Récursivité

La récursivité est une technique de programmation où une fonction s'appelle elle-même pour résoudre un problème en le divisant en sous-problèmes plus simples, jusqu'à atteindre une condition d'arrêt. C'est un concept central dans de nombreux algorithmes, notamment dans les structures de données comme les arbres ou pour résoudre des problèmes complexes de manière élégante.

Elle se définit comme ceci:

> *Une fonction F peut résoudre un problème P de manière récursive si*
>
> - *Le problème P est trivial. Dans ce cas la fonction retourne directement une valeur*
> - *Le problème P peut être divisé en problèmes simples (P1, P2, P3,...) qui peuvent tous être résolus par la fonction F. Dans ce cas, la fonction retourne le résultat de la résolution de tous les sous-problèmes par elle-même*

Il est essentiel que la décomposition en problèmes simples débouche sur un problème trivial, faute de quoi on part dans une boucle infinie.

Exemple classique: le calcul de la factorielle d'un nombre n :

Rappel mathématique:

> *`6! = 6 * 5 * 4 * 3 * 2 * 1 = 720`*

On peut donc écrire la fonction factorielle ainsi en C#:

```
int Factorielle(int x)
{
    int res = 1;
    for (int i = x; i > 1; i--) res *= i;
    return res;
}
```

En observant le rappel ci-dessus, on voit que l'on peut écrire:

> *`6! = 6 * 5!` (puisque `5! = 5 * 4 * 3 * 2 * 1`)*

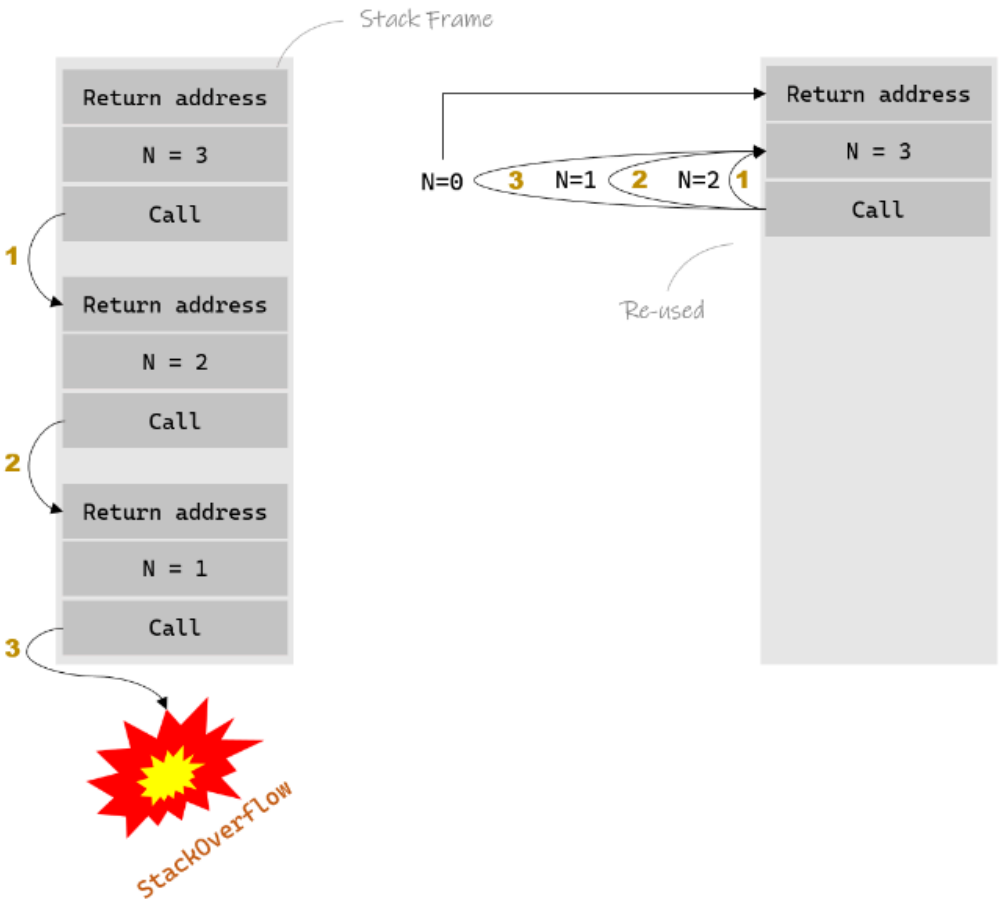
Du coup, comme le calcul de `1!` est trivial, on peut écrire la fonction ainsi:

```
int Factorielle(int x)
{
    if (x==1) return 1;
    return x * Factorielle(x - 1);
}
```

> *On remarque que l'expression récursive de la fonction n'utilise, dans cet exemple, que des variables immutables...*

Performances

Historiquement, les fonctions récursives étaient moins performantes à cause du fait qu'un appel de fonction impliquait d'allouer des éléments sur la pile, ce qui n'est plus le cas avec la technologie `Tail Call Optimisation` alias `TCO` :



Un test non exhaustif et sujet à des optimisations `JIT` avec .NET 6 donne les résultats suivants sur des factiorielles entre 200 et 210:

```
Factorielle séquentielle: 343 ticks
Factorielle récursive: 501 ticks
```

Cas d'application

La récursivité est particulièrement utile pour les problèmes comme les tris (ex. : quicksort), les arbres binaires, ou les algorithmes de recherche.