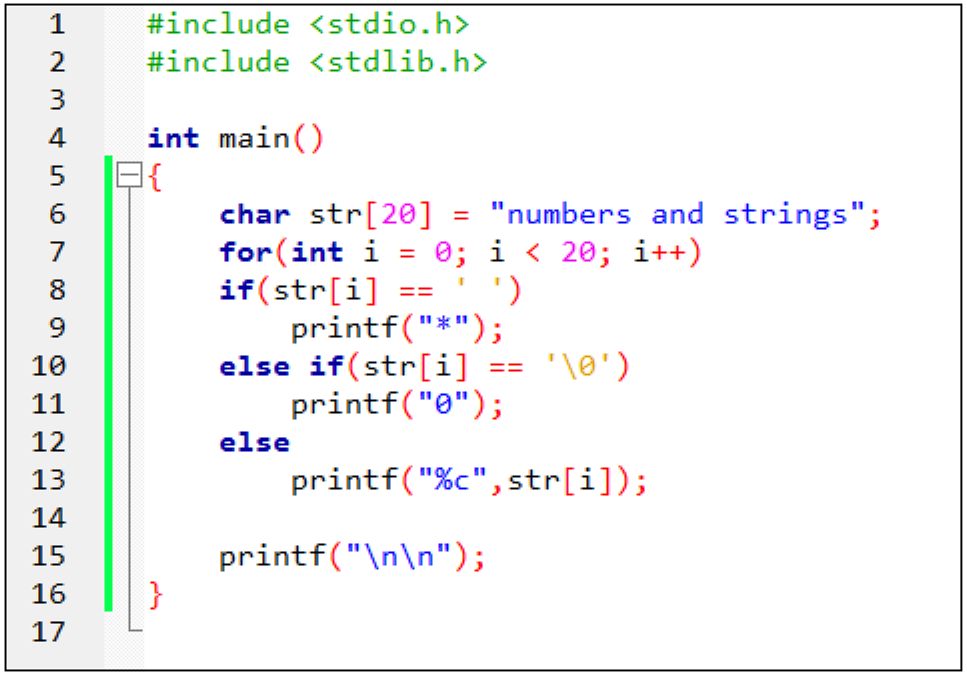
- To understand the buffer overflow dangers inherent in some string library functions
- To learn how C compares two strings to determine their relative order
- To see some of the operations that can be performed on individual characters using functions form the library `ctype`
- To learn how to write your own functions that perform some of the basic operations of a text editor program
- To understand basic principles of defensive programming

String Basics

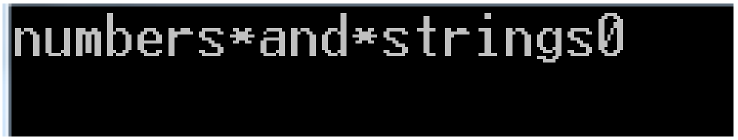
- A blank in a string is a valid character.
- null character
  - character '\0' that marks the end of a string in C
- A string constant can be associated with a symbolic name using #define directive
  - #define ERR_PREFIX   " *******Error- "
- A string in C is implemented as an array.
  - char string_var[30];
  - char str[20] = "Initial value";



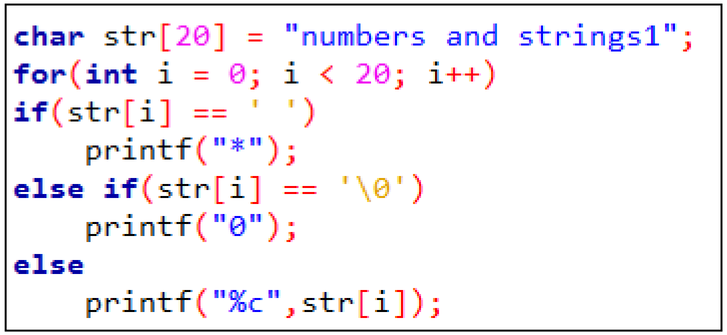
String Basics



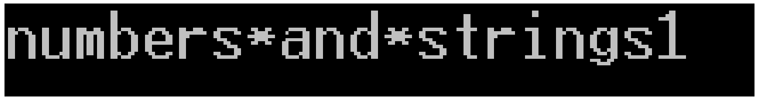
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char str[20] = "numbers and strings";
    for(int i = 0; i < 20; i++)
    if(str[i] == ' ')
        printf("*");
    else if(str[i] == '\0')
        printf("0");
    else
        printf("%c",str[i]);

    printf("\n\n");
}
```



```
numbers*and*strings0
```



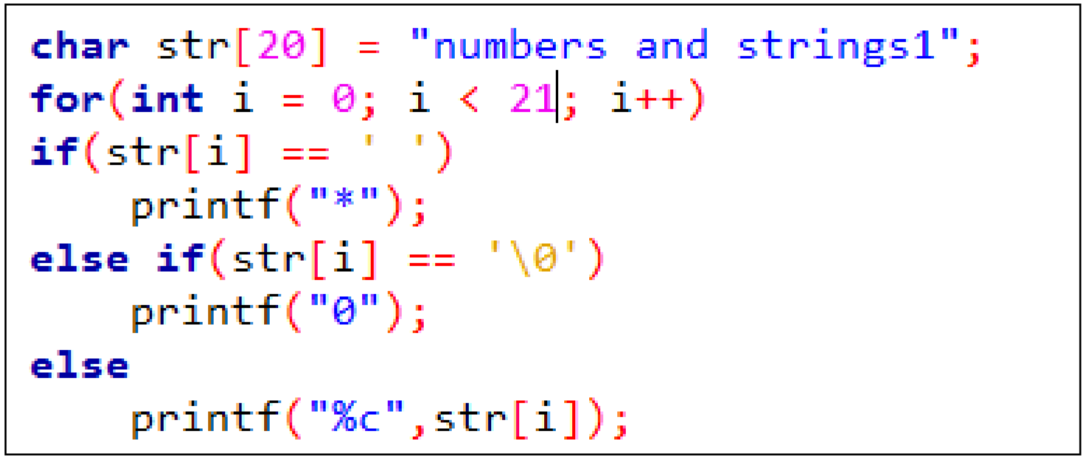
```
char str[20] = "numbers and strings1";
for(int i = 0; i < 20; i++)
if(str[i] == ' ')
    printf("*");
else if(str[i] == '\0')
    printf("0");
else
    printf("%c",str[i]);
```



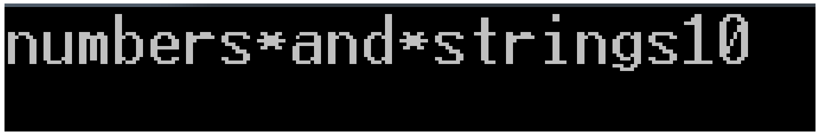
```
numbers*and*strings1
```

Where is \0 then?

String Basics

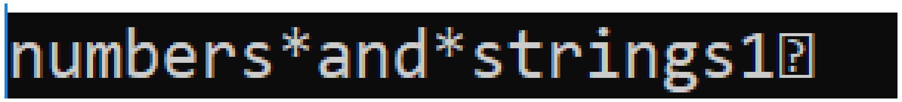


```c
char str[20] = "numbers and strings1";
for(int i = 0; i < 21; i++)
if(str[i] == ' ')
    printf("*");
else if(str[i] == '\0')
    printf("0");
else
    printf("%c",str[i]);
```



```
numbers*and*strings10
```

Output in one computer



```
numbers*and*strings1�
```

Output in another computer

String Basics

- An array of strings is a 2-dimensional array of characters in which each row is a string.
- **Quick Check:** declare an array of strings which keeps names (max. 25 char) of 30 people
  - char names [30][25]
  - Remember that in multidim. arrays, grouping is done row by row
  - We need 30 rows for people

Array of String Initialization at Declaration

- char month [12] [10] = { "January", "February", "March", "April", " May", " June", " July", " August",

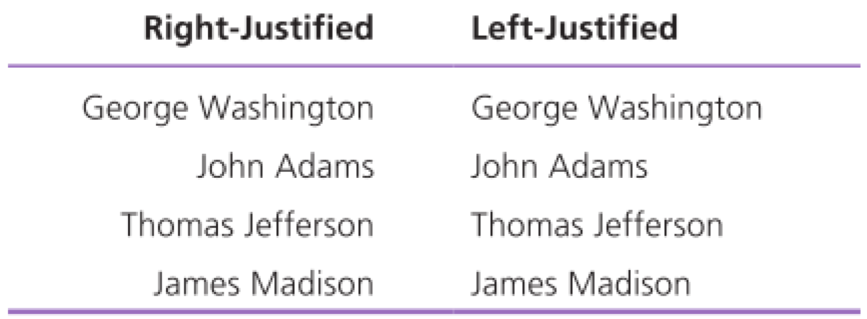
" September", " October", " November", " December"   }

Input/Output

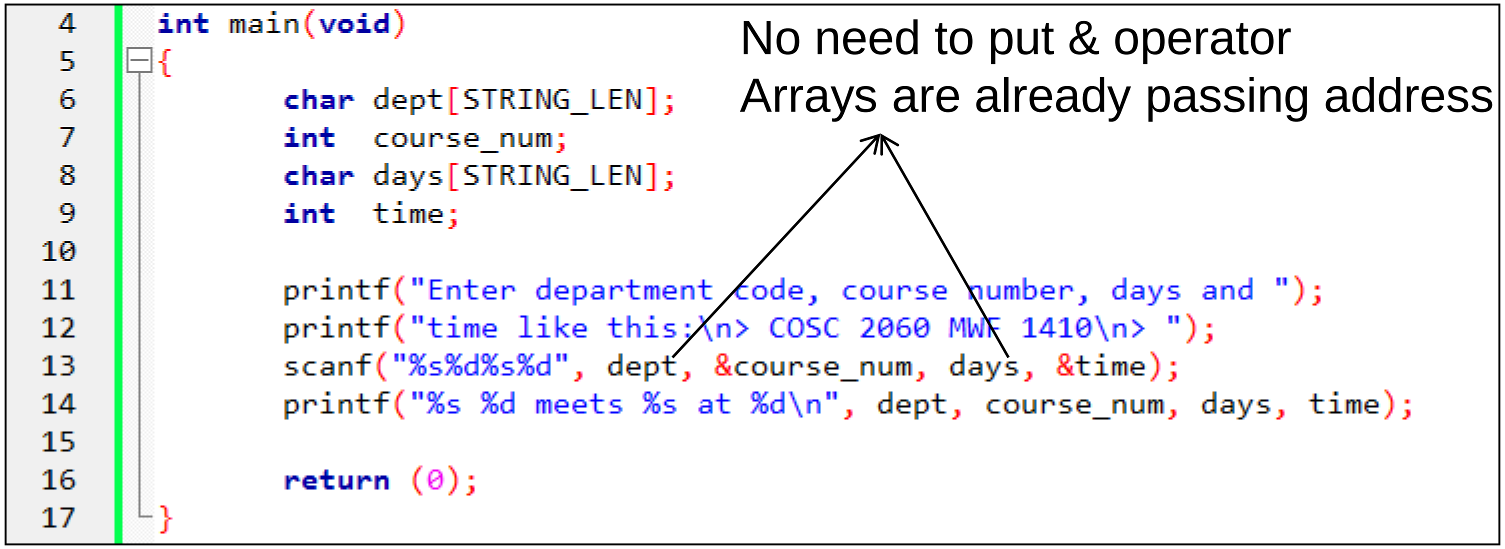
- printf and scanf can handle string arguments
- use %s as the placeholder in the format string
- use a – (minus) sign to force left justification
  - printf("%-20s\n", president);

FIGURE 8.1

Right and Left Justification of Strings

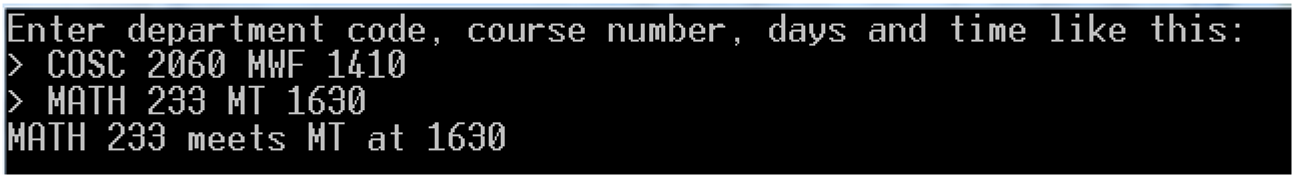


Right-Justified	Left-Justified
George Washington	George Washington
John Adams	John Adams
Thomas Jefferson	Thomas Jefferson
James Madison	James Madison

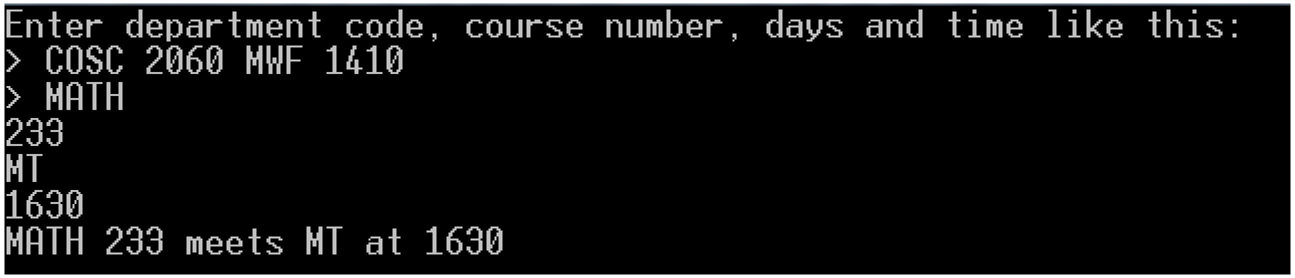


```
int main(void)
{
        char dept[STRING_LEN];
        int  course_num;
        char days[STRING_LEN];
        int  time;

        printf("Enter department code, course number, days and ");
        printf("time like this:\n> COSC 2060 MWF 1410\n> ");
        scanf("%s%d%s%d", dept, &course_num, days, &time);
        printf("%s %d meets %s at %d\n", dept, course_num, days, time);

        return (0);
}
```

No need to put & operator
Arrays are already passing address

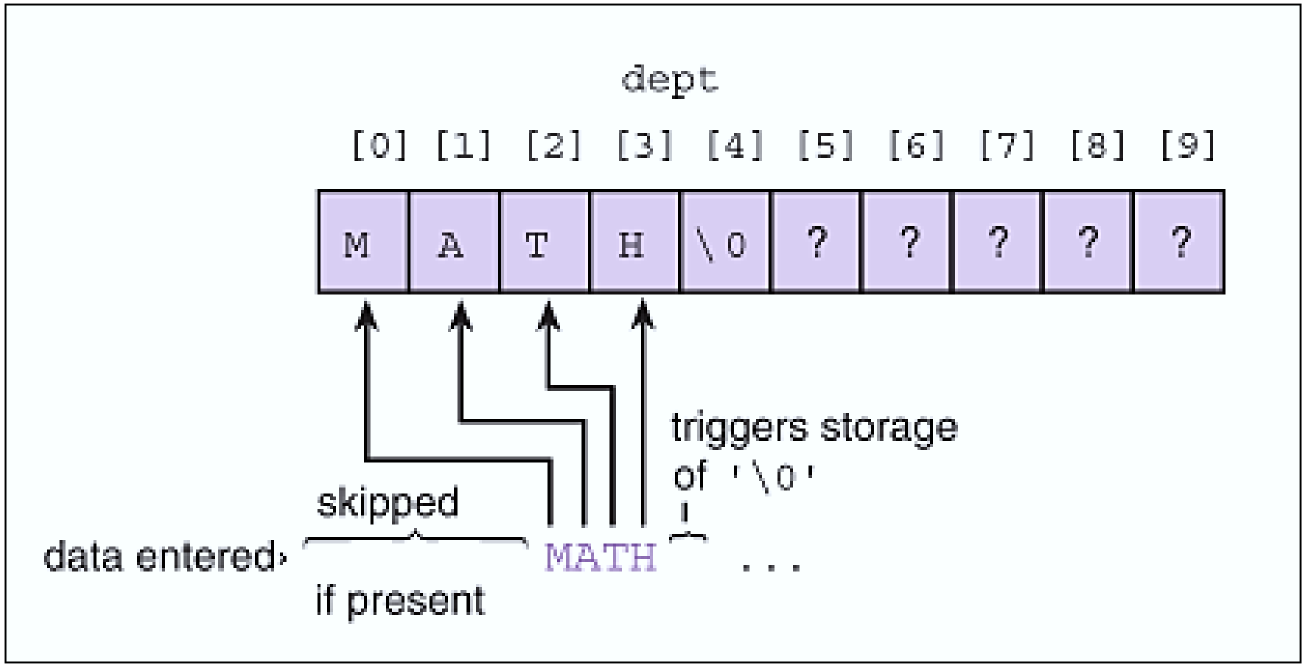


```
Enter department code, course number, days and time like this:
> COSC 2060 MWF 1410
> MATH 233 MT 1630
MATH 233 meets MT at 1630
```



```
Enter department code, course number, days and time like this:
> COSC 2060 MWF 1410
> MATH
233
MT
1630
MATH 233 meets MT at 1630
```

values can be spaced in many ways, treating whitespace is important


dept
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
M A T H \0 ? ? ? ? ?
triggers storage of '\0'
skipped if present
data entered
MATH
. . .

Function `scanf` would have difficulty if some essential whitespace between values were omitted or if a nonwhitespace separator were substituted. For example, if the data were entered as

```
> MATH1270 TR 1800
```

`scanf` would store the eight-character string `"MATH1270"` in `dept` and would then be unable to convert `T` to an integer for storage using the next parameter. The situation would be worse if the data were entered as

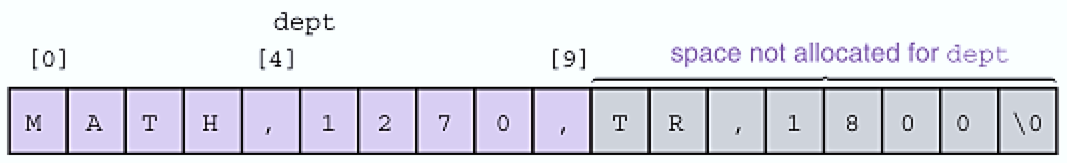
```
> MATH,1270,TR,1800
```

Then the `scanf` function would store the entire 17-character string plus `'\0'` in the `dept` array, causing characters to be stored in eight locations not allocated to `dept`, as shown in Fig. 8.4.

Buffer Overflow

- more data is stored in an array than its declared size allows
- a very dangerous condition
- unlikely to be flagged as an error by either the compiler or the run-time system

FIGURE 8.4 Execution of `scanf("%s%d%s%d", dept, &course_num, days, &time);` on Entry of Invalid Data


dept
[0]
[4]
[9]
space not allocated for dept
M A T H , 1 2 7 0 , T R , 1 8 0 0 \0

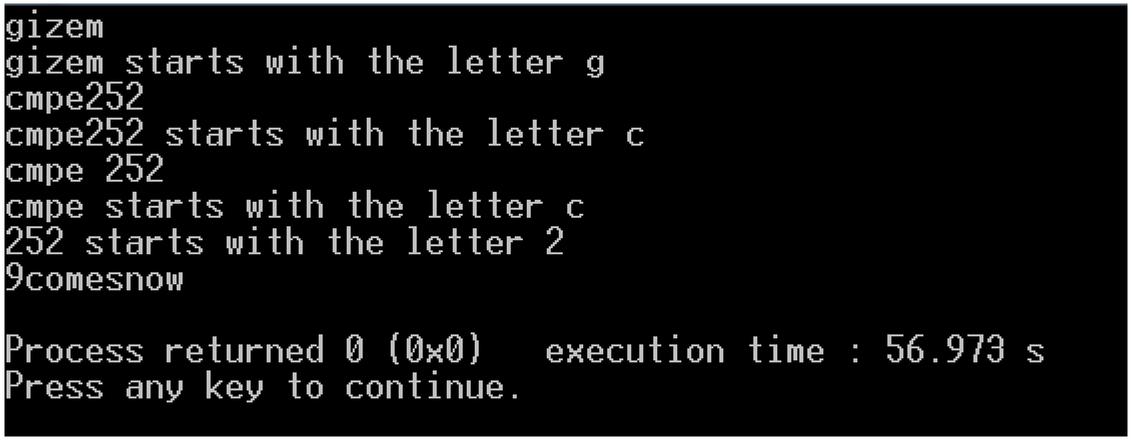
Quick Check

Write a program that takes a word less than 25 characters and prints a statement like this:

fractal starts with letter f

Have the program process words until it encounters a word beginning with the character ‘9’

```c
char in[25];

for (scanf("%s", in); in[0] != '9'; scanf("%s", in))
    printf("%s starts with the letter %c\n", in, in[0]);
```



```
gizem
gizem starts with the letter g
cmpe252
cmpe252 starts with the letter c
cmpe 252
cmpe starts with the letter c
252 starts with the letter 2
9comesnow

Process returned 0 (0x0)   execution time : 56.973 s
Press any key to continue.
```

= operator

- char one_str[20] = "Test string"; ✓

- char one_str[20] ;
- one_str = "Test string"; ✗

Array name with no subscript is an address, a pointer to initial array element. This address is constant which cannot be changed through assignment.

String Terminology

- string length
  - in a character array, the number of characters before the first null character
- empty string
  - a string of length zero
  - the first character of the string is the null character

string.h library

Function	Purpose	Parameters	Result Type
strlen	*Returns the number of characters without null character at the end* *strlen*("hello") returns 5	const char* s1	size_t

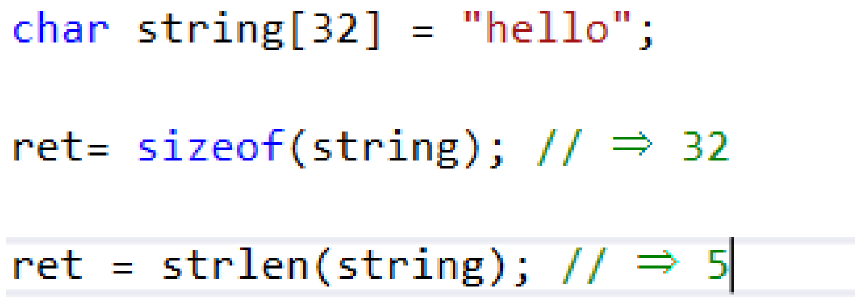
(In other words, it returns the offset of the terminating null byte within the array.)

```
strcpy(dest, "hello");
printf("%d",strlen(dest));
```

```
5
```

strlen

- !When applied to an array, the strlen function returns of the string stored there, not its allocated size. the length
- You can get the allocated size of the array that holds a string using the sizeof operator:



```
char string[32] = "hello";

ret= sizeof(string); // ⇒ 32

ret = strlen(string); // ⇒ 5
```

```
char *sptr = string;

ret = strlen(sptr); // ⇒ 5

ret = sizeof(sptr); // ⇒ 4
```