**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:**

1. By char array
2. By string literal

Example of declaring string by char array:

```
char ch[10]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
```

As we know, array index starts from 0.

While declaring string, size is not mandatory.

```
char ch[]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
```

**We can also define the string by the string literal:**

```
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 our self-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.

**Example of String:**

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4.   char ch[11]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
5.    char ch2[11]="javatpoint";
6.
7.    printf("Char Array Value is: %s\n", ch);
8.    printf("String Literal Value is: %s\n", ch2);
9.  return 0;
10.}
```

**Output:**

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

**Some important points:**

However, there are the following points which must be noticed while entering the strings by using scanf.

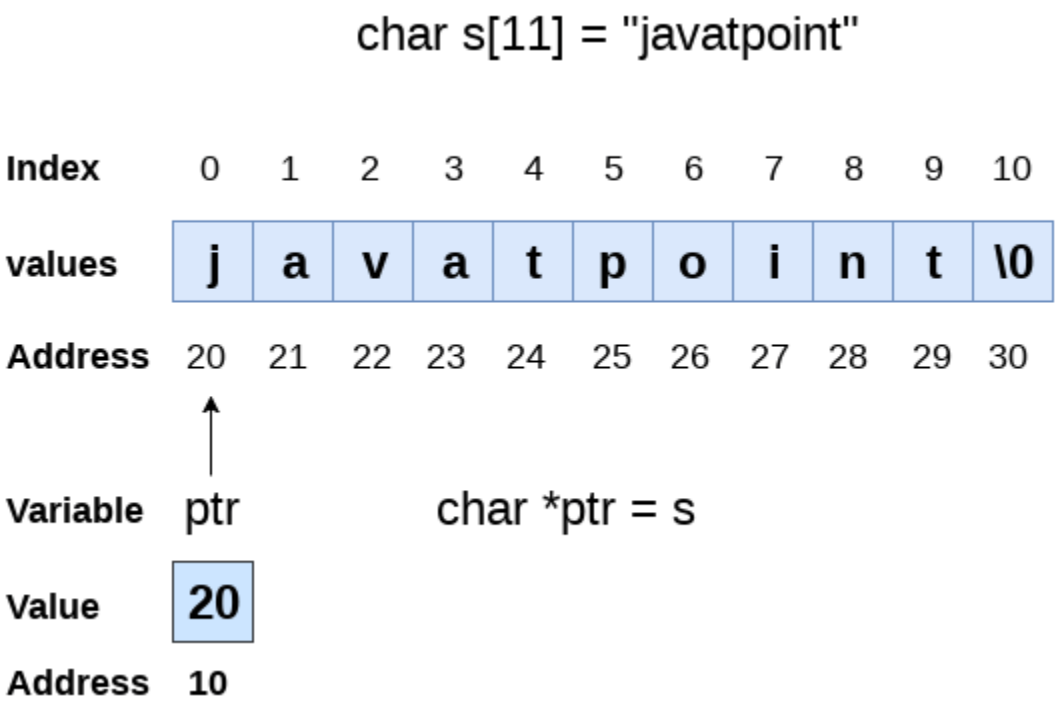
- The compiler doesn't perform bounds checking on the character array. Hence, there can be a case where the length of the string can exceed the dimension of the character array which may always overwrite some important data.
- Instead of using scanf, we may use gets() which is an inbuilt function defined in a header file string.h. The gets() is capable of receiving only one string at a time.

**Pointers with strings:**

We have used pointers with the array, functions, and primitive data types so far. However, pointers can be used to point to the strings. There are various advantages of using pointers to point strings. Let us consider the following example to access the string via the pointer.

```
1. #include<stdio.h>
2. void main ()
3. {
4.     char s[11] = "javatpoint";
5.     char *p = s; // pointer p is pointing to string s.
6.     printf("%s",p); // the string javatpoint is printed if we print p.
7. }
```

**Output:**
javatpoint




As we know that string is an array of characters, the pointers can be used in the same way they were used with arrays. In the above example, p is declared as a pointer to the array of characters s. P affects similar to s since s is the base address of the string and treated as a pointer internally. However, we cannot change the content of s or copy the content of s into another string directly. For this purpose, we need to use the pointers to store the strings. In the following example, we have shown the use of pointers to copy the content of a string into another.

```
1. #include<stdio.h>
2. void main ()
3. {
4.     char *p = "hello javatpoint";
5.     printf("String p: %s\n",p);
6.     char *q;
7.     printf("copying the content of p into q...\n");
8.     q = p;
9.     printf("String q: %s\n",q);
10.}
```

**Output:**

```
String p: hello javatpoint
copying the content of p into q...
String q: hello javatpoint
```

**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.

**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():**

```
1. #include<stdio.h>
2. void main ()
3. {
4.     char s [30];
5.     printf ("Enter the string? ");
```

```
6.      gets(s);
7.      printf ("You entered %s",s);
8.  }
```

**Output:**

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

**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[]);
```

**Example:**

```
1.  #include<stdio.h>
2.  #include <string.h>
3.  int main()
4.  {
5.  char name[50];
6.  printf("Enter your name: ");
7.  gets(name); //reads string from user
8.  printf("Your name is: ");
9.  puts(name);  //displays string
10.return 0;
11.}
```

**Output:**

Enter your name: Disha Computers
Your name is: Disha Computers

**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. |

**String Length: strlen() function:**

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

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4. char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
5.    printf("Length of string is: %d",strlen(ch));
6.  return 0;
7. }
```

**Output:**

Length of string is: 10

**Copy String: strcpy():**

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

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4.  char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
5.    char ch2[20];
6.    strcpy(ch2,ch);
7.    printf("Value of second string is: %s",ch2);
8.  return 0;
9. }
```

**Output:**

Value of second string is: javatpoint

**String Concatenation: strcat()**

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

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4.   char ch[10]={'H', 'e', 'l', 'l', 'o', '\0'};
5.    char ch2[10]={'w','o','r','l','d', '\0'};
6.    strcat(ch,ch2);
7.    printf("New string is: %s",ch);
8.  return 0;
9. }
```

**Output:**

Value of first string is: HelloWorld

**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.

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4.   char str1[20],str2[20];
5.   printf("Enter 1st string: ");
6.   gets(str1);//reads string from console
7.   printf("Enter 2nd string: ");
8.   gets(str2);
9.   if(strcmp(str1,str2)==0)
10.      printf("Strings are equal");
11. else
12.      printf("Strings are not equal");
13. return 0;
14.}
```

**Output:**

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

**Reverse String: strrev()**

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

```
1. #include<stdio.h>
2. #include <string.h>
3. int main(){
4.   char str[20];
```

```
5.    printf("Enter string: ");
6.    gets(str);//reads string from console
7.    printf("String is: %s",str);
8.    printf("\nReverse String is: %s",strrev(str));
9.   return 0;
10.}
```

**Output:**

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

**String Lowercase: strlwr()**

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

```
1.  #include<stdio.h>
2.  #include <string.h>
3.  int main(){
4.    char str[20];
5.    printf("Enter string: ");
6.    gets(str);//reads string from console
7.    printf("String is: %s",str);
8.    printf("\nLower String is: %s",strlwr(str));
9.   return 0;
10.}
```

**Output:**

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

**String Uppercase: strupr()**

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

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

```
2.  #include <string.h>
3.  int main(){
4.    char str[20];
5.    printf("Enter string: ");
6.    gets(str);//reads string from console
7.    printf("String is: %s",str);
8.    printf("\nUpper String is: %s",strupr(str));
9.   return 0;
10.}
```

**Output:**

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