CHAPTER

16

C Strings

The string in C programming language is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a **null-terminated** string contains the characters that comprise the string followed by a null.

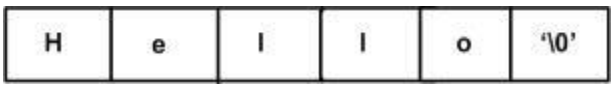
The following declaration and initialization create a string consisting of the word **"Hello"**. To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word **"Hello"**.

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

If you follow the rule of array initialization then you can write the above statement as follows:

```
char greeting[] = "Hello";
```

Following is the memory presentation of above-defined string in C/C++:



Actually, you do not place the null character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print above mentioned string:

```
#include <stdio.h>

int main ()
{
   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

   printf("Greeting message: %s\n", greeting );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
Greeting message: Hello
```

C supports a wide range of functions that manipulate null-terminated strings:

S.N.	Function & Purpose
1	**strcpy(s1, s2);** Copies string s2 into string s1.
2	**strcat(s1, s2);** Concatenates string s2 onto the end of string s1.
3	**strlen(s1);** Returns the length of string s1.
4	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);** Returns a pointer to the first occurrence of string s2 in string s1.

Following example makes use of few of the above-mentioned functions:

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

int main ()
{
   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ;

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
strcpy( str3, str1) :  Hello
```

```
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```

You can find a complete list of C string related functions in C Standard Library.