Récursivité

Récursivité simple

Récursivité simple

Une fonction est dite récursive **si elle s'appelle elle-même**

Un algorithme récursif se compose de deux parties :

- Au moins **une condition d'arrêt** des appels récursifs, où les valeurs à déterminer sont immédiatement connues
- **Un appel récursif** : la fonction s'appelle elle-même dans un autre environnement

Récursivité simple

Exemple : voici la fonction « **Factorielle** » notée n !

Elle se définit ainsi :

- Factorielle(n) = 1 x 2 x … x (n – 1) x n

Donc :

- Factorielle(1) = 1
- Factorielle(2) = 1 x 2 = 2
- Factorielle(3) = 1 x 2 x 3 = 6

Et donc :

- Factorielle(n) = Factorielle (n – 1) x n

Récursivité simple

On écrit la fonction :

```
Fonction Factorielle (nb : entier) : entier
Variables : f : entier ;
Début
    Si nb = 1 alors
        f ← 1 ;
    Sinon
        f ← nb x Factorielle (nb – 1) ;
    Fin si
    Retourne(f) ;
Fin
```

Récursivité simple

Appelons la fonction à partir d'un programme principal qui affiche la Factorielle de 3

Algorithme utiliser-Factorielle

Début

    Écrire (Factorielle(3)) ;

Fin

Voici le détail de l'exécution :

Factorielle(3) = 3 x Factorielle (2)

= 3 x (2 x Factorielle (1))

= 3 x (2 x 1)

= 6

Récursivité simple

On peut encore simplifier l'écriture de notre fonction :

```
Fonction Factorielle (nb : entier) : entier
Début
    Si nb = 1 alors
        Retourne(1) ;
    Fin si
    Retourne (nb x Factorielle (nb – 1)) ;
Fin
```

Récursivité simple

Autre exemple : la **suite de Fibonacci**

C'est une suite récurrente dont chaque terme dépend des 2 précédents.

Elle est définie par :

- $U_0 = 0$ et $U_1 = 1$
- $U_n = U_{n-1} + U_{n-2}$ pour tout entier n supérieur ou égal à 2

Donc :

- $U_2 = U_1 + U_0 = 1 + 0 = 1$
- $U_3 = U_2 + U_1 = 1 + 1 = 2$
- $U_4 = U_3 + U_2 = 2 + 1 = 3$
- $U_5 = U_4 + U_3 = 3 + 2 = 5$ …

Récursivité simple

On désire calculer tout élément de la suite de Fibonacci de rang n, pour tout entier n donné

Deux méthodes :

- **Itérative** avec une boucle qui calcule tous les éléments jusqu'au rang n
- **Récursive** en supposant connues toutes les valeurs retournées par $U_{(n-1)}$ et $U_{(n-2)}$

Pour écrire Fibonacci(n), on suppose donc que les valeurs retournées par Fibonacci(n-1) et Fibonacci(n-2) sont connues

Récursivité simple

L'algorithme avec la méthode récursive s'écrit ainsi :

```
Fonction Fibonacci (n : entier) : entier
Début
    Si (n = 0) alors
        Retourne(0) ;
    Sinon Si (n = 1) alors
            Retourne(1) ;
    Sinon
        Retourne (Fibonacci (n – 1) + Fibonacci (n – 2)) ;
Fin
```

Récursivité terminale

Une fonction est dite **récursive terminale** si elle retourne sans autre calcul la valeur obtenue par son appel récursif

La dernière ligne d'une telle fonction sera :

Retourne (fonction (paramètres)) ;

Revenons à la fonction Factorielle qui nous proposait :

Retourne (nb x Factorielle (nb – 1)) ;

Ce n'est pas de la récursivité terminale car pour le retour de valeur, on multiplie la valeur de la fonction par nb

Récursivité simple

La version récursivité terminale est donc :

```
Fonction Factorielle (nb : entier, résultat : entier) : entier
Début
    Si nb = 1 alors
        Retourne résultat ;
    Fin si
    Retourne (Factorielle (nb – 1, nb x résultat)) ;
Fin
```

Récursivité simple

Cette fonction est appelée en mettant initialement le paramètre *résultat* à 1 par :

```
Fonction Factorielle (nb : entier, 1) : entier
Début
    Retourne (Factorielle (nb, 1)) ;
Fin
```

Le paramètre *résultat* est calculé uniquement au fur et à mesure de l'appel récursif :Factorielle (3, 1) retourne la valeur Factorielle (2, 3) qui retourne la valeur Factorielle (1, 6) qui retourne 6