Paper Name: Programming Concepts

Topic: Strings in C

Paper Code: BCA GE202

Semester II

By Ms. Manisha Prasad

Head, Assistant Professor,

Department of Computer Science

Patna Women's College

E – mail : manisha_prasad@yahoo.com

STRINGS

Strings are one-dimensional array of characters terminated by a null character '\0'. This null character is treated as a single character by the compiler which is actually a combination of backslash '\' character and zero '0'. Thus a null-terminated string contains the characters of the string followed by a null.

Following are the two ways to initialize a string. To hold the null character at the end of the array, the size of the character array containing the string has to be one more than the number of characters in the word.

1. char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; or char str[] = {'H', 'e', 'l', 'l', 'o', '\0'};
   or
2. char str[6] = "Hello"; or char str[] = "Hello";
   If we do not initialize the index with the size of array, it is automatically computed by the compiler as per the initialization.

The memory presentation of the above defined string variable str[6] is:-

str[0]	str[1]	str[2]	str[3]	str[4]	str[5]
H	e	l	l	o	'\0'

In the first method of initialization, we have to specifically mention the null character. In the second approach the C compiler automatically places the '\0' at the end of the string when it initializes the array.
The strings can be printed using a printf() statement or puts() statement. Similarly we can accept a string as an input using scanf() statement or gets() statement. Unlike numeric arrays we do not need to print a string, element by element(character by character). The C language does not provide an inbuilt data type for strings but it has an access specifier "**%s**" which can be used to directly print and read strings.

Example1:

```
#include <stdio.h>
void main()
{
  char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
  printf("Greeting message: %s\n", str );
  getch();
}
```

On execution the output of the program will be:-

Greeting message: Hello

The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab etc.).

Example 2: scanf() to read a string

```
#include <stdio.h>
int main()
{
   char str[20];
   clrscr();
   printf("Enter name: ");
   scanf("%s", str);
   printf("Your name is %s", str);
   getch():
}
```

On execution the output of the program will be:-

```
Enter name: Amit Sinha
Your name is Amit.
```

Even though Amit Sinha was entered in the above program, only "Amit" was stored in the name string. It's because there was a space after Amit.

How to read and display a string containing white space character?

We can use gets() and puts() statement

Example 3: gets() and puts()

```
#include <stdio.h>
# include <conio.h>
void main()
{
   char str[20];
   clrscr();
   printf("Enter name: ");
   gets(str);            // read string
   printf("Name: ");
   puts(str);    // display string
   getch();
}
```

On execution the output of the program will be:-

```
Enter name: Amit Sinha
Name: Amit Sinha
```

We can also use the fgets() function to read a line of string and puts() to display the string.

Example 4: fgets() and puts()

```
#include <stdio.h>
# include <conio.h>
int main()
{
   char str[20];
   printf("Enter name: ");
   fgets(str, sizeof(str), stdin);  // read string
   printf("Name: ");
   puts(str);    // display string
   return 0;
}
```

Output:

```
Enter name: Amit Sinha
Name: Amit Sinha
```

Note: Both fgets() or gets() function take input from the user. Since gets() allows you to input any length of characters. Hence, there might be a buffer overflow. The sizeof(str) results to 20. Hence, we can take a maximum of 20 characters as input which is the size of the str[] string.

Many times we need to manipulate strings according to the need of a problem. Most the time string manipulation can be done manually but this makes programming complex and large. To solve this problem, C supports a large number of string handling functions defined in the header file "string.h" present in the standard library

Few commonly used string handling functions are discussed below:

Function	Work of Function
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

The following example uses some of the above-mentioned functions −

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str1[10] = "Hello";
   char str2[10] = "World";
   char str3[20];
   int  len ;

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

  /*concatenates(joins) str1 and str2 and after joining stores the resultant string in str1*/
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total length 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 the following result −

```
strcpy( str3, str1) :  Hello
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```