IFT 1015 - Récursion

Professeur:

Stefan Monnier

B. Kégl, S. Roy, F. Duranleau, S. Monnier

Département d'informatique et de recherche opérationnelle

Université de Montréal

hiver 2006

Au programme

[Niño: 17]

- Récursion
-
-
-
-
-
-
-

Une fonction qui s'appelle elle-même

Le code d'une fonction peut-être arbitraire

$\Rightarrow$ en particulier, une fonction peut s'appelle elle-même.

```
public static void fonction (int a)
{
  return a * fonction(a - 1);
}
```

Que se passe-t-il si on fait l'appel suivant?

```
fonction(4)
```

Une fonction qui s'appelle elle-même

Le code d'une fonction peut-être arbitraire

$\Rightarrow$ en particulier, une fonction peut s'appelle elle-même.

```
public static void fonction (int a)
{
  return a * fonction(a - 1);
}
```

Que se passe-t-il si on fait l'appel suivant?

```
fonction(4)
```

$\longrightarrow$ ça va planter! `fonction(4)` appelle `fonction(3)`, qui appelle `fonction(2)` et ainsi de suite, sans s'arrêter.

Remède

- Similairement aux boucles, il faut ajouter une condition d'arrêt pour éviter les appels récursifs infinis.
- Ces conditions d'arrêt sont reliées au cas de base du problème à résoudre par récurrence.

Ex.:

- Factoriel: $f_0 = 1$, $f_1 = 1$ et $f_n = n \cdot f_{n-1}$
- Fibonacci: $f_0 = 0$, $f_1 = 1$ et $f_n = f_{n-1} + f_{n-2}$

Trace d'exécution

Que se passe-t-il lors d'un appel d'une fonction?

$\longrightarrow$ La machine garde en mémoire une adresse de retour, i.e. où poursuivre l'exécution du programme au retour d'une fonction.

$\longrightarrow$ Un espace mémoire est réservé pour les paramètres de la fonction, et ses variables locales.

Toute cette mémoire s'appelle la pile d'exécution.