Strings

The string can be defined as the one-dimensional array of characters terminated by a null ('\0'). The character array or the string is used to manipulate text such as word or sentences. Each character in the array occupies one byte of memory, and the last character must always be 0. The termination character ('\0') is important in a string since it is the only way to identify where the string ends. When we define a string as char s[10], the character s[10] is implicitly initialized with the null in the memory.

Declaring a String in C

A String in C is an array with character as a data type. C does not directly support string as a data type, as seen in other programming languages like C++. Hence, character arrays must be used to display a string in C. The general syntax of declaring a string in C is as follows:

char variable[array_size];

Example:-

```
char str[5];
char str2[50];
```

Initializing a String in C

There are four methods of initializing a string in C:

1. Assigning a string literal with size

We can directly assign a string literal to a character array keeping in mind to keep the array size at least one more than the length of the string literal that will be assigned to it.

Note :While setting the initial size, we should always account for one extra space used by the null character. If we want to store a string of size **n**, we should set the initial size to be **n+1**.

For example:

```
char str[8] = "Scaler.";
```

The string length here is 7, but we have kept the size to be 8 to account for the Null character. The compiler adds the Null character(\0) at the end automatically**.**

Note: If the array cannot accommodate the entire string, it only takes characters based on its space.

For example:

```
char str[3] = "Scaler.";
printf("%s",str);
```

Output:

```
Sca
```

2. Assigning a string literal without size

It is also possible to directly assign a string literal to a character array without any size. The size gets determined automatically by the compiler at compile time.

char str[] = "Scaler.";

The critical thing to remember is that the name of the string, here "***str***" acts as a pointer because it is an array.

3. Assigning character by character with size

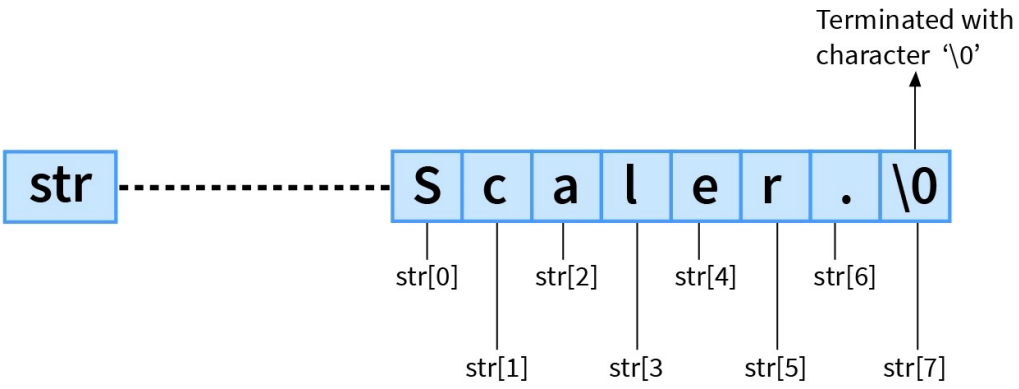
We can also assign a string character by character. However, it is essential to set the end character as '\0'. For example:

char str[8] = {'S', 'c', 'a', 'l', 'e', 'r', '.','\0'};

4. Assigning character by character without size

Like assigning directly without size, we also assign character by character with the Null Character at the end. The compiler will determine the size of the string automatically.

char str[] = {'S', 'c', 'a', 'l', 'e', 'r', '.','\0'};



Reading strings:-

1.gets() function in C

gets() is a pre-defined function in C which is used to read a string or a text line. And store the input in a well-defined string variable. The function terminates its reading session as soon as it encounters a newline character.

Syntax:

gets(variable name);

Example:-

gets(str);

2.scanf() function

scanf(), the most popular method for reading strings, reads a string of characters until it runs into whitespace (i.e., space, newline, tab, etc.). The string may be read by using the %s conversion with the function scanf().

Syntax:-

scanf(“%s”, variable);

Example :-

scanf(“%s”, month);

Read string with spaces

Read string with spaces by using "%[^\n]" format specifier. The format specifier "%[^\n]" tells to the compiler that read the characters until "\n" is not found.

Example:-

scanf(“%[^\n]s”,str);

String Handling Functions:

String is actually not a basic data type so we can say it as more or less derived data type. Using operators of C language we cannot actually handle or in other words manipulate strings. Hence C is equipped with string handling functions. These functions are included in “**string.h**” header file.

1. String Length, strlen(s):

The strlen() function takes a string as an argument and returns its length. This function calculates the length of a string to count characters up to '\0', excluding '\0' character in the string. The returned value is of type int (an unsigned integer type).

Syntax:-

var = strlen(string);

Example:-

```
int n;
char str[10];
n=strlen(str);
```

Program:-

```
#include <stdio.h>
#include <string.h>
main()
{
	char a[20]="Program";
	char b[20]={'P','r','o','g','r','a','m','\0'};
	printf("Length of string a = %d \n",strlen(a));
	printf("Length of string b = %d \n",strlen(b));
}
```

Output;
Length of string a = 7
Length of string b = 7

2. string Copy:-

This function copies the content of one string into other. While string is copied from source to destination, the null character / delimiter ('\0') is also copied. After this process the contents which were originally present is destination string are lost, so it will only have contents which are copied from source string . For this purpose, storage capacity or size of destination string should be larger and/or equal to source string.

Syntax:-

strcpy(des_string, src_string);
where des_string is destination string and src_string is source string.

Example:-

```
#include<stdio.h>
#include <string.h>
main()
{
	char ch[20],ch2[20];
	printf("Enter the string1:");
	scanf("%s",ch);
	strcpy(ch2,ch);
	printf("Fisrt string is:%s",ch);
	printf("\nSecond string is: %s",ch2);
}
```

Output:
Enter the string1:hai
Fisrt string is:hai
Second string is: hai

3. strcmp()

To compare two strings to know whether they are same or not we can use strcmp() function.This string functions in c, compares two strings. While comparing the strings takes two parameters into account namely –
1. str1
2. str2
On comparing the return value be determined basis the strings setup as shown below. The function returns a definite value that may be either 0, >0, or <0. In this function, the two values passed are treated as case sensitive means “A‟ and “a‟ are treated as different letters.
The values returned by the function are used as:
i) 0 is returned when two strings are the same

ii) If str1<str2 then a negative value is returned
iii) If str1>str2 then a positive value is returned

syntax:-
integer_variable = strcmp(string1, string2);
Example :-
n=strcmp(str1,str2);
Program:-

```
#include<stdio.h>
#include<string.h>
main()
{
	char str1[40], str2[40];
	int d;
	printf("Enter first string:");
	gets(str1);
	printf("Enter second string:");
	gets(str2);
	d = strcmp(str1, str2);
	if(d==0)
	{
		printf("Given strings are same.");
	}
	else
	{
		printf("Given strings are different.");
	}
}
```

Output:-
Enter first string: hai
Enter second string: hello
Given strings are different

4. strcat():
String handling function strcat() is used to concatenate two strings. Concatenation is the process of merging content of one string to another string. In the strcat() operation, the destination string's null character will be overwritten by the source string's first character, and the previous null character would now be added at the end of the new destination string which is a result of strcat() operation.
Syntax:-
strcat(dec_string,src_string);

Example:-

strcat(str1,str2);

Program :-

```
#include<stdio.h>
#include<string.h>
main()
{
	char str1[50], str2[50];
	printf("Enter first string:\n");
	gets(str1);
	printf("Enter second string:\n");
	gets(str2);
	strcat(str1,str2);
	printf("Concatenated string is: %s", str1);
}
```

Output:-

Enter first string: Vijay
Enter second string: Babu
Concatenated string is: Vijay Babu

5. strrev():-

String handling function strrev() is used to reverse string. Function strrev() is used to reverse the content of the string.

Syntax:-

strrev(string);

Program:-

```
#include<stdio.h>
#include<string.h>
main()
{
	char name[40];
	printf("Enter your name: ");
	gets(name);
	strrev(name);
	printf("Reversed name is: %s", name);
}
```

Output:-

Enter your name: vijay
Reversed name is: yajiv

strupr() & strlwr() :-

String handling function strupr() is used to convert all lower case letter in string to upper case. String handling function strlwr() is used to convert all upper case letter in string to lower case.

Syntax:-

```
strupr(string);
strlwr(string);
```

Program:-

```
#include<stdio.h>
#include<string.h>
main()
{
	char str[40];
	printf("Enter string (UPPER CASE):");
	gets(str);
	strlwr(str);
	printf("String in lowercase is:");
	puts(str);
	printf("String in uppercase is:");
	strupr(str);
	puts(str);
}
```

Output:-

```
Enter string (UPPER CASE): SRIKAR
String in lowercase is: srikar
String in uppercase is: SRIKAR
```

Implementation of string copy without using library functions

```
#include <stdio.h>
main()
{
	char s1[100], s2[100],i;
	printf("Enter string s1:");
	scanf("%s",s1);
	for(i = 0; s1[i] != '\0'; ++i) // copy to s2
	{
		s2[i] = s1[i];
	}
	s2[i] = '\0';
	printf("String s2: %s", s2);
}
```

Output:-

```
Enter string s1: srikar
String s2: srikar
```

Implementation of string concatenation without using library functions

```
#include<stdio.h>
main()
{
	char str1[25],str2[25];
	int i=0,j=0;
	printf("\nEnter First String:");
	gets(str1);
	printf("\nEnter Second String:");
	gets(str2);
	while(str1[i]!='\0')
	i++;
	while(str2[j]!='\0')
	{
		str1[i]=str2[j];
		j++;
		i++;
	}
	str1[i]='\0';
	printf("\nConcatenated String is %s",str1);
}
```

Output:-

```
Enter First String: hai
Enter Second String: hello
Concatenated String is haihello
```

To find the length of a string without using library function

```
#include <stdio.h>
main()
{
	char str1[50];
	int i, l = 0;
	printf("Input a string : ");
	scanf("%s", str1);
	for (i = 0; str1[i] != '\0'; i++)
	{
		l++;
	}
	printf("The length of the string %s is : %d", str1, l);
```

```
}
```

Output:-

```
Input a string: srikar
The length of the string srikar is: 6
```