Strings in C

A String in C programming is a sequence of characters terminated with a null character '\0'. The C String is stored as an array of characters. The difference between a character array and a C string is the string is terminated with a unique character '\0'.

C String Declaration Syntax

Declaring a string in C is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

```
char string_name[size];
```

In the above syntax **str_name** is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.
There is an extra terminating character which is the ***Null character ('\0') used to indicate the termination of a string that differs strings from normal character arrays***.

C String Initialization

A string in C can be initialized in different ways. We will explain this with the help of an example. Below are the examples to declare a string with the name str and initialize it with "GeeksforGeeks".

Ways to Initialize a String in C

We can initialize a C string in 4 different ways which are as follows:

1. Assigning a string literal without size

String literals can be assigned without size. Here, the name of the string str acts as a pointer because it is an array.

```
char str[] = "GeeksforGeeks";
```

2. Assigning a string literal with a predefined size

String literals can be assigned with a predefined size. But we should always account for one extra space which will be assigned to the null character. If we want to store a string of size n then we should always declare a string with a size equal to or greater than n+1.

```
char str[50] = "GeeksforGeeks";
```

3. Assigning character by character with size

We can also assign a string character by character. But we should remember to set the end character as '\0' which is a null character.

```
char str[14] = { 'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'};
```

4. Assigning character by character without size

We can assign character by character without size with the NULL character at the end. The size of the string is determined by the compiler automatically.

```
char str[] = { 'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'};
```

Note: *When a Sequence of characters enclosed in the double quotation marks is encountered by the compiler, a null character '\0' is appended at the end of the string by default.*

Below is the memory representation of the string "Geeks".

Note: *After declaration, if we want to assign some other text to the string, we have to assign it one by one or use built-in strcpy() function because the direct assignment of string literal to character arrray is only possible in declration.*

Let us now look at a sample program to get a clear understanding of declaring, and initializing a string in C, and also how to print a string with its size.

C String Example

- C

```c
// C program to illustrate strings

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

int main()
{
    // declare and initialize string
    char str[] = "Geeks";

    // print string
    printf("%s\n", str);

    int length = 0;
    length = strlen(str);
    // displaying the length of string
    printf("Length of string str is %d", length);

    return 0;
}
```

Output

```
Geeks
Length of string str is 5
```

We can see in the above program that strings can be printed using normal printf statements just like we print any other variable. Unlike arrays, we do not need to print a string, character by character.

Note: *The C language does not provide an inbuilt data type for strings but it has an access specifier “**%s**” which can be used to print and read strings directly.*

Read a String Input From the User

The following example demonstrates how to take string input using scanf() function in C

- C

```c
// C program to read string from user
#include<stdio.h>

int main()
{
    // declaring string
    char str[50];

    // reading string
    scanf("%s",str);

    // print string
    printf("%s",str);

    return 0;
}
```

Input

```
GeeksforGeeks
```

Output

```
GeeksforGeeks
```

You can see in the above program that the string can also be read using a single scanf statement. Also, you might be thinking that why we have not used the ‘&’ sign with the string name ‘str’ in scanf statement! To understand this you will have to recall your knowledge of scanf.
We know that the ‘&’ sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces ‘[‘ and ‘]’ will give the base address of this string. That’s why we have not used ‘&’ in this case as we are already providing the base address of the string to scanf.

Now consider one more example,

- C

```c
// C Program to take input string which is separated by
// whitespaces
#include <stdio.h>

// driver code
int main()
{

    char str[20];
    // taking input string
    scanf("%s", str);

    // printing the read string
    printf("%s", str);

    return 0;
}
```

Input

```
Geeks for Geeks
```

Output

```
Geeks
```

Here, the string is read-only till the whitespace is encountered. To read the string containing whitespace characters we can use the methods described below:

How to Read a String Separated by Whitespaces in C?

We can use multiple methods to read a string separated by spaces in C. The two of the common ones are:

1. We can use the fgets() function to read a line of string and gets() to read characters from the standard input (stdin) and store them as a C string until a newline character or the End-of-file (EOF) is reached.
2. We can also scanset characters inside the scanf() function

1. Example of String Input using gets()

- C

```c
// C program to illustrate

// fgets()

#include <stdio.h>

#define MAX 50

int main()

{

    char str[MAX];


    // MAX Size if 50 defined

    fgets(str, MAX, stdin);


    printf("String is: \n");


    // Displaying Strings using Puts

    puts(str);


    return 0;

}
```

Input

```
GeeksforGeeks
```

Output

```
String is:
GeeksforGeeks
```

2. Example of String Input using scanset

- C

```c
// C Program to take string separated by whitespace using
// scanset characters
#include <stdio.h>

// driver code
int main()
{

    char str[20];

    // using scanset in scanf
    scanf("%[^\n]s", str);

    // printing read string
    printf("%s", str);

    return 0;
}
```

Input

```
Geeks for Geeks
```

Output

```
Geeks for Geeks
```

C String Length

The length of the string is the number of characters present in the string except for the NULL character. We can easily find the length of the string using the loop to count the characters from the start till the NULL character is found.

Passing Strings to Function

As strings are character arrays, we can pass strings to functions in the same way we pass an array to a function. Below is a sample program to do this:

- C

```c
// C program to illustrate how to

// pass string to functions

#include <stdio.h>


void printStr(char str[]) { printf("String is : %s", str); }


int main()

{

    // declare and initialize string

    char str[] = "GeeksforGeeks";


    // print string by passing string

    // to a different function

    printStr(str);


    return 0;

}
```

Output:

```
String is : GeeksforGeeks
```

Note: We can't read a string value with spaces, we can use either gets() or fgets() in the C programming language.

Strings and Pointers in C

In Arrays, the variable name points to the address of the first element. Similar to arrays, In C, we can create a character pointer to a string that points to the starting address of the string which is the first character of the string. The string can be accessed with the help of pointers as shown in the below example.

- C

```c
// C program to print string using Pointers

#include <stdio.h>


int main()

{
    char str[20] = "GeeksforGeeks";

    // Pointer variable which stores
    // the starting address of
    // the character array str
    char* ptr = str;

    // While loop will run till
    // the character value is not
    // equal to null character
    while (*ptr != '\0') {
        printf("%c", *ptr);

        // moving pointer to the next character.
        ptr++;
    }

    return 0;
}
```

Output

```
GeeksforGeeks
```

Standard C Library – String.h Functions

The C language comes bundled with <string.h> which contains some useful string-handling functions. Some of them are as follows:

Function Name	Description
strlen(string_name)	Returns the length of string name.
strcpy(s1, s2)	Copies the contents of string s2 to string s1.

Function Name	Description
strcmp(str1, str2)	Compares the first string with the second string. If strings are the same it returns 0.
strcat(s1, s2)	Concat s1 string with s2 string and the result is stored in the first string.
strlwr()	Converts string to lowercase.
strupr()	Converts string to uppercase.
strstr(s1, s2)	Find the first occurrence of s2 in s1.

Must Read:

- puts() vs printf() to print a string
- Swap strings in C
- Storage for strings in C
- gets() is risky to use!