Strings in C

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character ‘\0’.

Char str[]=”hello”

Index →	0	1	2	3	4	5	
str →	h	e	l	l	o		
Address →							

Declaration of strings: Declaring a string is as simple as declaring a one-dimensional array. Syntax for declaring a string.

char str_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. Please keep in mind that there is an extra terminating character which is the Null character (‘\0’) used to indicate the termination of string which differs strings from normal character arrays.

Initializing a String: A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as str and initialize it with “Hello”.

1. char str[] = "Hello";
2. char str[50] = "Hello";
3. char str[] = {'H','e','l','l','o',’\0'};
4. char str[6] = {'H','e','l','l','o',’\0'};

The memory representation of a string

	0	**1**	**2**	**3**	**4**	**5**
str	**H**	**e**	**l**	**l**	**o**	**\0**
Address	**100**	**101**	**102**	**103**	**104**	**105**

```
// C program to illustrate strings
#include<stdio.h>
int main()
{
   char str[] = "Hello";
   printf("%s",str);
   return 0;
}
```

Program to read a string from user:

```
// C program to read strings
#include<stdio.h>
int main()
{
   char str[50];
   scanf("%s",str);
   printf("%s",str);
   return 0;
}
```

‘&’ sign not used the with string name ‘str’ in scanf statement!

1. ‘&’ is used to get the address of the variable. C does not have a string type, String is just an array of characters and an array variable stores the address of the first index location.
2. By default the variable itself points to the base address and therefore to access base address of string, there is no need of adding an extra ‘&’

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

```
// C program to illustrate how to  pass string to functions
#include<stdio.h>
void strfunc(char str[])
{
   printf("String is : %s",str);
}
int main()
{
   char str[] = "Hello how are you";
   strfunc(str);
   return 0;
}
```

Output:

```
String is : Hello how are you
```

String Functions

C String Length: strlen() function

The strlen() function returns the length of the given string. It doesn't count null character '\0'.

```
#include<stdio.h>
#include <string.h>
int main(){
char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
   printf("Length of string is: %d",strlen(ch));
 return 0;
}
```

Output:

```
Length of string is: 10
```

C Copy String: strcpy()

The strcpy(destination, source) function copies the source string in destination.

```
#include<stdio.h>
#include <string.h>
int main(){
 char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
   char ch2[20];
   strcpy(ch2,ch);
   printf("Value of second string is: %s",ch2);
 return 0;
}
```

Output:

```
Value of second string is: javatpoint
```

C String Concatenation: strcat()

The strcat(first_string, second_string) function concatenates two strings and result is returned to first_string.

```
#include<stdio.h>
#include <string.h>
int main(){
  char ch[10]={'h', 'e', 'l', 'l', 'o', '\0'};
   char ch2[10]={'c', '\0'};
   strcat(ch,ch2);
   printf("Value of first string is: %s",ch);
```

```c
 return 0;
}
```

Output:

```
Value of first string is: helloc
```

C Compare String: strcmp()

The strcmp(first_string, second_string) function compares two string and returns 0 if both strings are equal. Here, we are using *gets()* function which reads string from the console.

```c
#include<stdio.h>
#include <string.h>
int main(){
  char str1[20],str2[20];
  printf("Enter 1st string: ");
  gets(str1);//reads string from console
  printf("Enter 2nd string: ");
  gets(str2);
  if(strcmp(str1,str2)==0)
      printf("Strings are equal");
  else
      printf("Strings are not equal");
 return 0;
}
```

Output:

```
Enter 1st string: hello
Enter 2nd string: hello
Strings are equal
```

C Reverse String: strrev()

The strrev(string) function returns reverse of the given string. Let's see a simple example of strrev() function.

```c
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nReverse String is: %s",strrev(str));
 return 0;
}
```

Output:

```
Enter string: javatpoint
String is: javatpoint
Reverse String is: tnioptavaj
```

C String Lowercase: strlwr()

The strlwr(string) function returns string characters in lowercase. Let's see a simple example of strlwr() function.

```c
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nLower String is: %s",strlwr(str));
 return 0;
}
```

Output:

```
Enter string: JAVATpoint
String is: JAVATpoint
Lower String is: javatpoint
```

C String Uppercase: strupr()

The strupr(string) function returns string characters in uppercase. Let's see a simple example of strupr() function.

```c
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nUpper String is: %s",strupr(str));
 return 0;
}
```

Output:

```
Enter string: javatpoint
String is: javatpoint
Upper String is: JAVATPOINT
```

END