

## Correction TD Algorithmique

### 5.5 Récursivité

#### Définition 26 (Récursivité)

On appelle récursivité le fait pour un sous-programme (méthode, fonction ou procédure selon le contexte) de s'appeler au moins une fois. La vision récursive s'oppose bien souvent à la vision itérative.

#### Exemple 8 (Factorielle)

Le calcul d'une factorielle peut s'exprimer de deux manières :

$$n! = n \times n - 1 \times \dots \times 1$$

$$n! = n \times (n - 1)!$$

La première version permet l'implantation d'une fonction itérative (voir exercice 31).

La deuxième permet une implantation récursive de la fonction factorielle.

**Fonction** factorielle (nombre : entier) : entier

**var** res : entier

**Si** (nombre = 1 ou nombre = 0) **Alors**

        res ← 1

**Sinon**

        res ← factorielle(nombre-1)

**Fin Si**

    Renvoie(res)

**Fin Fonction**

Une des difficultés d'un algorithme récursif est d'en garantir la terminaison.

#### Exercice 78 (Fibonacci)

Écrire une fonction (récursive) permettant de calculer un élément de la suite de Fibonacci donnée par la définition par récurrence suivante :

$$U_0 = 1$$

$$U_1 = 1$$

$$U_n = U_{n-1} + U_{n-2}$$

**Fonction** fibonacci (nombre : entier) : entier

**var** res : entier

**Si** (nombre = 1 ou nombre = 0) **Alors**

        res ← 1

**Sinon**

        res ← fibonacci(nombre-1) + fibonacci(nombre-2)

**Fin Si**

    Renvoie(res)

**Fin Fonction**

#### Exercice 79 (Recherche dichotomique dans un tableau trié)

Réécrire une fonction de recherche dichotomique récursive.

**Fonction** RechDicho(tab : *Tableau*<E>, inf :entier, sup :entier, elt :type)

**var** resultat : entier

**var** milieu : entier

**Si** (inf > sup) **Alors**

```
    resultat ← -1
  sinon
    milieu ← (inf + sup)/2
    si (tab[milieu]=elt) alors
      resultat ← milieu
    sinon
      si tab[milieu] > elt alors
        resultat ← RechDicho(tab, inf, milieu-1, elt)
      sinon
        resultat ← RechDicho(tab, milieu + 1, sup, elt)
      fin si
    fin si
  fin si
Fin Fonction
```

### Exercice 80 (Aspirateur de site)

On suppose l'existence d'un type URL dotée de deux méthodes :

- analyse() : tableau [URL] qui renvoie dans un tableau tous les liens disponibles dans la page. Cela correspond aux valeurs des attributs suivants : <a href=..., <img src=..., <link rel="stylesheet" href=..., ...
- recupere() : vide qui enregistre l'URL

**Fonction** aspirateur(page : URL, profondeur : entier) : vide

```
  Page.recupere()
  Si (profondeur > 1) Alors
    Urls ← page.analyse()
    Pour i de 0 à urls.longueur() -1 faire
      Aspirateur (urls[i], profondeur - 1)
    Fin pour
  Fin si
Fin Fonction
```