Strings:

Definition :

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'.

Strings are actually one-dimensional array of characters terminated by a **null** character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a **null**.

Declaration of Strings:

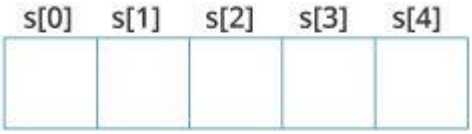
Declaring a string is as simple as declaring a one-dimensional array. Below is the basic syntax for declaring a string.

Syntax : `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.

Example:

```
char s[5];
```



The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

Initialize strings:

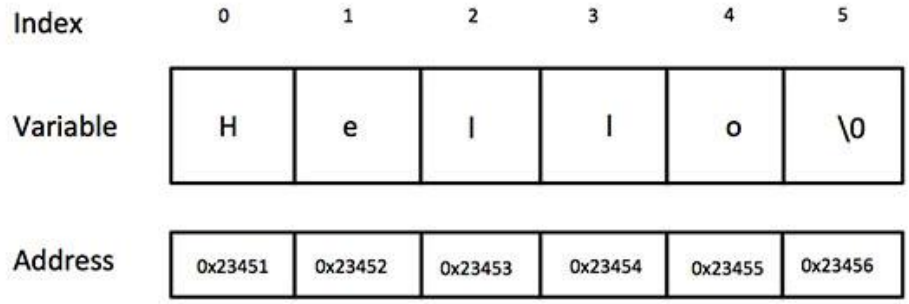
A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as str and initialize it with "Hello".

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

if you follow the rule of array initialization then you can write the above statement as follows

```
char greeting[] = "Hello";
```

Following is the memory presentation of the above defined string in C



You can initialize strings in a number of ways.

- `char c[] = "abcd";`
- `char c[50] = "abcd";`
- `char c[] = {'a', 'b', 'c', 'd', '\0'};`
- `char c[5] = {'a', 'b', 'c', 'd', '\0'};`

C supports a large number of string handling functions in the standard library `"string.h"`.

Actually, you do not place the *null* character at the end of a string constant.

The C compiler automatically places the '\0' at the end of the string when it initializes the array.

Let us try to print the above mentioned string –

Example:

```
int main()
{
  char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
  printf("%s", greeting );
  return 0;
}
```

Result – Hello

C Input and Output Functions:

Input means to provide the program with some data to be used in the program and **Output** means to display data on screen or write the data to a printer or a file.

- C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.

(1) "Printf and Scanf " Read and Display String :

```
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

You can use the `scanf()` function to read a string.

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

Read & write Strings in C using Printf() and Scanf() functions

```
#include <string.h>
int main()
{
        char nickname[20];                    /* String Declaration*/

    printf("Enter your Nick name:");

        scanf("%s", nickname);

        printf("%s",nickname);              /*Displaying String*/


    return 0;
}
```

(2) `getchar()` & `putchar()` functions:

- The `getchar()` function reads a character from the terminal and returns it as an integer.
- This function reads only single character at a time.
- You can use this method in a loop in case you want to read more than one character.
- The `putchar()` function displays the character passed to it on the screen and returns the same character.

This function too displays only a single character at a time. In case you want to display more than one characters, use `putchar()` method in a loop.

Example:

```
void main( )
{
     int c;
     printf("Enter a character");          /*  Take a character as input and
                                               store it in variable c      */

     c = getchar();                        /*  display the character stored   in variable c     */

     putchar(c);

}
```

(3) `gets()` & `puts()` functions:

- The `gets()` function reads a line from **stdin**(standard input) into the buffer pointed to by `str` pointer, until either a terminating newline or EOF (end of file) occurs.
- The `puts()` function writes the string `str.`

```
void main()
{
  /* character array of length 100 */
  char str[100];
  printf("Enter a string");
  gets( str );
  puts( str );
  getch();  }
```

String Handling functions:

SNo	Function	Purpose
1	**strcpy(s1, s2);**	Copies string s2 into string s1.
2	**strcat(s1, s2);**	Concatenates string s2 onto the end of string s1.
3	**strlen(s1);**	Returns the length of string s1.
4	**strcmp(s1, s2);**	Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);**	Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);**	Returns a pointer to the first occurrence of string s2 in string s1.
7	**strlwr()**	converts string to lowercase
8	**strupr()**	converts string to uppercase

1) strlen( ) Function :

strlen( ) function is used to find the length of a character string.

Example: **int n;**

char st[20] = "Bangalore";

n = strlen(st);

- This will return the length of the string 9 which is assigned to an integer variable n.
- Note that the null character "**\0**" available at the end of a string is not counted.

2) strcpy( ) Function :

strcpy( ) function copies contents of one string into another string. Syntax for strcpy function is given below.

Example:

strcpy (str1, str2) – It copies contents of str2 into str1.

strcpy (str2, str1) – It copies contents of str1 into str2.

If destination string length is less than source string, entire source string value won't be copied into destination string.

For example, consider destination string length is 20 and source string length is 30. Then, only 20 characters from source string will be copied into destination string and remaining 10 characters won't be copied and will be truncated.

Example : **char city[15];**

strcpy(city, "BANGALORE") ;

```
void main ()
{
  char str1[12] = "Hello";
  char str2[12] = "World";
  char str3[12];

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

3) strcat( ) Function :

strcat( ) function in C language concatenates two given strings. It concatenates source string at the end of destination string. Syntax for **strcat( )** function is given below.

Example :

strcat (str2, str1); - str1 is concatenated at the end of str2.

strcat (str1, str2); - str2 is concatenated at the end of str1.

- As you know, each string in C is ended up with null character ('**\0**').
- In **strcat( )** operation, null character of destination string is overwritten by source string's first character and null character is added at the end of new destination string which is created after **strcat( )** operation.

```
int main( )
{
   char source[ ] = " ftl" ;
   char target[ ]= " welcome to" ;

   printf ("\n Source string = %s", source ) ;
   printf ( "\n Target string = %s", target ) ;

   strcat ( target, source ) ;

   printf ( "\n Target string after strcat( ) = %s", target ) ;
}
```

Output :

```
Source string = ftl
Target string = welcome to
Target string after strcat() = welcome to ftl
```

4) Strncat() function :

strncat( ) function in C language concatenates (appends) portion of one string at the end of another string.

Example :

strncat (str2, str1, 3); – First 3 characters of str1 is concatenated at the end of str2.
strncat (str1, str2, 3); - First 3 characters of str2 is concatenated at the end of str1.

As you know, each string in C is ended up with null character (‘**\0**’).

In **strncat( )** operation, null character of destination string is overwritten by source string’s first character and null character is added at the end of new destination string which is created after **strncat( )** operation.

```
int main( )
{
   char source[ ] ="aitcse” ;
   char target[ ]= “welcome to” ;
   printf ( “\n Source string = %s”, source ) ;
   printf ( “\n Target string = %s”, target ) ;

   strncat ( target, source, 3 ) ;
   printf ( ””\n Target string after strncat( ) = %s”, target ) ;
}
```

Output :
Source string = aitcse
Target string = welcome to
Target string after strncat()=

welcome to cse

5) strcmp( ) Function :

strcmp( ) function in C compares two given strings and returns zero if they are same.

- If length of string1 < string2, it returns < 0 value.
- If length of string1 > string2, it returns > 0 value.

- **strcmp( )** function is case sensitive. i.e., “A” and “a” are treated as different characters.

Example :

char city[20] = “Madras”;
char town[20] = “Mangalore”;
n=strcmp(city, town);

printf(“%d”,n);

7) strlwr() function :

strlwr() function converts a given string into lowercase.

strlwr() function is non standard function which may not available in standard library in C.

```
int main()
{
    char str[ ] = “MODIFY This String To Lower”;
    printf(“%s\n”, strlwr (str));
    return 0;
}
```

Output :

```
modify this string to lower
```

8) strupr() function :

strupr() function converts a given string into uppercase.

Syntax : char *strupr(char *string);

strupr() function is non standard function which may not available in standard library in C.

```
int main()
{
    char str[ ] = “Modify This String To Upper”;

    printf(“%s\n”, strupr(str));
    return 0;
}
```

Output :

```
MODIFY THIS STRING TO UPPER
```

9) strrev() function :

strrev() function reverses a given string in C language.

Syntax : char *strrev(char *string);

strrev() function is non standard function which may not available in standard library in C.

Example :

```
int main()
{
   char name[30] = “Hello”;

   printf(“String before strrev( ) : %s\n”, name);

   printf(“String after strrev( )  : %s”, strrev(name));

   return 0;
}
```

Output :

```
String before strrev( ) : Hello
String after strrev( ) : olleH
```

10) strchr() function :

strchr() function returns pointer to the first occurrence of the character in a given string.

- **strchr( )** function is used to locate first occurrence of the character ‘i’ in the string ”This is a string ”. **Character ‘i’ is located at position 3** and pointer is returned at first occurrence of the character ‘i’.

```
#include <string.h>
int main ()
{
  char string[25] =”This is a string “;
  p = strchr (string,'i');
```

Example1:

```
#include <string.h>

void main ()
 {
  char str1[12] = "Hello";
  char str2[12] = "World";
  char str3[12];
  int  len ;

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

  strcat( str1, str2);                 /* concatenates str1 and str2 */
  printf(" %s\n", str1 );

  len = strlen(str1);                  /* total lenghth of str1 after concatenation */
  printf(" %d\n", len );

  getch();
}
```

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

```c
/*write a c program to read and display the string */
#incsude<htdio.l>
int main()
{
    char str1[50];
    char str2[10]={'A','I','T','S','\0'};
    char str3[20]="HELLO WELCOME";
    char str4[30];
    char str5[]="WELCOME TO THE WORLD";
    char *str6="WELCOME TO AITS";
    printf("\nEnter First String:\n");
    gets(str1);
    printf("\nThe First String is:\t");
    puts(str1);
    printf("\nThe Second String is:\t%s\n",str2);
    printf("\nThe Third String is:\t%s\n",str3);
    printf("\nEnter Fourth String:\n");
    scanf("%s",&str4);
    printf("\nThe Fourth String is:\t%s\n",str4);
    printf("\nThe Fifth String is:\t%s\n",str5);
    printf("\nThe Sixth String is:\t%s\n",str6);
    return 0;
}
/*
OUTPUT:
Enter First String:
THIS IS AITS COLLEGE

The First String is:    THIS IS AITS COLLEGE

The Second String is:   AITS

The Third String is:    HELLO WELCOME

Enter Fourth String:
HAI

The Fourth String is:   HAI

The Fifth String is:    WELCOME TO THE WORLD

The Sixth String is:    WELCOME TO AITS
*/
```

```c
/*write a c program to read line of text character by character*/
#include<stdio.h>
int main()
{
    char text[50],ch;
    int i=0;
    printf("ENTER TEXT:\n");
    while(ch!='\n')
    {
        ch=getchar();
        text[i]=ch;
        i++;
    }
    text[i]='\0';
    printf("YOUR TEXT IS:\t%s",text);
    return 0;
}
/*
OUTPUT:
ENTER TEXT:
THIS IS AITS
YOUR TEXT IS:   THIS IS AITS
*/
```