Static in C

Static is a keyword used in C programming language. It can be used with both variables and functions, i.e., we can declare a static variable and static function as well. An ordinary variable is limited to the scope in which it is defined, while the scope of the static variable is throughout the program.

Static keyword can be used in the following situations:

* **Static global variable**  
  When a global variable is declared with a static keyword, then it is known as a static global variable. It is declared at the top of the program, and its visibility is throughout the program.
* **Static function**  
  When a function is declared with a static keyword known as a static function. Its lifetime is throughout the program.
* **Static local variable**  
  When a local variable is declared with a static keyword, then it is known as a static local variable. The memory of a static local variable is valid throughout the program, but the scope of visibility of a variable is the same as the automatic local variables. However, when the function modifies the static local variable during the first function call, then this modified value will be available for the next function call also.
* **Static member variables**  
  When the member variables are declared with a static keyword in a class, then it is known as static member variables. They can be accessed by all the instances of a class, not with a specific instance.
* **Static method**  
  The member function of a class declared with a static keyword is known as a static method. It is accessible by all the instances of a class, not with a specific instance.

**Let's understand through an example.**

1. #include <stdio.h>
2. **int** main()
3. {
4. printf("%d",func());
5. printf("\n%d",func());
6. **return** 0;
7. }
8. **int** func()
9. {
10. **int** count=0; // variable initialization
11. count++; // incrementing counter variable
13. **return** count; }

In the above code, the func() function is called. In func(), count variable gets updated. As soon as the function completes its execution, the memory of the count variable will be removed. If we do not want to remove the count from memory, then we need to use the count variable as static. If we declare the variable as static, then the variable will not be removed from the memory even when the function completes its execution.

**Output**

1

1

Static variable

A static variable is a variable that persists its value across the various function calls.

**Syntax**

The syntax of a static variable is given below:

1. **static** data\_type variable\_name;

**Let's look at a simple example of static variable.**

1. #include <stdio.h>
2. **int** main()
3. {
4. printf("%d",func());
5. printf("\n%d",func());
7. **return** 0;
8. }
9. **int** func()
10. {
11. **static** **int** count=0;
12. count++;
13. **return** count;
14. }

In the above code, we have declared the count variable as static. When the func() is called, the value of count gets updated to 1, and during the next function call, the value of the count variable becomes 2. Therefore, we can say that the value of the static variable persists within the function call.

**Output**

1

2

Static Function

As we know that non-static functions are global by default means that the function can be accessed outside the file also, but if we declare the function as static, then it limits the function scope. The static function can be accessed within a file only.

**The static function would look like as:**

1. **static** **void** func()
2. {
3. printf("Hello javaTpoint");
4. }

Differences b/w static and global variable

Global variables are the variables that are declared outside the function. These global variables exist at the beginning of the [program](https://www.javatpoint.com/cpp-programs), and its scope remains till the end of the program. It can be accessed outside the program also.

Static variables are limited to the source file in which they are defined, i.e., they are not accessible by the other source files.

Both the static and global variables have static initialization. Here, static initialization means if we do not assign any value to the variable then by default, 0 value will be assigned to the variable.

Differences b/w static local and static global variable

**Static global variable**

If the variable declared with a static keyword outside the function, then it is known as a static global variable. It is accessible throughout the program.

**Static local variable**

The variable with a static keyword is declared inside a function is known as a static local variable. The scope of the static local variable will be the same as the automatic local variables, but its memory will be available throughout the program execution. When the function modifies the value of the static local variable during one function call, then it will remain the same even during the next function call.

Properties of a static variable

**The following are the properties of a static variable:**

* The memory of a static variable is allocated within a static variable.
* Its memory is available throughout the program, but the scope will remain the same as the automatic local variables. Its
* value will persist across the various function calls.
* If we do not assign any value to the variable, then the default value will be 0.
* A global static variable cannot be accessed outside the program, while a global variable can be accessed by other source files.