# Recursion in C

Recursion is the process which comes into existence when a function calls a copy of itself to work on a smaller problem. Any function which calls itself is called recursive function, and such function calls are called recursive calls. Recursion involves several numbers of recursive calls. However, it is important to impose a termination condition of recursion. Recursion code is shorter than iterative code however it is difficult to understand.

Recursion cannot be applied to all the problem, but it is more useful for the tasks that can be defined in terms of similar subtasks. For Example, recursion may be applied to sorting, searching, and traversal problems.

Generally, iterative solutions are more efficient than recursion since function call is always overhead. Any problem that can be solved recursively, can also be solved iteratively. However, some problems are best suited to be solved by the recursion, for example, tower of Hanoi, Fibonacci series, factorial finding, etc.

**void recursion()**

**{**

**recursion(); /\* function calls itself \*/**

**}**

**int main()**

**{**

**recursion();**

**}**

The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.

**Number Factorial**

The following example calculates the factorial of a given number using a recursive function −

[Live Demo](http://tpcg.io/os8QlL)

#include <stdio.h>

unsigned long long int factorial(unsigned int i)

{

if(i <= 1)

return 1;

return i \* factorial(i - 1);

}

main()

{

int i = 12;

printf("Factorial of %d is %d\n", i, factorial(i));

}

When the above code is compiled and executed, it produces the following result −

Factorial of 12 is 479001600

Fibonacci Series

The following example generates the Fibonacci series for a given number using a recursive function −

[Live Demo](http://tpcg.io/GsweWK)

#include <stdio.h>

int fibonacci(int i)

{

if(i == 0)

return 0;

if(i == 1)

return 1;

return fibonacci(i-1) + fibonacci(i-2);

}

main()

{

int i;

for (i = 0; i < 10; i++)

printf("%d\t\n", fibonacci(i));

}

When the above code is compiled and executed, it produces the following result −

0

1

1

2

3

5

8

13

21

34

## Memory allocation of Recursive method

Each recursive call creates a new copy of that method in the memory. Once some data is returned by the method, the copy is removed from the memory. Since all the variables and other stuff declared inside function get stored in the stack, therefore a separate stack is maintained at each recursive call. Once the value is returned from the corresponding function, the stack gets destroyed. Recursion involves so much complexity in resolving and tracking the values at each recursive call. Therefore we need to maintain the stack and track the values of the variables defined in the stack.