Content 50

Function Pointer in C

To understand this concept of***function pointers***, you should have the basic knowledge of **Pointers** and **dynamic memory allocation**. We have already discussed pointers and dynamic memory allocation in our previous lectures. The quick review of the pointer is that pointer is a variable containing another variable's address. It means it is a variable, but this variable contains any other variable's address or memory address. The pointer can be of type int, char, array, function, or any other pointer. The symbol \* is used to get the value of the variable that the pointer is pointing to, and the & symbol is to get the address of the variables. In C, we can create a generic pointer like a dangling, null, etc. Now let us revise the concept of dynamic memory allocation.

A statically allocated variable or array has a fixed size in memory. Whereas the dynamic memory allocation is a way in which the size of the data structure can be changed during runtime. The memory assigned to a program in a typical architecture can be broken down into four segments. These are code, global & static variables, stack, and heap.

Now let us move to our main topic, **"Function Pointers in C."** In C programming, the pointer to a function is known as **function pointer in C**. In this tutorial, we will learn how to declare a function pointer in C and call a function using this pointer.

#### Example:-

void (\*fun)(int, char);

In this example, fun is a pointer to a function taking two arguments, an integer and character, and that returns void. It's as if we are declaring a function called "\*fun", which takes an int and char, and returns void; now, if \*fun is a function, then fun must be a pointer to a function. The following are the legal way of declaring function pointers

#### Some Important points regarding function pointer:

1. We can declare a function pointer and assign a function to it in a single statement like this:

void (\*function\_ptr)(int) = &fun;

1. We can remove the ampersand(&) from this statement because a function name alone represents the function address.

void (\*function\_ptr)(int) = fun;

**Code to understand The function Pointer:**

#include <stdio.h>

int sum(int a, int b)

{

    int sum;

    sum = a + b;

    return sum;

}

int main(int argc, char const \*argv[])

{

    printf("Addition of two numbers is: %d \n", sum(2, 3));

    int (\* fptr) (int ,int);  //creating a pointer

    fptr=&sum;                  // pointing a pointer to a funcion

    // int d= fptr (4,2);    or   // it is dereferecing of pointer

    printf("The sum using function pointer is: %d",fptr(4,2));

    return 0;

}

**Output:**

Addition of two numbers is: 5

The sum using function pointer is: 6