Chapter-8
String

String

- A string is a sequence of characters terminated with a null character \0'
- C' always treats a string a single data even though it contains whitespaces.
- A single character is defined using single quote representation. A string is represented using double quote marks.
- 'C' provides standard library **<string.h>** that contains many functions which can be used to perform complicated operations easily on Strings in C.

Declare and Initialize a String in C

A C String is a simple array with char as a data type. 'C' language does not directly support string as a data type. Hence, to display a String in C, you need to make use of a character array.

How to declare a string and initialize string?
There are two ways to declare a string in c language.

1. By char array
2. By string literal

1. By using char array

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

While declaring string, size is not mandatory. So we can write the above code as given below:

char g[] = {'H', 'e', 'l', 'l', 'o', '\0'};

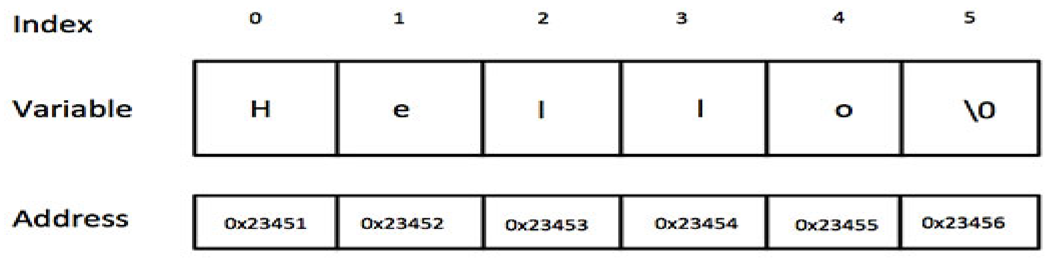
2. By string literal

char g[] = "Hello";
char g[6] = "Hello";

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

char g[] = "Hello"; **is the most approachable way.**

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



Difference between char array and string literal

There are two main differences between char array and literal.

- We need to add the null character '\0' at the end of the array by ourself whereas, it is appended internally by the compiler in the case of the string literal.
- The string literal cannot be reassigned to another set of characters whereas, we can reassign the characters of the array.

Program to display a string.

```
#include<stdio.h>
int main()
{
  char name[]={'n','e','h','a','\0'};
  printf("Name is %s",name);

  return 0;
}
```

Output:

```
Name is neha
```

Program to display a string using null character in between.

```
#include<stdio.h>
int main()
{
  char name[]={'N','e','\0','h','a'};
  printf("Name is %s",name);

  return 0;
}
```

Output:

```
Ne
```

Program to display each character of a string.

```
#include<stdio.h>
int main()
{
  char n[]="Morning";
  int i=0;

          printf("%c\n",n[0]);
          printf("%c\n",n[1]);
          printf("%c\n",n[2]);
          printf("%c\n",n[3]);
          printf("%c\n",n[4]);
```

```
            printf("%c\n",n[5]);
            printf("%c\n",n[6]);
  return 0;
}
```

Output:

```
M
o
r
n
i
n
g
```

Program to display a string using loop.

```
#include<stdio.h>
int main()
{
  char n[]="hello how are you?";
  int i=0;

  while(n[i]!='\0')
  {
            printf("%c",n[i]);
             i++;
  }
  return 0;
}
```

String Functions

There are many important string functions defined in "string.h" library.
list of string functions are as follows:

SR NO	STRING FUNCTION	DESCRIPTION
1	strlen()	returns the length of string.
2	strrev()	reverse string.
3	strlwr()	string characters in lowercase.
4	strupr()	string characters in uppercase.
5	strcmp()	compares the first string with second string. If both strings are same, it returns 0.
6	strcpy()	copies the contents of source string to destination string.
7	strcat()	concats or joins first string with second string. The result of the string is stored in first string.

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

syntax:
strlen(string);

Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
char ch[20]={'s','a','h','y','o','g','\0'};

 printf("Length of string %s is: %d",ch,strlen(ch));
 return 0;
}
```

Output:

```
 Length of string sahyog is: 6
```

2. **strrev()**
   The strrev(string) function returns reverse of the given string.

   Syntax:
   strrev(string)

   Example:

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

Output:

```
Enter string: priya
String is: priya
Reverse String is: ayirp
```

3. **strlwr()**

The strlwr function returns string characters in lowercase.

syntax:
strlwr(string)
Example

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str[20];
  printf("Enter string: ");
  gets(str);
  printf("String is: %s",str);
  printf("\n String in Lowercase : %s",strlwr(str));
  return 0;
}
```

Output:

```
Enter string: PROGRAMMING IN C
String is: PROGRAMMING IN C
 String in Lowercase : programming in c
```

4. **strupr()**

The strupr function returns string characters in uppercase.

Syntax:
strupr(string)

Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str[20];
  printf("Enter string: ");
  gets(str);
  printf("String is: %s",str);
  printf("\n String in Uppercase : %s",strupr(str));
  return 0;
}
```

Output:

```
Enter string: Good Morning
String is: Good Morning
String in Uppercase : GOOD MORNING
```

5. **strcpy():**

- We can not assign a string value directly to another string variable.to assign a string value we can use strcpy function.
- The strcpy() function copies the source string in destination.

Syntax:
strcpy(destination, source)
Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str[20],t[20];
  printf("Enter string: ");
  gets(str);
  printf("String is: %s\n",str);

  strcpy(t,str);
  printf("t=%s",t);
  return 0;
}
```

Output:

```
Enter string: programming
String is: programming
t=programming
```

6. **strcat()**

The strcat() function concatenates two strings and result is returned to destination string.

syntax:
strcat(destination, source)

Example:

```
#include <stdio.h>
#include <string.h>
int main() {
   char str1[100] = "C++ is ", str2[] = "programming Language";

   // concatenates str1 and str2
   // the resultant string is stored in str1.
   strcat(str1, str2);
```

```
    puts(str1);
    puts(str2);

    return 0;
  }
```

Output:

```
C++ is programming Language
programming Language
```

Note: When we use strcat(), the size of the destination string should be large enough to store the resultant string. If not, we will get the segmentation fault error.

7. strcmp()

The strcmp() function compares two strings and returns 0 if both strings are equal.

Return Value from strcmp()

Return Value	**Remarks**
0	if both strings are identical (equal)
negative	if the ASCII value of the first unmatched character is less than the second.
positive integer	if the ASCII value of the first unmatched character is greater than the second.

syntax:

strcmp(str1,str2)

Example2:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str1[20]="HELLO",str2[20]="Hello";
  char str3[20]="neha",str4[20]="Priya";
    char str5[20]="sahyog",str6[20]="sahyog";

  int ch1,ch2,ch3;
  ch1=strcmp(str1,str2);
```

```
    ch2=strcmp(str3,str4);
    ch3=strcmp(str5,str6);

  printf("ch1=%d\n",ch1);
  printf("ch2=%d\n",ch2);
  printf("ch3=%d\n",ch3);

   return 0;
  }
```

Output:

```
ch1=-1
ch2=1
ch3=0
```

Example2:

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

Output 1:

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

Output 2:

```
Enter 1st string: PRIYA
Enter 2nd string: Priya
Strings are not equal
```

***