

Characters and String Functions in C

By: Atiya Jokhio

Character-Handling library

- The character-handling library (<ctype.h>) includes several functions that perform useful tests and manipulations of character data.
- There are many functions in this library.

- Functions **islower**, **isupper**, **tolower** and **toupper**
 - **islower** determines whether its argument is a lowercase letter (a–z).
 - Function **isupper** determines whether its argument is an uppercase letter (A–Z).
 - Function **tolower** converts an uppercase letter to a lowercase letter and returns the lowercase letter. If the argument is not an uppercase letter, **tolower** returns the argument unchanged.
 - Function **toupper** converts a lowercase letter to an uppercase letter and returns the uppercase letter. If the argument is not a lowercase letter, **toupper** returns the argument unchanged.

Functions (islower, isupper, tolower and toupper)

```
// Using functions islower, isupper, tolower, toupper
#include <stdio.h>
#include <ctype.h>

int main( void )
{
    printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
        "According to islower:",
        islower( 'p' ) ? "p is a " : "p is not a ",
        "lowercase letter",
        islower( 'P' ) ? "P is a " : "P is not a ",
        "lowercase letter",
        islower( '5' ) ? "5 is a " : "5 is not a ",
        "lowercase letter",
        islower( '!' ) ? "! is a " : "! is not a ",
        "lowercase letter" );
```

```
printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
    "According to isupper:",
    isupper( 'D' ) ? "D is an " : "D is not an ",
    "uppercase letter",
    isupper( 'd' ) ? "d is an " : "d is not an ",
    "uppercase letter",
    isupper( '8' ) ? "8 is an " : "8 is not an ",
    "uppercase letter",
    isupper( '$' ) ? "$ is an " : "$ is not an ",
    "uppercase letter" );
```

Functions (islower, isupper, tolower and toupper)

```
printf( "%s%c\n%s%c\n%s%c\n%s%c\n",  
        "u converted to uppercase is ", toupper( 'u' ),  
        "7 converted to uppercase is ", toupper( '7' ),  
        "$ converted to uppercase is ", toupper( '$' ),  
        "L converted to lowercase is ", tolower( 'L' ) );  
} // end main
```

According to islower:

p is a lowercase letter
P is not a lowercase letter
5 is not a lowercase letter
! is not a lowercase letter

According to isupper:

D is an uppercase letter
d is not an uppercase letter
8 is not an uppercase letter
\$ is not an uppercase letter

u converted to uppercase is U
7 converted to uppercase is 7
\$ converted to uppercase is \$
L converted to lowercase is l

Strings

- Strings are the form of data used in programming languages for storing and manipulating text such as words, names, and sentences.
- In C, a string is not a formal data type as it is in some languages (e.g., Pascal, BASIC, VB, etc).
- String is an array of type **char**.
- All strings must end with a null character, '\0', which has a numerical value of 0.

String Functions

The string functions provide the ability to manipulate string variables and literals. String functions behave similarly to the standard C language functions. The return values also follow the C language convention. Most of the commons are giving in next slides.

String-Manipulation Functions of the String-Handling Library

- The string-handling library (`<string.h>`) provides many useful functions for manipulating string data (copying strings and concatenating strings), comparing strings, searching strings for characters and other strings, and determining the length of strings.

String-Manipulation Functions of the String-Handling Library

- Function `strcat()`: concatenates (joins) two strings.

`char *strcat(char *dest, const char *src)`

- appends its second argument (a string) to its first argument (a character array containing a string). The first character of the second argument replaces the null (`'\0'`) that terminates the string in the first argument.
- we must ensure that the array used to store the first string is large enough to store the first string, the second string and the terminating null character copied from the second string.

String-Manipulation Functions of the String-Handling Library

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[10] = "Hello";
    char s2[10] = "World";
    strcat(s1,s2);
    printf("Output string after concatenation: %s", s1);
    return 0;
}
```

Output:

```
Output string after concatenation: HelloWorld
```

String-Manipulation Functions of the String-Handling Library

- Function `strcmp()`: Compares the string `s1` with the string `s2`.

`int strcmp(const char*s1, const char*s2);`

- Function `strcmp` compares its first string argument with its second string argument, character by character.
- The function returns 0 if the strings are equal, a negative value if the first string is less than the second string and a positive value if the first string is greater than the second string.

String-Manipulation Functions of the String-Handling Library

- Function strcmp():

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[20] = "BeginnersBook";
    char s2[20] = "BeginnersBook.COM";

    if (strcmp(s1, s2) == 0)
    {
        printf("string 1 and string 2 are equal");
    }
    else if (strcmp(s2, s1) > 0)
    {
        printf("string 1 is greater than string 2");
    }
    else
    {
        printf("string 1 is less than string 2");
    }
    return 0;
}
```



The screenshot shows a Windows command prompt window with a red title bar. The title bar text is "D:\FAST UNIVERSITY\Spring 2020\Programming Fundamental\Le...". The window contains the following text: "string 1 is greater than string 2", followed by a horizontal line, "Process exited with return value 0", and "Press any key to continue . . .".

String-Manipulation Functions of the String-Handling Library

- Function `strcpy()`: Copies string `s2` into array `s1`. The value of `s1` is returned.

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

- Function `strcpy` copies its second argument (a string) into its first argument—a character array that we must ensure is large enough to store the string and its terminating null character, which is also copied.

String-Manipulation Functions of the String-Handling Library

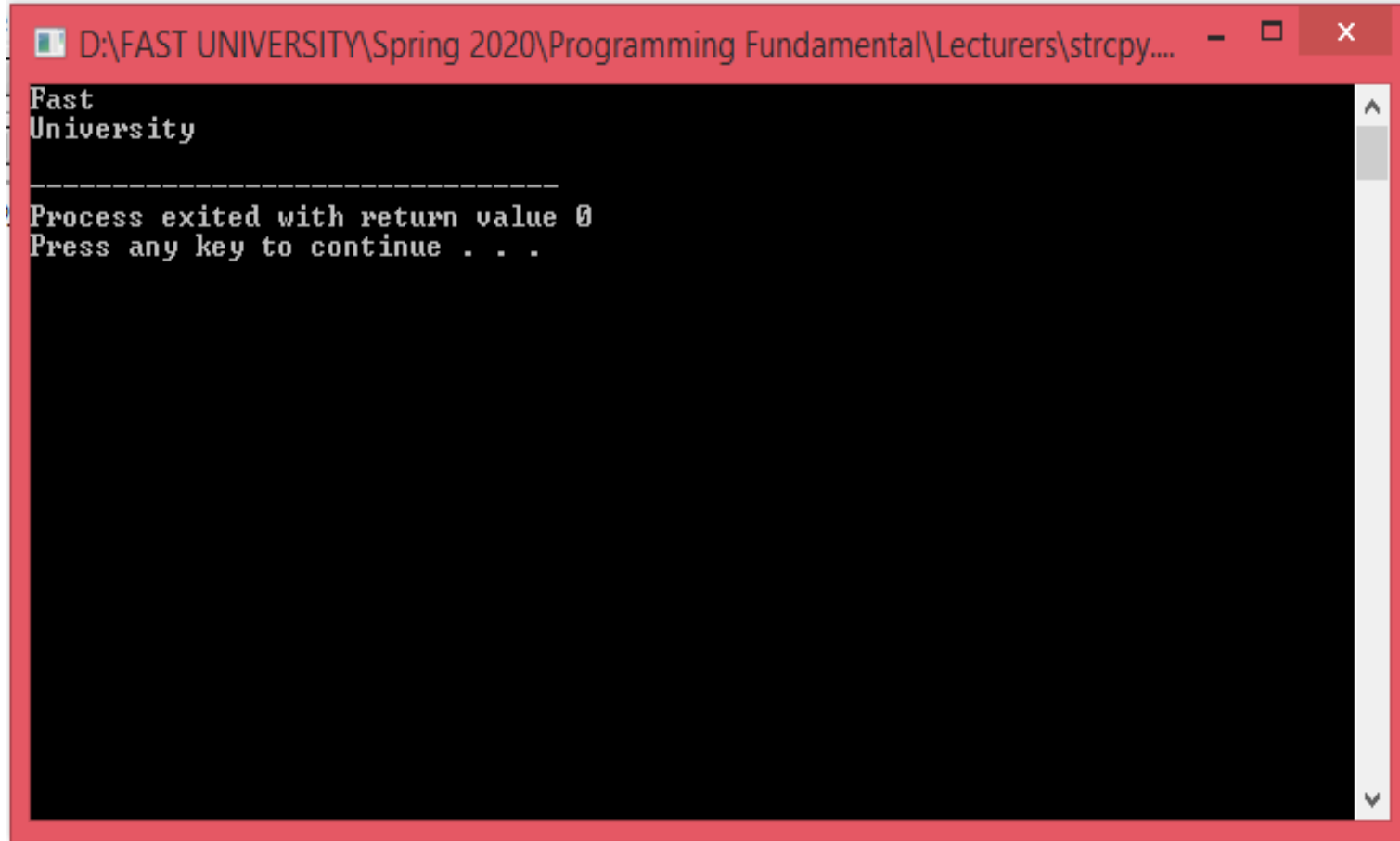
```
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[10]= "Fast";
    char str2[15];
    char str3[15];

    strcpy(str2, str1);
    strcpy(str3, "University");
    puts(str2);
    puts(str3);

    return 0;
}
```

String-Manipulation Functions of the String-Handling Library



```
D:\FAST UNIVERSITY\Spring 2020\Programming Fundamental\Lecturers\strcpy....  
Fast  
University  
-----  
Process exited with return value 0  
Press any key to continue . . .
```

String-Manipulation Functions of the String-Handling Library

- Function `strlen()`: calculates the length of a given string.

`int strlen(const char *str);`

- Computes the length of the string *str* up to but not including the terminating null character.
- Returns the number of characters in the string.

String-Manipulation Functions of the String-Handling Library

- Function strlen():

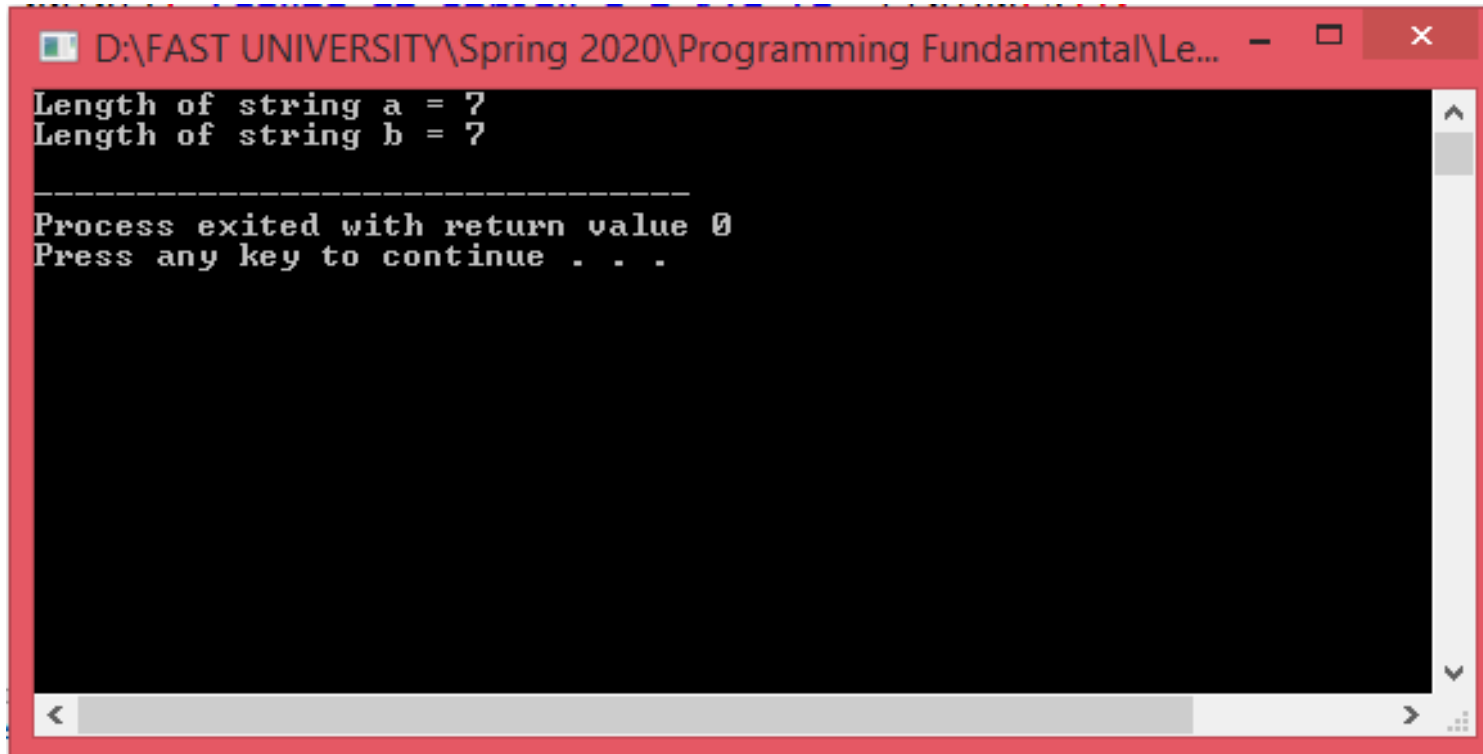
```
#include <stdio.h>
#include <string.h>
int main()
{
    char a[20]="Program";
    char b[20]={'P','r','o','g','r','a','m','\0'};

    printf("Length of string a = %ld \n",strlen(a));
    printf("Length of string b = %ld \n",strlen(b));

    return 0;
}
```

String-Manipulation Functions of the String-Handling Library

- Function `strlen()`:



```
D:\FAST UNIVERSITY\Spring 2020\Programming Fundamental\Le...  
Length of string a = ?  
Length of string b = ?  
-----  
Process exited with return value 0  
Press any key to continue . . .
```