A **string** in C is merely an array of characters. The length of a string is determined by a terminating null character: `'\0'`. So, a string with the contents, say, `"abc"` has four characters: `'a'`, `'b'`, `'c'`, and the terminating null (`'\0'`) character.

The terminating null character has the value zero.

Syntax

In C, string constants (literals) are surrounded by double quotes (`"`), e.g. `"Hello world!"` and are compiled to an array of the specified `char` values with an additional null terminating character (0-valued) code to mark the end of the string. The type of a string constant is `char []`.

The `<string.h>` Standard Header

Because programmers find raw strings cumbersome to deal with, they wrote the code in the `<string.h>` library. It represents not a concerted design effort but rather the accretion of contributions made by various authors over a span of years.

First, three types of functions exist in the string library:

- the `mem` functions manipulate sequences of arbitrary characters without regard to the null character;
- the `str` functions manipulate null-terminated sequences of characters;
- the `strn` functions manipulate sequences of non-null characters.

String Library functions – Explanation with Example

Library functions are inbuilt functions provided by the compilers. These library functions can be easily accessed by importing the particular header file which contains the library function. Similarly, there are many library functions that operate on **strings**. These library functions are included in the program by importing the header file **<string.h>**. Let's have a look at the various library functions that operate on strings:

Basics of Strings in C programming

Input/output library functions

We already saw in the last tutorial how to input string from the user. The problem we faced was that the *scanf()* function could only take the input till a blank space or new line. Also, we need to add the **'\0'** afterward. This can be simplified further by using a library function which takes direct string inputs and adds a **'\0'** at the end of the string input. ***gets()*** function takes direct string inputs from the user and terminates when it encounters a new line or reaches the end of the file.

String Library functions – Explanation with Example

Library functions are inbuilt functions provided by the compilers. These library functions can be easily accessed by importing the particular header file which contains the library function. Similarly, there are many library functions that operate on **strings**. These library functions are included in the program by importing the header file **<string.h>**. Let's have a look at the various library functions that operate on strings:

Read more :- Basics of Strings in C programming

Table of Contents

Input/output library functions

We already saw in the last tutorial how to input string from the user. The problem we faced was that the *scanf()* function could only take the input till a blank space or new line. Also, we need to add the **'\0'** afterward. This can be simplified further by using a library function which takes direct string inputs and adds a **'\0'** at the end of the string input. ***gets()*** function takes direct string inputs from the user and terminates when it encounters a new line or reaches the end of the file.

Similarly to print a string we do not need to display the whole string using a loop, which makes the code less readable and also increases the chances of mistakes. Instead of a loop, we can use the ***puts()*** function to display the whole string. These library functions are defined in the header file **<stdio.h>**. Here is a program which shows how to use *gets()* and *puts()* library functions:

```
#include <stdio.h>

int main() {

char name[50]; // declaring a string

printf("Enter a string : ");

gets(name); // using function to take string input

printf("The string is : ");

puts(name); // prints the string that has been entered by the user
```

```
return 0;

}
```

Output:-

```
Enter a string : Codingeek- A home for coders
The string is : Codingeek- A home for coders
```

Other string related library functions

String Library functions – Explanation with Example

Library functions are inbuilt functions provided by the compilers. These library functions can be easily accessed by importing the particular header file which contains the library function. Similarly, there are many library functions that operate on **strings**. These library functions are included in the program by importing the header file **<string.h>**. Let's have a look at the various library functions that operate on strings:

Read more :- Basics of Strings in C programming

Table of Contents

Input/output library functions

We already saw in the last tutorial how to input string from the user. The problem we faced was that the *scanf()* function could only take the input till a blank space or new line. Also, we need to add the **'\0'** afterward. This can be simplified further by using a library function which takes direct string inputs and adds a **'\0'** at the end of the string input. ***gets()*** function takes direct string inputs from the user and terminates when it encounters a new line or reaches the end of the file.

Similarly to print a string we do not need to display the whole string using a loop, which makes the code less readable and also increases the chances of mistakes. Instead of a loop, we can use the ***puts()*** function to display the whole string. These library functions are defined in the header file **<stdio.h>**. Here is a program which shows how to use *gets()* and *puts()* library functions:

```
#include <stdio.h>

int main() {
```

```
char name[50]; // declaring a string

printf("Enter a string : ");

gets(name); // using function to take string input

printf("The string is : ");

puts(name); // prints the string that has been entered by the user

return 0;

}
```

Output:-

```
Enter a string : Codingeek- A home for coders
The string is : Codingeek- A home for coders
```

Other string related library functions

There are other string related library functions which operate on the string and produces certain results. Here are some of the commonly used **string library functions**:

- ***strlen()***: This function returns the length of the string. Example: **strlen(name);** This will return the length of the string stored in the variable *name[]*.
- ***strcat()***: This function concatenates two strings. Example: **strcat(name,name1);** This will concatenate the strings stored in the variables *name[]* and *name1[]* in the order in which it is written.
- ***strcpy()***: This function copies the value of the second string to the first string. Example: **strcpy(name1,name);** This will copy the string in *name[]* to the variable *name1[]*.
- ***strcmp()***: It compares two strings. Example: **strcmp(name,name1);** This compares the string in *name[]* with the string in *name1[]*. It returns 0 if the strings are same. It returns a value less than 0 if *name[]<name1[]*. Otherwise, it returns a value greater than 0.
- ***strlwr()***: It changes all the characters of the string to lower case. Example: **strlwr(name);** It shall convert the whole string stored in the variable *name[]* to lowercase.
- ***strupr()***: It changes all the characters of the string to upper case. Example: **strupr(name);** It shall convert the whole string stored in the variable *name[]* to uppercase.
- ***strchr()***: It returns the location or the pointer of the first occurrence of a character in a string. Example: **strchr(name,ch);** It returns the location of the first occurrence of the character in ch in the string *name[]*. It returns null if the character is not found.
- ***strstr()***: It returns the location or the pointer of the first occurrence of one string in another. Example: **strstr(name,name1);** It returns the location of the first occurrence of *name1[]* in *name[]*. It returns null if the string is not found.

These library functions are defined in the header file <**string.h**>.

Program on String Library Functions

Here is a simple program which takes two string inputs from the user and displays the result of the various string library functions:

```
#include <stdio.h>

#include <string.h> //taking string.h header file which defines most of the library functions.

int main() {

char name[50], name1[50], name2[50], ch = 'B'; // declaring strings

char *x; // pointer

printf("Enter 1st string: ");

gets(name); // input 1st string

printf("Enter 2nd string: ");

gets(name1); // input second string

printf("strlen() function: %d\n",

strlen(name)); // prints the length of the name[] string

printf("strcat() function: %s\n",

strcat(name, name1)); // concatenates the two strings and stores the

// result in name[]

strcpy(name2, name); // copying the string in name[] to name2[]

printf("strcpy() function: %s\n", name2); // display the copied string

printf("strcmp() function: %d\n",

strcmp(name, name1)); // compare the two strings

printf("strlwr() function: %s\n",

strlwr(name)); // convert the string to lowercase

printf("strupr() function: %s\n",

strupr(name)); // convert the string to uppercase

x = strchr(name, ch); // store the pointer of the character in the variable

printf("strchr()function. The string after ch is: %s\n",

x); // display the rest of the string
```

```
x = strstr(name, name1); // stores the pointer of the string in the variable

printf("strstr() function: %s", x); // displays the rest of the string.

return 0;

}
```

Output:-

```
Enter 1st string:  The bell rings
Enter 2nd string:  bell
strlen() function: 14
strcat() function: The bell ringsbell
strcpy() function: The bell ringsbell
strcmp() function: -1
strlwr() function: the bell ringsbell
strupr() function: THE BELL RINGSBELL
strchr()function. The string after ch is: BELL RINGSBELL
strstr() function:
```