Strings in C
By
Dr. A. B. Kadam

Strings

- C implements the **string** data structure using arrays of type `char`.
- You have already used the string extensively.
  - printf(“This program is terminated!\n”);
  - #define ERR_Message “Error!!”
- Since **string** is an array, the declaration of a string is the same as declaring a `char` array.
  - char string_var[30];
  - char string_var[20] = “Initial value”;

Memory Storage for a String

- The string is always ended with a **null character ‘\0’**.
- The characters after the null character are ignored.
- e.g., char str[20] = “Initial value”;

[0]													[13]			
I	n	i	t	i	a	l		v	a	l	u	e	**\0**	?	?	…

Arrays of Strings

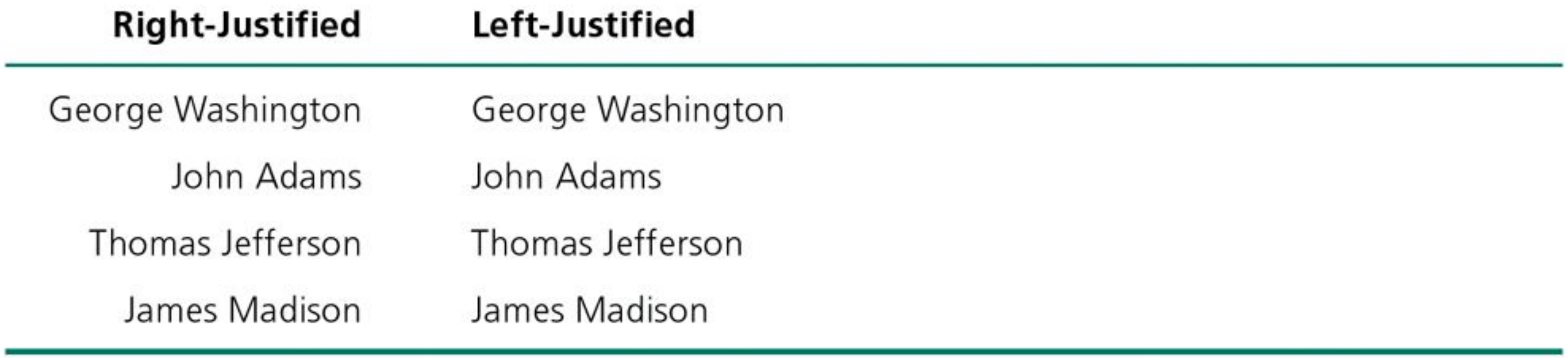
- An array of strings is a two-dimensional array of characters in which each row is one string.
  - `char names[People][Length];`
  - `char month[5][10] = {"January", "February", "March", "April", "May"};`

Input/Output of a String

- The placeholder **%s** is used to represent string arguments in printf and scanf.
  - printf(“Topic: %s\n”, string_var);
- The string can be right-justified by placing a positive number in the placeholder.
  - printf(“**%8s**”, str);
- The string can be left-justified by placing a negative number in the placeholder.
  - Printf(“**%-8s**”, str);

Right and Left Justification of Strings

The "%8s" placeholder displays a string which is right-justified and in 8-columns width.
If the actual string is longer than the width, the displayed field is expanded with no padding.



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

An Example of Manipulating String with scanf and printf

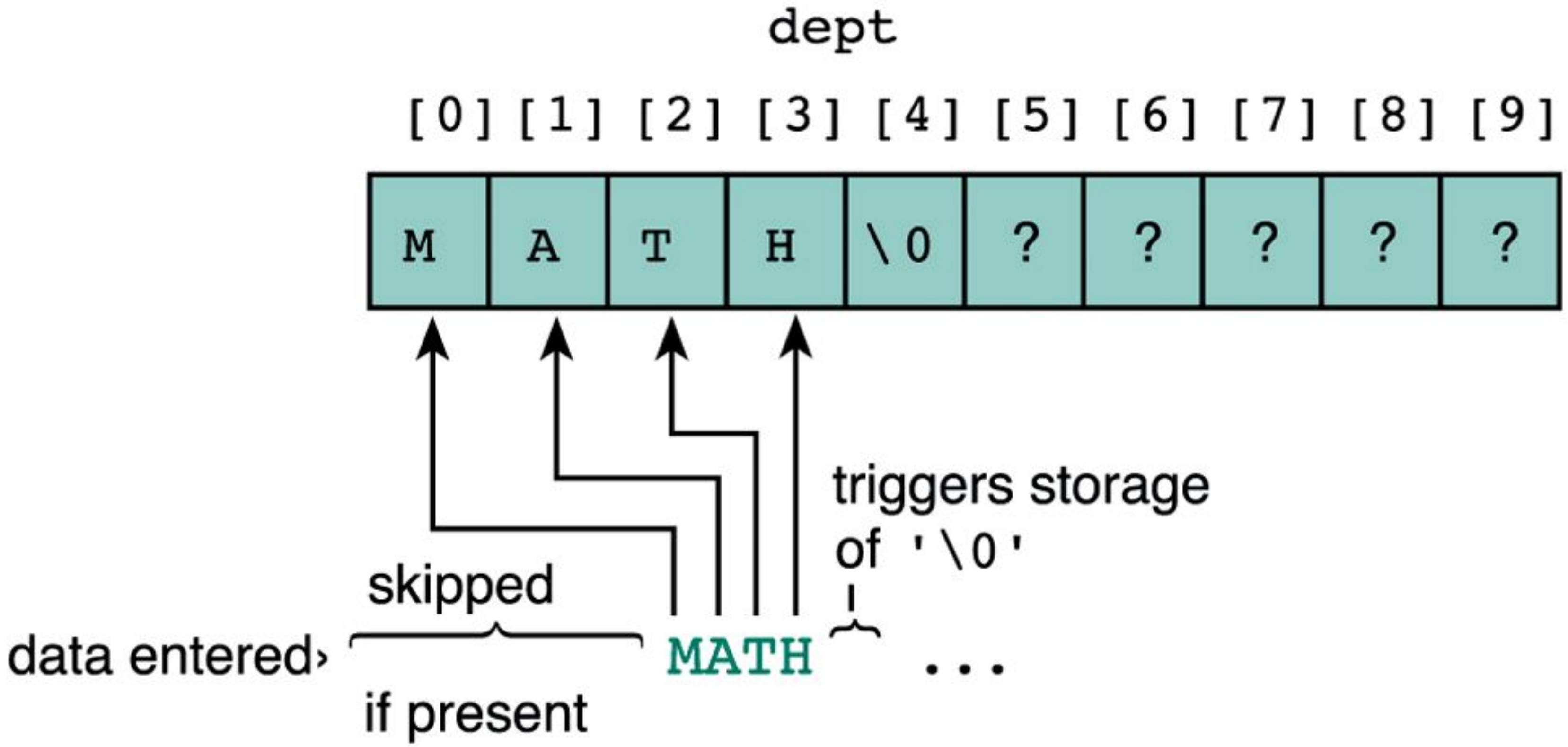
```c
#include <stdio.h>

#define STRING_LEN  10

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);
}
```

```
Enter department code, course number, days and time like this:
> COSC 2060 MWF 1410
> MATH 1270 TR 800
MATH 1270 meets TR at 800
```

The `dept` is the initial memory address of the string argument. Thus we don't apply the `&` operator on it.

Execution of scanf ("%s", dept);

- Whenever encountering a white space, the scanning stops and `scanf` places the null character at the end of the string.
- e.g., if the user types “MATH 1234 TR 1800,” the string “MATH” along with ‘0’ is stored into `dept`.



String Library Functions

- The string can not be copied by the assignment operator '='.
  - e..g, "`str = "Test String"`" is not valid.
- C provides string manipulating functions in the "string.h" library.
  - The complete list of these functions can be found in Appendix B of the textbook.

Some String Functions from String.h

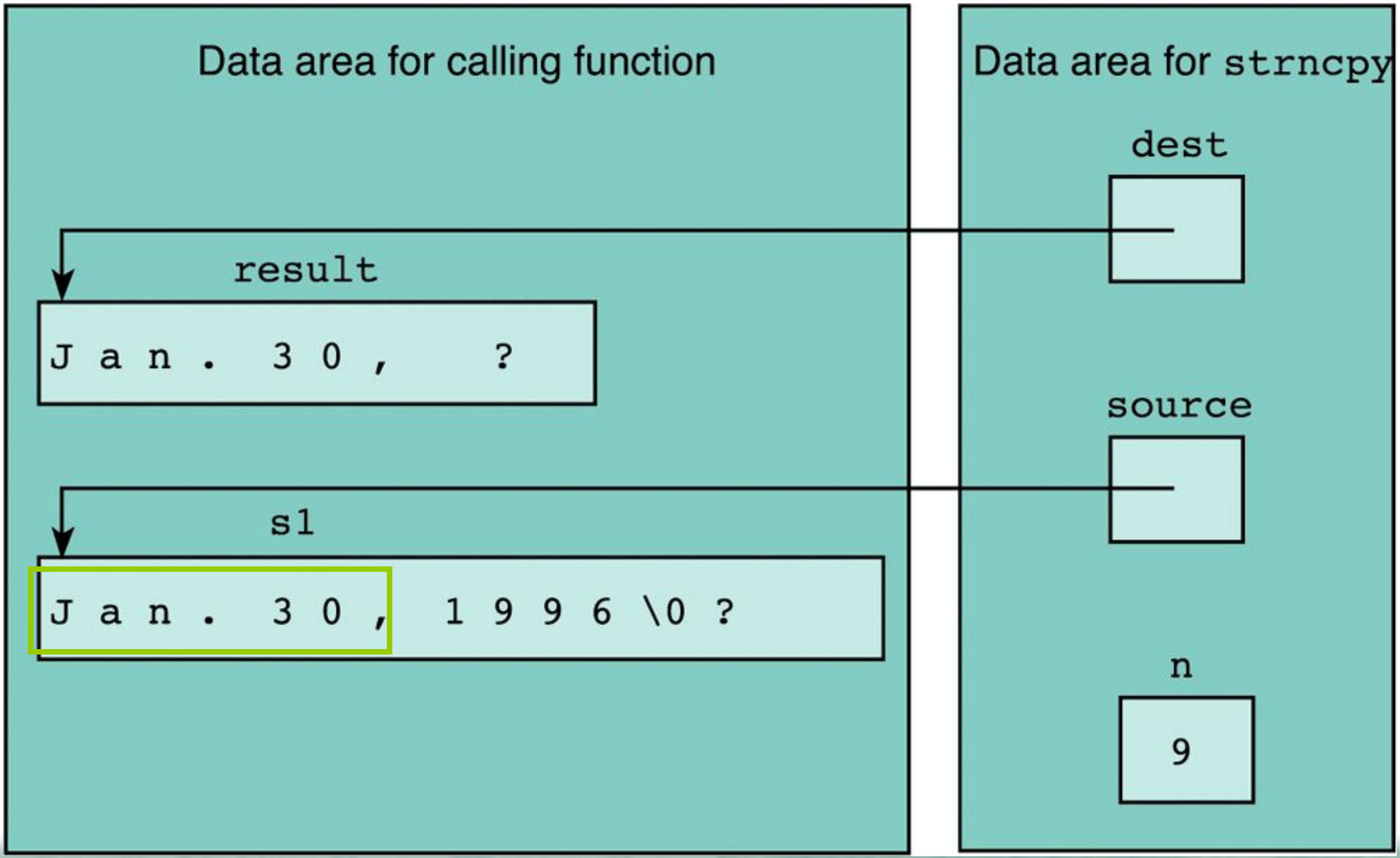
Function	Purpose	Example
`strcpy`	Makes a copy of a string	strcpy(s1, “Hi”);
`strcat`	Appends a string to the end of another string	strcat(s1, “more”);
`strcmp`	Compare two strings alphabetically	strcmp(s1, “Hu”);
`strlen`	Returns the number of characters in a string	strlen(“Hi”) returns 2.
`strtok`	Breaks a string into tokens by delimiters.	strtok(“Hi, Chao”, “ ,”);

Functions `strcpy` and `strncpy`

- Function `strcpy` copies the string in the second argument into the first argument.
  - e.g., strcpy(dest, “test string”);
  - The **null character** is appended at the end automatically.
  - If source string is longer than the destination string, the overflow characters may occupy the memory space used by other variables.
- Function `strncpy` copies the string by specifying the number of characters to copy.
  - You have to place the null character manually.
  - e.g., strncpy(dest, “test string”, 6); **dest[6] = ‘\0’;**
  - If source string is longer than the destination string, the overflow characters are discarded automatically.

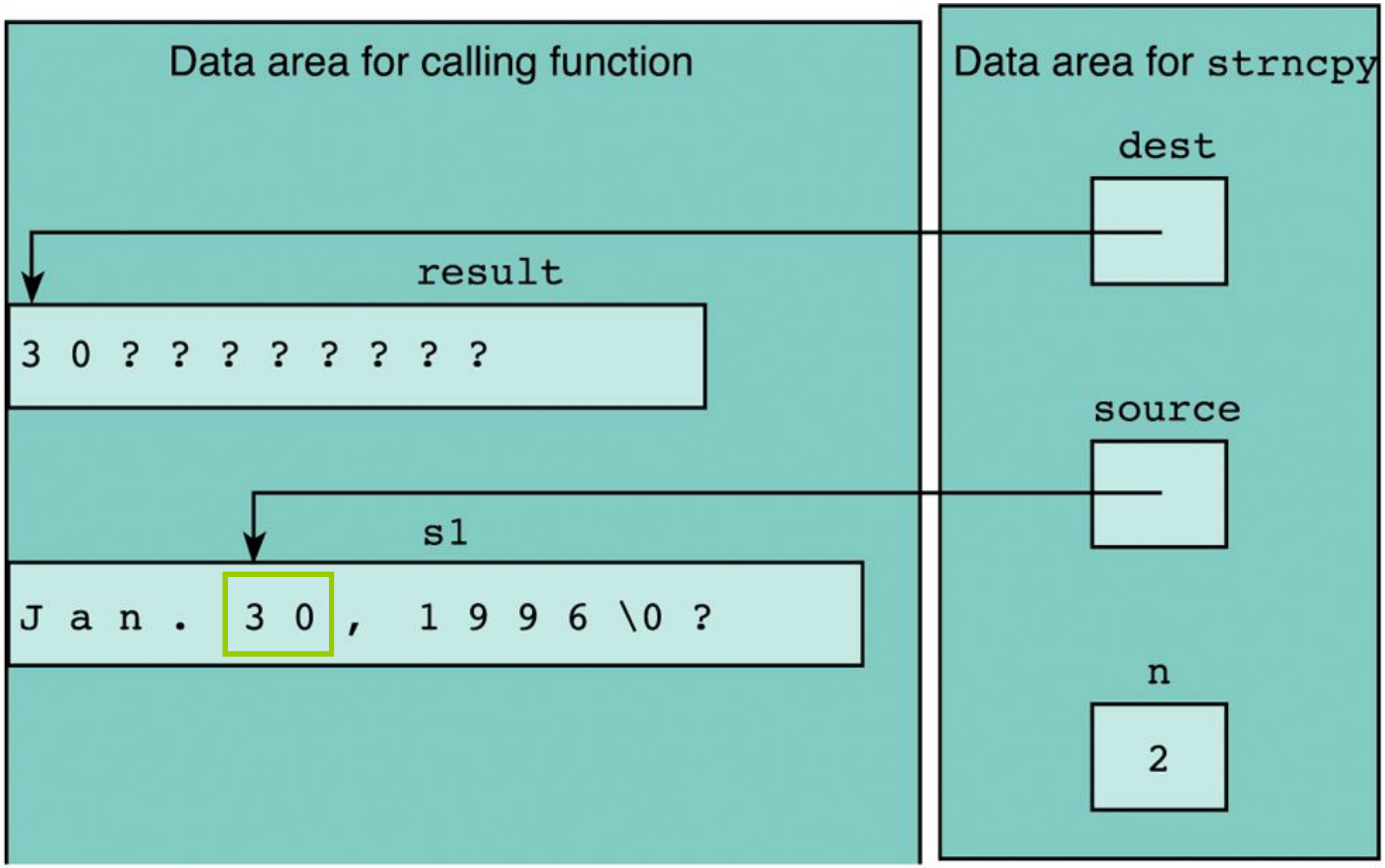
Extracting Substring of a String (1/2)

- We can use `strncpy` to extract substring of one string.
  - e.g., `strncpy(result, s1, 9);`



Extracting Substring of a String (2/2)

- e.g., `strncpy(result, &s1[5], 2);`

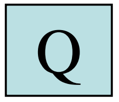


Functions `strcat` and `strlen`

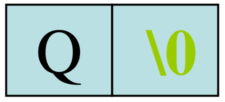
- Functions `strcat` and `strncat` concatenate the fist string argument with the second string argument.
  - `strcat(dest, “more..”);`
  - `strncat(dest, “more..”, 3);`
- Function `strlen` is often used to check the length of a string (i.e., the number of characters before the fist null character).
  - e.g., dest[6] = “Hello”;
    strncat(dest, “more”, 5-strlen(dest));
    dest[5] = ‘\0’;

Distinction Between Characters and Strings

- The representation of a char (e.g., ‘Q’) and a string (e.g., “Q”) is essentially different.
  - A string is an array of characters ended with the null character.



Character ‘Q’



String “Q”

String Comparison (1/2)

- Suppose there are two strings, `str1` and `str2`.
  - The condition `str1 < str2` compare the **initial memory address** of `str1` and of `str2`.
- The comparison between two strings is done by comparing each corresponding character in them.
  - The characters are comapared against the ASCII table.
  - "thrill" < "throw" since 'i' < 'o';
  - "joy" < joyous";
- The standard string comparison uses the `strcmp` and `strncmp` functions.

String Comparison (2/2)

Relationship	Returned Value	Example
`str1 < str2`	Negative	“Hello”< “Hi”
`str1 = str2`	0	“Hi” = “Hi”
`str1 > str2`	Positive	“Hi” > “Hello”

- e.g., we can check if two strings are the same by

```
if(strcmp(str1, str2) != 0)
  printf(“The two strings are different!”);
```

Input/Output of Characters and Strings

- The `stdio` library provides `getchar` function which gets the next character from the standard input.
  - "`ch = getchar();`" is the same as "`scanf("%c", &ch);`"
  - Similar functions are `putchar`, `gets`, `puts`.
- For IO from/to the file, the `stdio` library also provides corresponding functions.
  - `getc`: reads a character from a file.
  - Similar functions are `putc`, `fgets`, `fputs`.

Character Analysis and Conversion

- The **<ctype.h>** library defines facilities for character analysis and conversion.

Functions	**Description**
`isalpha`	Check if the argument is a letter
`isdigit`	Check if the argument is one of the ten digits
`isspace`	Check if argument is a space, newline or tab.
`tolower`	Converts the lowercase letters in the argument to upper case letters.

Conversions Between Strings Numbers

- The <stdlib.h> defines some basic functions for conversion from strings to numbers:
  - **atoi("123")** converts a string to an integer.
  - **atol("123")** converts a string to a long integer.
  - **atof("12.3")** converts a string to a float.
- However, there is no functions such as itoa, itof, …etc,
  - because there is a function called **sprintf** which can converts many formats to a string.

The `sprintf` and `sscanf` Functions

- The **sprintf** function substitutes values for placeholders just as **printf** does except that it stores the result into a character array
  - `sprintf(s, “%d%d%d”, mon, day, year);`
- The **sscanf** function works exactly like **scanf** except that it takes data from the string as its input argument.
  - `sscanf(“ 11 22.2 Hello”, “%d%lf%s”, &num, &val, word);`