I. La récursivité

Introduction

La récursivité est un concept qui consiste à définir un mode de calcul d'une manière déclarative (par définition). Le principe est basé sur la récurrence. Une structure est dite récursive si elle est constituée en partie ou est décrite par elle-même. La récursivité est un moyen particulièrement puissant dans les définitions mathématiques.

Comme par exemple la valeur de n! est obtenue par la valeur du calcul n*(n-1)! sachant que 0!=1. La récurrence est le fait de définir la **factorielle de n** par la **factorielle de (n-1)**.

En programmation la récursivité est une approche qui remplace les instructions de boucles (*while, for*) par l'auto-appel de fonctions (direct ou indirecte).

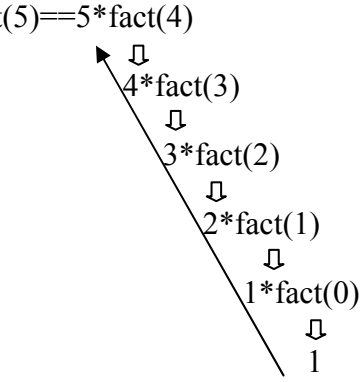
Exemple :

```
#include <stdio.h>

int fact( int n ) /* on suppose   n positive ou nulle*/
{
if (n==0) return 1;
   else return  n*fact(n-1);
}
main()
{
printf(''%d '',fact(5) );
}
```

Principe de calcul :

La valeur qui sera affichée dans printf est la valeur de retour de l'appel fact(5) qui est elle même résultat de la multiplication de 5 par la valeur de retour de l'appel de fact(4) ainsi de suite comme on le schématisera comme suit :



L'exécution des appels récursifs est liée à l'utilisation du **concept de pile et des contextes** de chaque appel.

<u>Le cas particulier :</u>

Ce qu'il faut noter dans cet exemple que pour toutes les valeurs de n il y'a un appel de la même fonction en changeant la valeur de l'argument d'appel sauf pour la valeur 0 où il y a retour de la valeur 1. La valeur 0 associée à n constitue la condition d'arrêt des appels récursifs.

Note : la conception de programmes récursifs est basée sur la procédure ou la fonction. Si une fonction P contient une référence (un appel) explicite à elle-même, on parle alors de récursivité directe. Si une fonction contient une référence explicite à une autre fonction G qui contient une référence (directe ou indirecte) à P, on parle dans ce cas de récursivité indirecte.

Définition : La résolution par récurrence consiste à définir la forme du concept d'un problème par la même forme mais en en plus petit.

Exemple 1 : écrire une fonction en C qui permet de vérifier si un mot est palindrome, sachant qu'un mot est palindrome :

1) s'il est vide ou
2) si le premier caractère est égale au dernier caractère et le mot sans ces deux caractères est palindrome.

Exemple 2 : écrire une fonction en C qui permet de calculer la somme des éléments d'un tableau de taille n :

1) Somme(v,n) = 0 si n=0
2) Somme(v,n) = v[n]+Somme(v,n-1)

On peut écrire une fonction void et utiliser un paramètre de sortie pour l'exemple 2 il faut noter ici que les le calcul est au retour des appels comme suit :

```
void somme (int v [], int n, int *s)
{
   Si (n==0) *s=0;
     Sinon { Somme(v,n-1,s) ;
               *s=*s+v[n-1] ;
           }
}
```

Exercice 1 : La vérification si un tableau est ordonné. Donner une définition récursive. Ecrire une fonction récursive qui vérifie si un tableau d'entiers est trié.

Exercice 2 : Recherche dichotomique d'une valeur val si elle se trouve dans un tableau ordonné.