Algorithmique

Récursivité

JDD Nkapkop, PhD

Mél : jdd.nkapkop@gmail.com

Université de Douala

Récursivité et Récurrence

Deux notions très proche :

- mathématiques : récurrence
- informatique : récursivité

De nombreuses définitions mathématiques sont récursives :

Définition (Peano)

- *0 est un entier naturel.*
- *Tout entier n a un successeur unique Sn $(= n + 1)$;*
- *Tout entier sauf 0 est le successeur d'un unique entier ;*
- *Pour tout énoncé $P(n)$ si $P(0)$ est vrai et si pour tout n, $P(n)$ implique $P(Sn)$ alors l'énoncé $\forall n : P(n)$ est vrai.*

Définition

Moyen simple et élégant de résoudre certain problème.

Définition

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

```
Algorithme Fact
Entrée : un entier positif N
Sortie : factorielle de N

si N = 0 retourner 1
sinon retourner N x Fact(N-1)
```

Exemple dans un vrai langage de programmation

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

Exemple dans un vrai langage de programmation

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

Ça marche ! ! !

Comment ça marche ?

```
Appel à fact(4)
.   4*fact(3) = ?
.   Appel à fact(3)
.   .   3*fact(2) = ?
.   .   Appel à fact(2)
.   .   .   2*fact(1) = ?
.   .   .   Appel à fact(1)
.   .   .   .   1*fact(0) = ?
.   .   .   .   Appel à fact(0)
.   .   .   .   Retour de la valeur 1
.   .   .   .   1*1
.   .   .   Retour de la valeur 1
.   .   .   2*1
.   .   Retour de la valeur 2
.   .   3*2
.   Retour de la valeur 6
.   4*6
Retour de la valeur 24
```

Notion de pile d'exécution

Définition (Pile d'exécution)

*La **Pile d'exécution** du programme en cours est un emplacement mémoire destiner à mémoriser les paramètres, les variables locales ainsi que les adresses de retour des fonctions en cours d'exécution.*

Elle fonctionne selon le principe LIFO (Last-In-First-Out) : dernier entré premier sorti.

Attention ! La pile à une taille fixée, une mauvaise utilisation de la récursivité peut entraîner un débordement de pile.

Notion de pile d'exécution

Définition (Pile d'exécution)

*La **Pile d'exécution** du programme en cours est un emplacement mémoire destiner à mémoriser les paramètres, les variables locales ainsi que les adresses de retour des fonctions en cours d'exécution.*

Elle fonctionne selon le principe LIFO (Last-In-First-Out) : dernier entré premier sorti.

Attention ! La pile à une taille fixée, une mauvaise utilisation de la récursivité peut entraîner un débordement de pile.

Notion de pile d'exécution

Définition (Pile d'exécution)

*La **Pile d'exécution** du programme en cours est un emplacement mémoire destiner à mémoriser les paramètres, les variables locales ainsi que les adresses de retour des fonctions en cours d'exécution.*

Elle fonctionne selon le principe LIFO (Last-In-First-Out) : dernier entré premier sorti.

Attention ! La pile à une taille fixée, une mauvaise utilisation de la récursivité peut entraîner un débordement de pile.

Point terminal

Retenir

Comme dans le cas d'une boucle, il faut un cas d'arrêt où l'on ne fait pas d'appel récursif.

```
procédure récursive(paramètres):
    si TEST_D'ARRET:
        instructions du point d'arrêt
    sinon
        instructions
        récursive(paramètres changés);  // appel récursif
        instructions
```

La récursivité terminale

Définition

On dit qu'un fonction est récursive terminale, si tout appel récursif est de la forme `return f(...)`

Autrement dit, la valeur retournée est directement la valeur obtenue par un appel récursif, sans qu'il n'y ait aucune opération sur cette valeur. Il n'y a ainsi rien à retenir sur la pile.

```
Entrée : Entiers positifs n, a
Sortie : a*n!

    si n == 0 retourner a
    sinon     retourner Algo(n-1,n*a)
```

La récursivité terminale

Définition

On dit qu'un fonction est récursive terminale, si tout appel récursif est de la forme `return f(...)`

Autrement dit, la valeur retournée est directement la valeur obtenue par un appel récursif, sans qu'il n'y ait aucune opération sur cette valeur. Il n'y a ainsi rien à retenir sur la pile.

```
Entrée : Entiers positifs n, a
Sortie : a*n!

    si n == 0 retourner a
    sinon     retourner Algo(n-1,n*a)
```

La récursivité terminale (2)

Idée : Il n'y a rien à retenir sur la pile.

Retenir

Quand une fonction est récursive terminale, on peut transformer l'appelle récursif en une boucle, sans utilisation de mémoire supplémentaire.

Attention ! cette optimisation n'est pas supportée par tous les compilateurs et est optionnelle (ex : -O3 avec gcc).

```
si n == 0 retourner a
sinon     retourner Algo(n-1,n*a)
```

Devient :

```
Tant que n <> 0:
    a <- n*a; n <- n-1
retourner a
```

La récursivité terminale (2)

Idée : Il n'y a rien à retenir sur la pile.

Retenir

Quand une fonction est récursive terminale, on peut transformer l'appelle récursif en une boucle, sans utilisation de mémoire supplémentaire.

Attention ! cette optimisation n'est pas supportée par tous les compilateurs et est optionnelle (ex : `-O3` avec `gcc`).

```
si n == 0 retourner a
sinon     retourner Algo(n-1,n*a)
```

Devient :

```
Tant que n <> 0:
    a <- n*a; n <- n-1
retourner a
```