

Lecture 9

Strings



Fundamentals of Characters and Strings

- ▶ Characters in C consist of any printable or nonprintable character in the computer's character set including lowercase letters, uppercase letters, decimal digits, special characters and escape sequences.
- ▶ A character is usually stored in the computer as an 8-bits (1 byte) integer.
- ▶ The integer value stored for a character depends on the character set used by the computer on which the program is running.

Fundamentals of Characters and Strings

- ▶ There are two commonly used character sets:
 - ASCII (American Standard Code for Information Interchange)
 - EBCDIC (Extended Binary Coded Decimal Interchange Code)

Difference Between an Integer Digit and a Character Digit

- ▶ `char num = 1` and `char num = '1'` are not the same.
- ▶ `char num = 1` is represented in the computer as `00000001`.
- ▶ `char num = '1'` on the other hand is number 49 according to the ASCII character set. Therefore, it is represented in the computer as `00110001`.

Example: ASCII character

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

```
void main(void)
```

```
{
```

```
    char my_A = 'A';
```

```
    char my_Z = 'Z';
```

```
    char my_a = 'a';
```

```
    char my_z = 'z';
```

```
    printf("\nASCII value for A is %d", my_A);
```

```
    printf("\nASCII value for Z is %d",my_Z);
```

```
    printf("\nASCII value for a is %d", my_a);
```

```
    printf("\nASCII value for z is %d",my_z);
```

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

```
    printf("\n65 in ASCII represents %c",65);
```

```
    printf("\n90 in ASCII represents %c",90);
```

```
    printf("\n97 in ASCII represents %c",97);
```

```
    printf("\n122 in ASCII represents %c",122);
```

```
}
```

Sample output

ASCII value for A is 65

ASCII value for Z is 90

ASCII value for a is 97

ASCII value for z is 122

65 in ASCII represents A

90 in ASCII represents Z

97 in ASCII represents a

122 in ASCII represents z

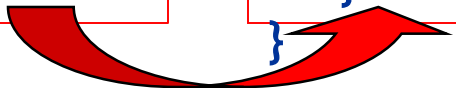
Example cont...

```
#include <stdio.h>
void main(void)
{
    char ch;
    printf("enter a character: ");
    scanf("%c", &ch);
    if (ch >= 'A' && ch <= 'Z')
    {
        printf("\ncapital
letter\n");
    }
}
```

```
#include <stdio.h>
void main(void)
{
    char ch;

    printf("enter a character: ");
    scanf("%c", &ch);

    if (ch >= 65 && ch <=
(65+26))
    {
        printf("\ncapital
letter\n");
    }
}
```



equivalent
to

Fundamentals of Characters and Strings

- ▶ A string in C is an **array** of characters ending with the null character (“\0”). It is written inside a double quotation mark (“ ”)
- ▶ A string may be assigned (in a declaration) to either a char array or to a char pointer:
 - `char color[] = “green”;` OR
 - `char *color = “green”;`

Fundamentals of Characters and Strings

- ▶ A string can also be defined by specifying the individual characters:
 - `char color[] = {'g', 'r', 'e', 'e', 'n', '\0'};`
- ▶ A string is accessed via a pointer to the first character in the string.
- ▶ In memory, these are the characters stored:

g	r	e	e	n	\0
----------	----------	----------	----------	----------	-----------

Fundamentals of Characters and Strings

- ▶ Notice that even though there are only five characters in the word 'green', six characters are stored in the computer. The last character, the character '\0', is the NULL character which indicates the end of the string.
- ▶ Therefore, if an array of characters is to be used to store a string, the array must be large enough to store the string and its terminating NULL character.

Briefly review about strings :

- ▶ We can initialize string variables at compile time such as;
 - `char name[10] = "Arris";`
 - This initialization creates the following spaces in storage :

A	r	r	i	s	\0	\0	\0	\0	\0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Fundamentals of Characters and Strings

- ▶ If we happen to declare a string like this:
`char my_drink[3] = "tea";`
- ▶ We will get the following syntax error:
error C2117: 'tea' : array bounds overflow
- ▶ Instead, we need to at least declare the array with (the size of the string + 1) to accommodate the null terminating character `'\0'`.
`char my_drink[4] = "tea";`

Example: string and '\0'

```
#include <stdio.h>

void main(void) /* a program that counts the number of characters in a
string */
{
    char sentence[] = "I love C language";

    int i, count = 0;

    for (i = 0; sentence[i] != '\0'; i++)
    {
        count++;
    }

    printf("%s has %d characters including the whitespace", sentence,
count);
}
```

Sample output:

I love C language has 17 characters including the whitespace

Briefly review about strings :

- ▶ Standard Functions Input
 - scanf()
 - gets()
- ▶ Standard Functions Output
 - printf()
 - puts()
- ▶ Use scanf function together with the format specifier %s for interactive input string. (no whitespace character)
- ▶ If the string to be read as an input has embedded whitespace characters, use standard *gets* function.

String Handling in C :

String :

A string is a collection of characters. Strings are always enclosed in double quotes as "string constant".

Strings are used in string handling operations such as,

- ✓ Counting the length of a string.
- ✓ Comparing two strings.
- ✓ Copying one string to another.
- ✓ Converting lower case string to upper case.
- ✓ Converting upper case string to lower case.
- ✓ Joining two strings.
- ✓ Reversing string.

String Handling in C :

Declaration :

The string can be declared as follow :

Syntax:

```
char string_nm[size];
```

Example:

```
char name[50];
```


String Handling in C :

String Structure :

When compiler assigns string to character array then it automatically supplies **null character ('\0')** at the end of string. Thus, size of string = original length of string + 1.

```
char name[7];  
name = "TECHNO"
```

'T'	'E'	'C'	'H'	'N'	'O'	'\0'
1	2	3	4	5	6	7

String Handling in C :

Read Strings :

To read a string, we can use scanf() function with format specifier %s.

```
char name[50];  
scanf("%s",name);
```

The above format allows to accept only string which does not have any blank space, tab, new line.

Write Strings :

To write a string, we can use printf() function with format specifier %s.

```
char name[50];  
scanf("%s",name);  
printf("%s",name);
```

String handling functions :

string.h header file :

'string.h' is a header file which includes the declarations, functions, constants of string handling utilities. These string functions are widely used today by many programmers to deal with string operations.

Some of the standard member functions of string.h header files are,

Function Name	Description
strlen –	Returns the length of a string.
strlwr – lower case.	Returns upper case letter to lower case.
strupr – upper case.	Returns lower case letter to upper case.
strcat –	Concatenates two string.
strcmp –	Compares two string.
strrev –	Returns length of a string.
strcpy – destination.	Copies a string from source to destination.

String handling functions :

Program :

```
/* Program to demonstrate string.h header file
working.
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[50];
    clrscr();
    printf("\n\t Enter your name : ");
    gets(str);
    printf("\nLower case of string:
%s",strlwr(str));
    printf("\nUpper case of string:
%s",strupr(str));
    printf("\nReverse of string: %s",strrev(str));
    printf("\nLength of String: %d",strlen(str));
    getch();
}
```

Output :

```
Enter your name : Technoexam
Lower case of string:
technoexam Upper case of
string: TECHNOEXAM Reverse of
string: MAXEONHCET Length of
String: 10_
```