Récursivité

1 Récursivité

On appelle récursivité toute fonction ou procédure qui s'appelle elle même.

Exemple La factorielle d'un entier naturel n.

$$n! = n * (n-1)!$$

```
Fonction fact(n: entier): entier
Debut
    si n = 0 alors
        retourner 1
    sinon
        retourner n * fact(n - 1)
    Fin si
Fin
```

```
int fact(int n) {
    if(n == 0) return 1;
    else return n * fact(n - 1);
}
```

2 Comment ça marche ?

Dans le cas où n = 4

```
4 * fact(3)
    3 * fact(2)
        2 * fact(1)
            1 * fact(0)
            1 * 1 = 1
        2 * 1 = 2
    3 * 2 = 6
4 * 6 = 24
```

3 Point terminal

Comme dans les boucles, il faut un point d'arrêt où l'on ne fait pas d'appel récursif.

4 La récursivité terminale et non terminale

4.1 Récursivité non terminale

Une fonction récursive non terminale si le résultat de l'appel récursif est utilisé pour réaliser un traitement (en plus du retour d'une valeur).

Exemple de non terminalité Forme récursive non terminale de la factorielle, les calculs se font à la remontée.

```
Fonction factNonTerminale(n: entier): entier
Debut
    si n = 0 alors retourner 1
    sinon retourner n * factNonTerminale(n - 1)
Fin
```

4.2 Récursivité terminale

On dit qu'une fonction est récursive terminale, si tout appel récursif est de la forme *retourner f(. . .)*. Autrement dit, la valeur retournée est directement la valeur obtenue par un appel récursif, sans qu'il y ait aucune opération sur cette valeur.

Exemple Calcul de $n!$

```
Procedure factTerminale(n: entier, resultat: entier)
Debut
    si n = 0 alors retourner resultat
    sinon retourner factTerminale(n - 1, n * resultat)
Fin
```

Exemple Calcul de $a * n!$

```
Fonction f(n: entier, a: entier): entier
Debut
    si n = 0 alors retourner a
    sinon retourner f(n - 1, n * a)
    Fin si
Fin
```

5 La récursivité croisée

Consiste à écrire des fonctions qui s'appellent l'une l'aute.

Exemple

```
Fonction pair(n: entier): booléen
Debut
    si n = 0 alors retourner vrai
    sinon retourner impair(n - 1)
    Fin si
Fin

Fonction impair(n: entier): entier
Debut
    si n = 0 alors retourner faux
    sinon retourner pair(n - 1)
    Fin si
Fin
```