RECURSION

- Recursion is a powerful programming tool and technique that can be used to solve the problems that can be expressed in terms of similar problems of smaller size.

- for example → The problem of finding the factorial of n can be expressed in terms of a similar problem of smaller size as $n! = n \times (n-1)!$. Recursion provide elegant way of solving such problem.

- In recursive programming, a function call itself. A function that calls itself is known as Recursive Function, and the phenomena is known as "RECURSION".

- Recursion is classified according to following criteria :→

1. Whether the function calls itself directly (i.e direct recursion) or indirectly (i.e indirect recursion).

2. Whether there is a pending operation on return from a Recursive call. If the recursion / recursive call is the last operation of a function, the recursion is known as TAIL RECURSION.

3. Pattern of Recursive calls. According to the pattern of recursive calls, Recursion is classified as :→

   a. LINEAR RECURSION.
   b. BINARY RECURSION.
   c. n-ary RECURSION.

Scanned with OKEN Scanner

The important points about how to develop recursive function are as follows :→

1. Thinking Recursively is the first step to solve a problem using Recursion.

2. Every Recursive Solution consists of two cases :→

   a. Base Case → forms Terminating conditions of the Recursion. There may be more than One Base condition case in a recursive solution.

   W/o Base case, the recursion will never terminate and will be knownas INFINITE RECURSION.

   b. Recursive Case :- In a recursive case, the problem is defined in terms of itself, while reducing the problem size. For example, when fact(n) is expressed as n*fact(n-1), the size of the problem is reduced from n to n-1.

3. Express the solution in the form of Base cases and Recursive Cases. For example, the factorial problem can be expressed as :→

$$fact(n_0) = \begin{cases} 1 & \text{when } n_0 = 1 \\ n_0 * fact(n_0 - 1) & \text{when } n_0 > 1 \end{cases}$$

Relation of the above form is known as Recurrence Relation.

4. Code of the Recurrence Relation.

Scanned with OKEN Scanner

Recursion → Direct Recursion → When function is directly recursive if it calls itself i.e function body conteins an explicit call to itself.

Recursion → Indirect Recursion

→ when a function calls another function calls, which in turn calls another function, eventually resulting in the original function being called again.

Functions involved in INDIRECT RECURSION are k/a MUTUALLY RECURSIVE FUNCTIONS.

DIRECT RECURSION

```
A()        // Direct Recursive
{
------
-------    // statements

A();       // call to itself

--------
--------
}
```

INDIRECT RECURSION

```
A()   // Mutually recursive function A.
{
------   // Statements
B();  // Function A calls function B.
-------
}
B()   // Mutually Recursive func. B.
{
-------   // statements
A();  // function B calls func. A
--------
}
```

Scanned with OKEN Scanner

// Recursion to find the factorial of a number

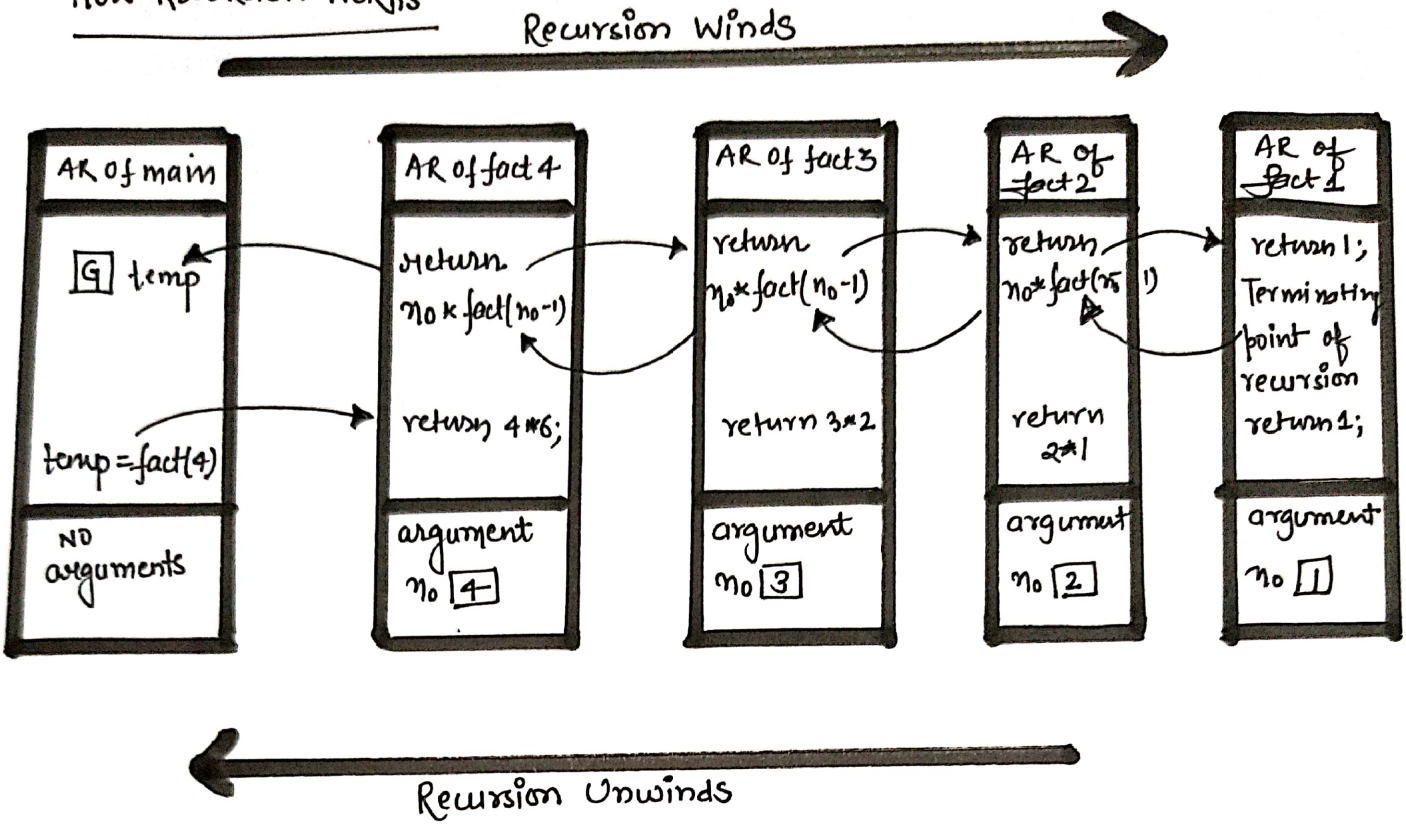
```
#include<stdio.h>
int fact(int);
void main()
{
  int no, factorial;
  printf("Enter the number \t");
  scanf("%d", &no);
  factorial= fact(no);
  printf("Factorial of %d is %d", no, factorial);
}
// Definition of directly recursive function fact
int fact(int no)
{
  if(no==1)
    return 1;
  else
    Return no*fact(no-1);
}
```

o/p

Enter the number 3
factorial of 3 is 6.

Scanned with OKEN Scanner

HOW RECURSION WORKS



G= stands and signifies garbage value of local variable temp.

AR= activation record.

The term activation means execution of a function.
If a function is executing, it is said to be active.

For example, suppose function main calls a function fun1, which in turns calls another function fun2.
While the function fun2 is executing, the functions main, fun1, fun2 are all active.

When the function fun2 completes its execution and returns the program control to fun1, only the function main and fun1 remains active and fun2 becomes inactive.

Activation of each function requires a separate Activation Record.
An Activation Record refers to the chunk of memory, which holds the following :→

(1). DYNAMIC LINK: it point to the activation record of the caller.

(2). SAVED STATE: It refers to the contents of the program counter and registers when the function called. It is used to restore the context of the called function when the program control returns.

(3). PARAMETERS:- They refer to the memory space required by the parameters declared within the header of the function.

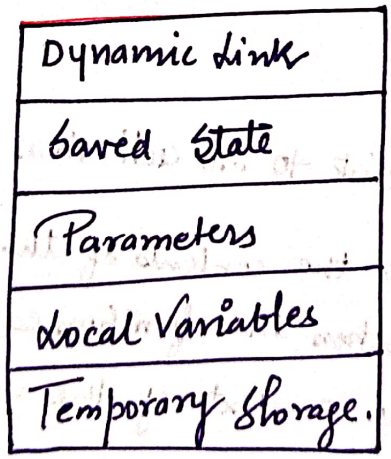
Scanned with OKEN Scanner

(4) LOCAL VARIABLE :- They refer to the memory space required by the automatic local variables.

(5). Temporary storage :- it refers to the storage used for evaluating the expressions.

An activation record is automatically created when a function starts the execution and is automatically destroyed when a function returns the control to its caller.

The activation records for all of the activation functions are stored in the region of memory called the stack.



Scanned with OKEN Scanner