**23043**

**MUNYANEZA KENNY ROGER October 1, 2022**

**TYPES OF RECURSION IN C PROGRAMMING**

**Recursion** is the process in which a function calls itself up to n-number of times. If a program allows the user to call a function inside the same function recursively, the procedure is called a recursive call of the function.

**Types of Recursions**

Recursion are mainly of two types depending on whether a function calls itself from within itself or more than one function call one another mutually. The first one is called **direct recursion** and another one is called **indirect recursion**. Thus, the two types of recursion are:

1. Direct Recursion
2. Indirect Recursion

**Direct Recursion**

When a function calls itself within the same function repeatedly, it is called the direct recursion.

**Structure of the direct recursion**

fun ()

{

// write some code

fun ();

// some code

}

Direct Recursion can be further categorized into four types:

1. Tail Recursion
2. No Tail/ Head Recursion
3. Nested Recursion
4. Tree Recursion

**Tail Recursion**

A recursive function is called the tail-recursive if the function makes recursive calling itself, and that recursive call is the last statement executes by the function. After that, there is no function or statement is left to call the recursive function.

Example:

#include <stdio.h>

// function definition

void fun1( int num)

{

// if block check the condition

if (num == 0)

return;

else

printf ("\n Number is: %d", num); // print the number

return fun1 (num - 1); // recursive call at the end in the fun() function

}

**Non-Tail / Head Recursion**

A function is called the non-tail or head recursive if a function makes a recursive call itself, the recursive call will be the first statement in the function. It means there should be no statement or operation is called before the recursive calls. Furthermore, the head recursive does not perform any operation at the time of recursive calling. Instead, all operations are done at the return time.

#include <stdio.h>

void head\_fun (int num)

{

if ( num > 0 )

{

// Here the head\_fun() is the first statement to be called

head\_fun (num -1);

printf (" %d", num);

}

}

**Nested Recursion**

In this recursion, a recursive function will pass the parameter as a recursive call. That means “recursion inside recursion”. Let see the example to understand this recursion.

#include <stdio.h>

int fun(int n)

{

if (n > 100)

return n - 10;

// A recursive function passing parameter

// as a recursive call or recursion

// inside the recursion

return fun(fun(n + 11));

}

**Tree Recursion**

To understand Tree Recursion let’s first understand Linear Recursion. If a recursive function calling itself for one time then it’s known as Linear Recursion. Otherwise if a recursive function calling itself for more than one time then it’s known as Tree Recursion.

Example:

**Pseudo Code for linear recursion**

fun(n)

{

// some code

if(n>0)

{

fun(n-1); // Calling itself only once

}

// some code

}

**Program for tree recursion**

#include <stdio.h>

// Recursive function

void fun(int n)

{

if (n > 0) {

printf("%d ", n);

// Calling once

fun(n - 1);

// Calling twice

fun(n - 1);

}

}

**Indirect Recursion**

When a function is mutually called by another function in a circular manner, the function is called an indirect recursion function.

**Structure of the indirect recursion**

In this structure, there are four functions, fun1(), fun2(), fun3() and fun4(). When the fun1() function is executed, it calls the fun2() for its execution. And then, the fun2() function starts its execution calls the fun3() function.

fun1()

{

// write some code

fun2();

}

fun2()

{

// write some code

fun3();

// write some code

}

fun3()

{

// write some code

fun1();

}