Cours 1: Récursivité

1 Introduction

Définition: La récursivité est une méthode de description d'algorithmes qui permet à une procédure (ou une fonction) de s'appeler elle-même.

L'expression d'algorithmes sous forme récursive permet généralement l'écriture des fonctions sous une forme concise et plus simple à comprendre bien qu'elle soit moins naturelle à concevoir. Lorsque le problème traité peut se décomposer en une succession de sous-problèmes identiques, la récursivité est généralement bien indiquée.

2 Itératif vs récursif

Tous les algorithmes récursifs peuvent s'écrire sous forme itérative. Voici un exemple de deux écritures de la fonction calculant la factorielle d'un entier N :

Itérative

```
int Factorielle (int N)
{
    int i,Res=1;
    for (i=2;i<=N;i++)
        Res*=i;
    return Res;
}
```

Récursive

```
int Factorielle (int N)
{
    if (N<=1)
        return 1;
    else
        return N*Factorielle(N-1);
}
```

La forme récursive est généralement plus simple à comprendre et plus élégante, elle peut être séduisante dans sa conception intellectuelle. Mais les appels récursifs occasionnent la sauvegarde du contexte (les valeurs des variables) avant chaque appel et sa restitution au retour de l'appel, ce qui peut diminuer l'efficacité du programme.

3 Profondeur

La profondeur correspond au nombre d'appel(s) de la fonction. Une fonction traditionnelle (non récursive) aura une profondeur de 1. Une fonction ayant une profondeur de 5 signifie qu'elle s'est appelée elle-même 4 fois et a été appelée de l'extérieur une fois (que l'on appellera l'appel principal). La profondeur n'est généralement pas une propriété intrinsèque à la fonction mais dépend des paramètres qui lui sont passés.

4 Variables

Dans les fonctions récurrentes, les variables peuvent être classées en 3 catégories :

- Les variables passée en paramètres: ces variables sont liées à un appel de la fonction. La mémoire est allouée à chaque nouvel appel de la fonction et est respectivement libérée à chaque terminaison de la fonction. Ces variables sont initialisées lors de l'appel de la fonction.
- Les variables locales à la fonction : elles ont les mêmes propriétés que les variables passées en paramètres sauf qu'elles ne sont pas initialisées à l'appel de la fonction.
- Les variables statiques: ces variables sont communes à tous les appel de la fonctions. La

mémoire est allouée lors du premier appel de la fonction est n'est libérée qu'à la fin du programme. La valeur initiale n'est affectée qu'une fois, lors du premier appel de la fonction.

5 Limite de profondeur

Afin d'éviter des profondeurs infinies, une fonction récursive doit nécessairement comporter un test d'arrêt qui met un terme à la récursivité. Lorsque le test d'arrêt est vrai, on exécute la récursion terminale qui est l'action réalisée lors du dernier appel de la fonction. Sans cette condition d'arrêt, les appels vont se perpétrer jusqu'à atteindre la limite du nombre d'appel ou jusqu'à saturation de la mémoire. Voici la structure préconisée pour une fonction récursive:

```
… Fct (…) {
        if (Test) {
                …                // Récursion terminale (pas d'appel récursif)
        }
        else {
                …
                Fct (…);         // Appel récursif de la fonction
                …
        }
}
```

6 Récursivité croisée

La récursivité croisée traduit le fait que deux fonctions s'appellent mutuellement. Une fonction `f1()` effectue un calcul en appelant une fonction `f2()`, qui elle-même appelle la fonction `f1()`. Là encore, le problème à surveiller est le risque que les conditions d'arrêt ne soient jamais vraie et que la limite de profondeur soit atteinte.