String in C Language

1. Introduction

- In C language, a **string** is a sequence of characters that ends with a **null character (`\0`)**.
- Strings are widely used to store and manipulate text.
- In C, there is **no special string data type**. Strings are represented using **character arrays**.

Example:

```
char str[10] = "Hello";
```

Here, memory stores it as:

```
H   e   l   l   o   \0
```

2. Declaration of Strings

Strings can be declared in two ways:

1. Using character array

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

2. Using string literal (easy method)

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

3. Input and Output of Strings

Output (`printf`)

```
char name[] = "Neeraj";
printf("%s", name);   // Output: Neeraj
```

Input (`scanf`)

```
char name[20];
scanf("%s", name);    // Input: Neeraj
printf("Hello %s", name);
```

Note: `scanf("%s", name)` stops input at the first space.

For full line input (with spaces), use `gets()` or `fgets()`.

```
char sentence[50];
fgets(sentence, sizeof(sentence), stdin);
printf("%s", sentence);
```

4. String Functions (from <string.h>)

C provides many built-in functions for string handling:

1. **`strlen(str)`** – returns length of string (excluding `\0`).

```
printf("%d", strlen("Hello")); // 5
```

2. **`strcpy(dest, src)`** – copies one string into another.

```
char str1[20], str2[20] = "World";
strcpy(str1, str2);  // str1 = "World"
```

3. **`strcat(str1, str2)`** – concatenates two strings.

```
char a[20] = "Hello ", b[] = "World";
strcat(a, b);  // a = "Hello World"
```

4. **`strcmp(str1, str2)`** – compares two strings.

    - Returns `0` if equal.
    - Returns `>0` if str1 > str2.
    - Returns `<0` if str1 < str2.

```
strcmp("abc", "abc"); // 0
strcmp("abc", "abd"); // -1
```

5. **`strupr(str)`** (compiler dependent) – converts to uppercase.
6. **`strlwr(str)`** (compiler dependent) – converts to lowercase.
7. **`strrev(str)`** (compiler dependent) – reverses string.

Static Variables in C Language

Definition

A **static variable** in C is a variable that retains its value **between multiple function calls**. It is initialized only once and its lifetime is throughout the execution of the program.

Key Points:

1. Declared using the keyword **`static`**.
2. **Scope:** Limited to the block/function where it is declared.
3. **Lifetime:** Entire program execution (not destroyed after function ends).
4. **Default value:** `0` (if not explicitly initialized).
5. **Storage:** Stored in the **Data Segment** (not in stack like auto variables).

Syntax

```
static data_type variable_name = value;
```

Example 1: Static inside a function

```
#include <stdio.h>

void demo() {
    static int count = 0; // initialized only once
    count++;
```

```
    printf("Count = %d\n", count);
}

int main() {
    demo();
    demo();
    demo();
    return 0;
}
```

Output

```
Count = 1
Count = 2
Count = 3
```