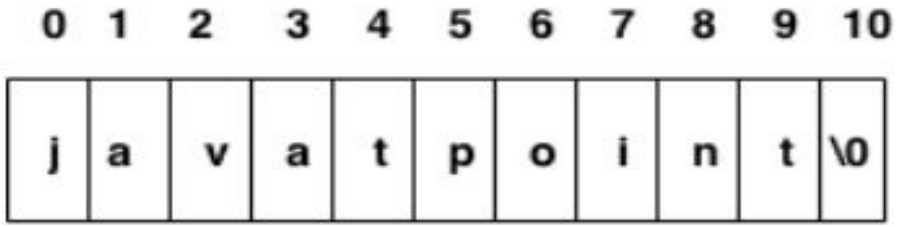
String

- **C 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.
- There are two ways to declare a string in c language.
- By char array
- By string literal
- Let's see the example of declaring string by char array in C language.
- char ch[10]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

- As we know, array index starts from 0, so it will be represented as in the figure given below.



- While declaring string, size is not mandatory. So we can write the above code as given below:
- char ch[]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

- We can also define the string by the string literal in C language. For example:
- char ch[]="javatpoint";
- In such case, '\0' will be appended at the end of the string by the compiler.

- **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 character array.
- The string literal cannot be reassigned to another set of characters whereas, we can reassign the characters of the array.

- **String Example in C**
- Let's see a simple example where a string is declared and being printed. The '%s' is used as a format specifier for the string in c language.

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

```
char ch[11]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
   char ch2[11]="javatpoint";
   printf("Char Array Value is: %s\n", ch);
   printf("String Literal Value is: %s\n", ch2);
 return 0;
}
```

Output

```
Char Array Value is: javatpoint
String Literal Value is: javatpoint
```

- **Traversing String**
- Traversing the string is one of the most important aspects in any of the programming languages.
- We may need to manipulate a very large text which can be done by traversing the text.
- Traversing string is somewhat different from the traversing an integer array.

- We need to know the length of the array to traverse an integer array, whereas we may use the null character in the case of string to identify the end the string and terminate the loop.
- Hence, there are two ways to traverse a string.
- By using the length of string
- By using the null character.
- Let's discuss each one of them.
- **Using the length of string**
- Let's see an example of counting the number of vowels in a string.

```
#include<stdio.h>
void main ()
{
    char s[11] = "javatpoint";
    int i = 0;
    int count = 0;
while(i<11)
    {
```

```
 if(s[i]=='a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'u' || s[i] == 'o')
        {
            count ++;
        }
        i++;
    }
    printf("The number of vowels %d",count);
}
```

- **Output**
- The number of vowels 4
- **Using the null character**
- Let's see the same example of counting the number of vowels by using the null character.

```
#include<stdio.h>
void main ()
{
    char s[11] = "javatpoint";
    int i = 0;
    int count = 0;
    while(s[i] != NULL)
    {
```

```
if(s[i]=='a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'u' || s[i] == 'o')
        {
            count ++;
        }
        i++;
    }
    printf("The number of vowels %d",count);
}
```

Output

The number of vowels 4

Accepting string as the input

Till now, we have used scanf to accept the input from the user. However, it can also be used in the case of strings but with a different scenario. Consider the below code which stores the string while space is encountered.

```
#include<stdio.h>
void main ()
{
    char s[20];
    printf("Enter the string?");
    scanf("%s",s);
    printf("You entered %s",s);
}
```

Output

Enter the string?javatpoint is the best

You entered javatpoint

```
#include<stdio.h>
void main ()
{
    char s[20];
    printf("Enter the string?");
    scanf("%[^\n]s",s);
    printf("You entered %s",s);
}
```

Output

```
Enter the string?javatpoint is the best
You entered javatpoint is the best
```

- **C gets() and puts() functions**
- The gets() and puts() are declared in the header file stdio.h. Both the functions are involved in the input/output operations of the strings.
- **C gets() function**
- The gets() function enables the user to enter some characters followed by the enter key.
- All the characters entered by the user get stored in a character array. The null character is added to the array to make it a string.
- The gets() allows the user to enter the space-separated strings. It returns the string entered by the user.

- **Declaration**
- char[] gets(char[]);

Reading string using gets()

```
#include<stdio.h>
void main ()
{
```

```
char s[30];
    printf("Enter the string? ");
    gets(s);
    printf("You entered %s",s);
}
```

Output

```
Enter the string?
javatpoint is the best
You entered javatpoint is the best
```

- The gets() function is risky to use since it doesn't perform any array bound checking and keep reading the characters until the new line (enter) is encountered.
- It suffers from buffer overflow, which can be avoided by using fgets(). The fgets() makes sure that not more than the maximum limit of characters are read. Consider the following example.

```
#include<stdio.h>
void main()
{
   char str[20];
   printf("Enter the string? ");
   fgets(str, 20, stdin);
   printf("%s", str);
}
```

Output

```
Enter the string? javatpoint is the best website
javatpoint is the b
```

- **C puts() function**
- The puts() function is very much similar to printf() function. The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function.
- The puts() function returns an integer value representing the number of characters being printed on the console.
- Since, it prints an additional newline character with the string, which moves the cursor to the new line on the console, the integer value returned by puts() will always be equal to the number of characters present in the string plus 1.

- **Declaration**
- int puts(char[])
- Let's see an example to read a string using gets() and print it on the console using puts().

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

```
printf("Enter your name: ");
gets(name); //reads string from user
printf("Your name is: ");
puts(name);  //displays string
return 0;
}
```

Output:

```
Enter your name: Sonoo Jaiswal
Your name is: Sonoo Jaiswal
```

- **C String Functions**
- There are many important string functions defined in "string.h" library.

No.	Function	Description
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.

4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
5)	strrev(string)	returns reverse string.
6)	strlwr(string)	returns string characters in lowercase.
7)	strupr(string)	returns string characters in uppercase.

- **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);
 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.

```
#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.

```
#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.

```
#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.

```
#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