Strings in C

String :
-> One dimensional character array.
-> It is a collection of characters & symbols.

syntax :

```
        char  identity[size];
```

ex :

```
        char  name[20];
```

-> We represents the string with double quotes.

```
        char  name[5] = "amar" ;
```

-> Last character of String is null('\0')
-> '\0' character will be appended automatically at the end of String.
-> '\0' character ascii value is 0 and the symbol is blank

```
#include<stdio.h>
int main()
{
        char sym='\0';
        printf("Value is : %d \n", sym);
        printf("Character is : %c \n", sym);
        return 0;
}
```

```
#include<stdio.h>
int main()
{
        printf("NULL char value : %d \n" , '\0');
        printf("NULL char symbol : %c \n" , '\0');
        return 0;
}
```

%s :
        -> A format specifier.
        -> Used to read and display strings.

-> Array variable stores base address of memory block
-> String is a character array.
-> String variable also stores base address of block.
-> %s display the string if we specify base address of memory block.

-> We display array elements using loops:
-> For loop display element by element as follows:

```
#include<stdio.h>
```

```
int main()
{
	int arr[5] = {10,20,30,40,50}, i;
	printf("Elements : \n");
	for(i=0 ; i<5 ; i++)
	{
		printf("%d \n", arr[i]);
	}
	return 0;
}
```

"%s" display all characters in the string from base address to '\0' character.

```
#include<stdio.h>
int main()
{
	char arr[20]="Online class";
	printf("String is : %s \n", arr);
	return 0;
}
```

To display character by character, we need to use '\0' character to stop

```
#include<stdio.h>
int main()
{
	char arr[20]="Online";
	int i;
	printf("String is : ");
	for(i=0 ; arr[i]!='\0' ; i++)
	{
		printf("%c", arr[i]);
	}
	return 0;
}
```

```
#include<stdio.h>
int main()
{
	char arr[20]="Online";
	printf("%c \n", arr);
	printf("%c \n", arr[0]);
	return 0;
}
```

We can assign values(char by char) to string type variable:

```
#include<stdio.h>
int main()
{
	char name[20] = {'a','n','n','i','e'} ;
	printf("Hello %s \n", name);
	return 0;
}
```

Display string using while loop:

```
#include<stdio.h>
int main()
{
	char name[20] = "online class";
	int i=0;
	while(name[i] != '\0')
	{
		printf("%c \n",name[i]);
		i++;
	}
	return 0;
}
```

Reading a string:
-> Using '%s' we can read a string.
-> Generally we use loops to read elements into arrays
-> String is a character array but no need to use loops to read elements.
-> No need to specify & operator in the scanf() function to read.

```
#include<stdio.h>
int main()
{
	char name[20];
	printf("Enter your name : ");
	scanf("%s", name);
	printf("Hello %s \n", name);
	return 0;
}
```

Why we are not using loops to read strings?
-> In case of array, we read specified number of elements into array.
-> We use loops to read more than one element.
-> We mention the address of each location to read the element.

```
#include<stdio.h>
int main()
{
	int arr[5],i;

	printf("Enter 5 elements : ");
	for(i=0 ; i<5 ; i++)
```

```
        {
                scanf("%d",&arr[i]);
        }
        return 0;
}
```

-> If we know the length of string, we use loops to read string elements.

```
#include<stdio.h>
int main()
{
        char vowels[6], i;

        printf("Enter vowels : ");
        for(i=0 ; i<6 ; i++)
        {
                scanf("%c", &vowels[i]);
        }

        printf("Vowels are : \n");
        for(i=0 ; i<5 ; i++)
        {
                printf("%c\n", vowels[i]);
        }
        return 0;
}
```

-> While reading names or any other strings from the console, we cannot specify its length.
-> %s collects characters one by one and place into locations from base address.
-> When we stop input, it place '\0' character at the end automatically.

```
#include<stdio.h>
int main()
{
        char name[20];

        printf("Enter name : ");
        scanf("%s", name);

        printf("Hello %s \n", name);
        return 0;
}
```

Output :

```
        Enter name : hari haran
        Hello hari
```

"%s" : can read a single word strings.

gets(): We can read multi word strings(sentense, paragraph...), we use gets().
gets() functions stops reading characters into string only when we press enter.

```
#include<stdio.h>
int main()
{
	char name[20];
	printf("Enter name : ");
	gets(name);

	printf("Hello %s \n", name);
	return 0;
}
```

Representing Strings with Quotes:
-> We represents strings with double quotes.
-> If we want to display a sub string or entire string in double quotes, we use escape characters.

```
/*
Output :
	It is 'C-Online' class
*/
#include<stdio.h>
int main()
{
	char str[50] = "It is 'C-Online' class";
	printf("String : %s\n", str);
	return 0;
}


/*
Output :
	It is "Live" session

Escape characters:
	\n = new line
	\t = tab space
	\b = back space
	\r = carriage return
	\' = '
	\" = "
	\\ = \
*/
#include<stdio.h>
int main()
{
	char str[50] = "It is \"Live\" session";
	printf("String : %s\n", str);
	return 0;
}
```

We can read the string directly with quotes:

```
#include<stdio.h>
int main()
{
	char str[50];

	printf("Enter String : ");
	gets(str);

	printf("String : %s\n", str);
	return 0;
}
```

Output:

```
	Enter String : "it is live session"
	String : "it is live session"
```

-> '\0' represents end of string
-> printf() function stops printing the string when it reaches '\0'

```
#include<stdio.h>
int main()
{
	printf("Hello\0World\n");
	return 0;
}
```

Output : \0 character ASCII value is : 0

```
#include<stdio.h>
int main()
{
	printf("\0 ASCII value is : %d \n", '\0');
	return 0;
}
```

```
#include<stdio.h>
int main()
{
	printf("\\0 ASCII value is : %d \n", '\0');
	return 0;
}
```

-> Just like array variables, strings also store base address of memory block.
-> Different strings(including duplicates) will get memory in different locations.

```
#include<stdio.h>
int main()
```

```
{
	char s1[10] = "Hello" ;
	char s2[10] = "Hello" ;
	if(s1==s2)
		printf("Strings are equal \n");
	else
		printf("Strings are not equal \n");
	return 0;
}
```

string.h:

- It is a pre defined header file.
- It is providing functions to process strings.
- Example functions strlen(), strcmp(), strcat()…..

strcmp():

- strcmp() is the pre-defined function.
- Compare both the strings, if equal returns 0 if not equal return non zero.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s1[10] = "Hello";
	char s2[10] = "Hello";

	if(strcmp(s1,s2)==0)
		printf("Strings are equal \n");
	else
		printf("String are not equal \n");
	return 0;
}
```

- strcmp() consider the case while checking equality.
- stricmp() will not consider the case while checking equality.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s1[10] = "hello";
	char s2[10] = "HELLO";

	if(stricmp(s1,s2)==0)
		printf("Strings are equal \n");
	else
		printf("String are not equal \n");
	return 0;
}
```

strncmp() is used to check 'n' characters are equal or not in the input strings.

```
#include<stdio.h>
#include<string.h>
int main()
{
        char s1[20] = "hello";
        char s2[20] = "hello world";

        if(strncmp(s1,s2,5)==0)
                printf("Strings have idential start \n");
        else
                printf("String not have identical start \n");
        return 0;
}
```

Finding the length:

- strlen() function returns length of string.
- It excludes null character.
- size_t is returntype and represents unsigned integer value.
    - size_t strlen(char s[ ]) ;

```
#include<stdio.h>
#include<stdio.h>
#include<string.h>
int main()
{
        char s[20];
        size_t l;
        printf("Enter string : ");
        gets(s);

        l = strlen(s);
        printf("Length is : %u\n", l);
        return 0;
}
```

Display string character by character using functions:

- We can pass String as input to function like arrays.
- We collect this input by defining argument in the function.

```
#include<stdio.h>
void display(char[]);
int main()
{
        char s[20];
        printf("Enter string : ");
```

```
	gets(s);
	display(s);
	return 0;
}
void display(char s[])
{
	int i;
	for(i=0 ; s[i]!='\0' ; i++)
	{
		printf("%c\n", s[i]);
	}
}
```

Finding the length without using library function:

```
#include<stdio.h>
size_t length(char[]);
int main()
{
	char s[20];
	size_t l;
	printf("Enter string : ");
	gets(s);
	l=length(s);
	printf("Length is : %u\n", l);
	return 0;
}
size_t length(char s[])
{
	int i;
	size_t len=0;
	i=0;
	while(s[i]!='\0')
	{
		len++;
		i++;
	}
	return len;
}
```

How to reverse the string:

- strrev() function reverse the string.
- After reverse, it stores the string in the input string variable only.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s[20];
	printf("Enter string : ");
	gets(s);

	strrev(s);
	printf("Reverse string is : %s \n", s);
	return 0;
}
```

Reverse the string without using library function:

```
#include<stdio.h>
#include<string.h>
void reverse(char[]);
int main()
{
	char s[20];
	printf("Enter string : ");
	gets(s);

	reverse(s);
	printf("Reverse string is : %s\n", s);
	return 0;
}
void reverse(char s[])
{
	int i=0;
	int j=strlen(s)-1;
	char t;
	while(i<j)
	{
		t=s[i];
		s[i]=s[j];
		s[j]=t;
		i++;
		j--;
	}
}
```

How to convert upper case characters into lower case: strlwr() converts all upper cases characters into lower case.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s[20];
	printf("Enter upper case string : ");
	gets(s);

	strlwr(s);
```

```
        printf("Lower case string is : %s\n", s);
        return 0;
}
```

Covert into lower case without using library function:

```
#include<stdio.h>
#include<string.h>
int main()
{
        int i;
        char s[20];
        printf("Enter upper case string : ");
        gets(s);

        for(i=0 ; s[i]!='\0' ; i++)
        {
                if(s[i]>='A' && s[i]<='Z')
                s[i]=s[i]+32;
        }
        printf("Lower case string is : %s\n", s);
        return 0;
}
```

Merge 2 strings:

```
#include<stdio.h>
#include<string.h>
int main()
{
        char s1[20], s2[20];
        int i, l1, l2;

        printf("Enter s1 : ");
        gets(s1);

        printf("Enter s2 : ");
        gets(s2);

        l1 = strlen(s1);
        l2 = strlen(s2);

        for(i=0 ; i<=l2 ; i++)
        {
                s1[l1+i] = s2[i];
        }
        printf("After merge s1 is : %s\n", s1);
        return 0;
}
```

Read the string and sort:

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s[20],t;
	int i,j,l;

	printf("Enter string : ");
	gets(s);

	l = strlen(s);
	for(i=0 ; i<l-1 ; i++)
	{
		for(j=0 ; j<l-1-i ; j++)
		{
			if(s[j]>s[j+1])
			{
				t=s[j];
				s[j]=s[j+1];
				s[j+1]=t;
			}
		}
	}
	printf("Sorted string : %s\n", s);
	return 0;
}
```