Recursion

Recursion is an approach in which a function calls itself with an argument. Upon reaching a termination condition, the control returns to the calling function.

Properties

A recursive function can go infinite like a loop. To avoid infinite running of recursive function, there are two properties that a recursive function must have −

- Base criteria − There must be at least one base criteria or condition, such that, when this condition is met the function stops calling itself recursively.
- Progressive approach − The recursive calls should progress in such a way that each time a recursive call is made it comes closer to the base criteria.

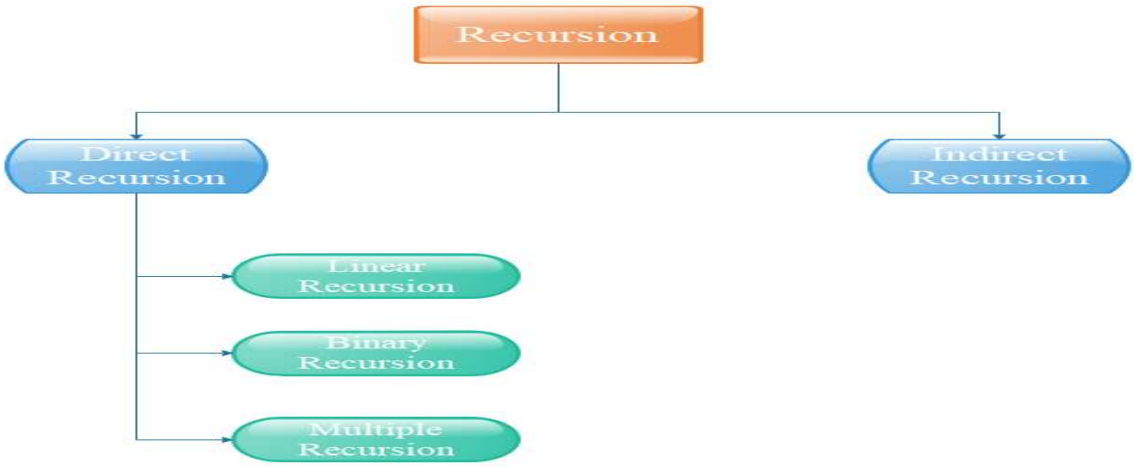
Implementation

Many programming languages implement recursion by means of stacks. Generally, whenever a function (caller) calls another function (callee) or itself as callee, the caller function transfers execution control to the callee. This transfer process may also involve some data to be passed from the caller to the callee.

This implies, the caller function has to suspend its execution temporarily and resume later when the execution control returns from the callee function. Here, the caller function needs to start exactly from the point of execution where it puts itself on hold. It also needs the exact same data values it was working on. For this purpose, an activation record (or stack frame) is created for the caller function.

Recursion in data structures

- Recursion is one of the most powerful tools in a programming language, but one of the most threatening topics-as most of the beginners and not surprising to even experienced students feel.
- When function is called within the same function, it is known as recursion in C. The function which calls the same function, is known as recursive function.
- Recursion is defined as defining anything in terms of itself. Recursion is used to solve problems involving iterations, in reverse order.



Types of Recursion

There are two types of Recursion

- Direct recursion
- Indirect recursion

Direct Recursion

When in the body of a method there is a call to the same method, we say that the method is directly recursive.

There are three types of Direct Recursion

- Linear Recursion
- Binary Recursion
- Multiple Recursion

Linear Recursion

- Linear recursion begins by testing for a set of base cases there should be at least one.

In Linear recursion we follow as under :

- Perform a single recursive call. This recursive step may involve a test that decides which of several possible recursive calls to make, but it should ultimately choose to make just one of these calls each time we perform this step.
- Define each possible recursion call, so that it makes progress towards a base case.

Binary Recursion

- Binary recursion occurs whenever there are two recursive calls for each non base case.

Multiple Recursion

- In multiple recursion we make not just one or two but many recursive calls.

```
//C program for GCD using recursion
#include int
Find_GCD(int, int);

void main()
{
int n1, n2, gcd;
scanf("%d %d",&n1, &n2);
gcd = Find_GCD(n1, &n2);
printf("GCD of %d and %d is %d", n1, n2, gcd);
```

```
}
int Find_GCD(int m, int n)
{
int gcdVal;
if(n>m)
{
gcdVal = Find_GCD(n,m);
}
else if(n==0)
 {
gcdVal = m;
}
else
{
gcdVal = Find_GCD(n, m%n);
}
return(gcdVal);
}
```

Advantages of Recursion

1. A complex problem seems logically convincible when we solve it using recursion.
2. Recursion uses fairly lesser programming constructs to solve the problem than its iterative counterpart.

Disadvantages of Recursion

- It consumes more storage space the recursive calls along with automatic variables are stored on the stack.
- The computer may run out of memory if the recursive calls are not checked.
- It is not more efficient in terms of speed and execution time.
- According to some computer professionals, recursion does not offer any concrete advantage over non-recursive procedures/functions.
- Recursive solution is always logical and it is very difficult to trace.(debug and understand).
- In recursive we must have an if statement somewhere to force the function to return without the recursive call being executed, otherwise the function will never return.
- Recursion takes a lot of stack space, usually not considerable when the program is small and running on a PC.
- Recursion uses more processor time.
- Recursion is not advocated when the problem can be through iteration.
- Recursion may be treated as a software tool to be applied carefully and selectively.

Difference between recursion and iteration

Iteration	Recursion
In iteration,a problem is converted into a train of steps that are finished one at a time, one after another	Recursion is like piling all of those steps on top of each other and then quashing the mall into the solution.
With iteration,each step clearly leads on to the next, like stepping stones across a river	In recursion, each step replicates itself at a smaller scale, so that all of them combined together eventually solve the problem.
Any iterative problem is solved recursively	Not all recursive problem can solved by iteration
It does not use Stack	It uses Stack

Stack Based Implementation

If the base case is not reached or not defined, then the stack overflow problem may arise. Let us take an example to understand this.

```
int fact(int n)
{
// wrong base case (it may cause
// stack overflow).
if (n == 100)
return 1;

else
return n*fact(n-1);
}
```

If fact(10) is called, it will call fact(9), fact(8), fact(7) and so on but the number will never reach 100. So, the base case is not reached. If the memory is exhausted by these functions on the stack, it will cause a stack overflow error.