ALGORITMIQUE

3. RECURSIVITE

3.1. Définition

Un algorithme récursif est un algorithme qui fait appel à lui-même au sein de son corps.

Par contre, un algorithme itératif est un algorithme qui utilise des itérations.

Remarque

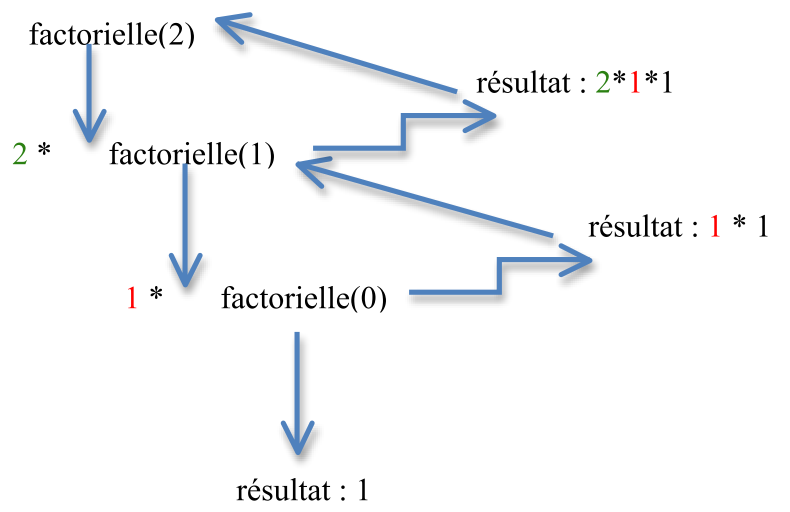
Tout algorithme itératif peut s'écrire en récursif et inversement.

Exemple 1

Considérons l'algorithme qui détermine la factorielle d'un entier naturel.

On sait que, 0 ! =1 et que n ! = n * (n-1) ! ∀ n ≥ 1.

Autrement dit, factorielle(0) = 1 et factorielle(n) = n * factorielle(n - 1) ∀ n ≥ 1.



L'algorithme récursif correspondant est le suivant.

Fonction factorielle (d n : entier naturel) : entier naturel;

{Cette fonction utilise en entrée n qui est un entier positif, et calcule la factorielle de n}

Début

Si (n=0) alors

factorielle:= 1 {On commence par la ou les conditions d'arrêt}

Sinon

factorielle:= n* factorielle (n-l) {On effectue l'appel récursif}

Fin ;

Exemple 2

Considérons la suite de Fibonacci définit comme suit.

$$Fib(0) = 0$$

$$Fib(1) = 1$$

$$Fib(n) = Fib(n - 1) + Fib(n - 2) \ \forall \ n \geq 2.$$

L'algorithme récursif correspondant est le suivant.

Fonction Fib (d n : N) : N;

{Cette fonction utilise en entrée n qui est un entier positif, et calcule la factorielle de n}

Début

Si (n ≤ 1) alors

Fib := n {On commence par les conditions d'arrêt}

Sinon

Fib := Fib(n - 1) + Fib(n - 2) {On effectue l'appel récursif}

Fin ;

3.2. Récursivité indirecte

Quand un algorithme fait appel à lui-même au sein de son corps, on appelle cela récursivité directe.

Par contre, quand un algorithme récursif A fait appel à un algorithme récursif B, et que l'algorithme B fait appel à l'algorithme A, on appelle cela récursivité indirecte. Autrement dit, quand les appels sont sous forme d'un circuit, alors c'est la récursivité indirecte.

Exemple

Considérons l'algorithme qui suit.

Un entier naturel n est dit pair si n-1 est impair, et il est dit impair si n-1 est pair. Les conditions d'arrêt sont données par les valeurs n=0 qui est paire, et n=1, qui est impaire.

Les algorithmes récursifs correspondants sont les suivants.

```
Fonction pair (d n : N) : booléen;
{…}
Début
    Si (n = 0) alors
            pair := vrai
    Sinon Si (n = 1) alors
                pair := faux
            Sinon
            pair := impair(n - 1)    {On effectue l'appel récursif indirect}
Fin ;

Fonction impair (d n : N) : booléen;
{…}
Début
    Si (n = 0) alors
            impair := faux
    Sinon Si (n = 1) alors
                impair := vrai
            Sinon
            impair := pair(n - 1)    {On effectue l'appel récursif indirect}
Fin ;
```

3.3. Récursivité directe et indirecte

Exemple

Considérons les suites suivantes.

$$U0 = 1 \; ; \; Un = 2*Un\text{-}1 + 3*Vn\text{-}1$$

$$V0 = 2 \; ; \; Vn = 2*Vn\text{-}1 + Un\text{-}1$$

Les algorithmes récursifs correspondants sont les suivants.

```
Fonction U (d n : N) : N;
{…}
Début
    Si (n = 0) alors
        U := 1
    Sinon
        U := 2 * U(n - 1) + 3 * V(n - 1)      {On effectue l'appel récursif direct et
                                                indirect}
Fin ;

Fonction V (d n : N) : N;
{…}
Début
    Si (n = 0) alors
        V := 2
    Sinon
        U := 2 * V(n - 1) + U(n - 1)    {On effectue l'appel récursif direct et
                                          indirect}
Fin ;
```

3.4. Récursivité terminale

Une fonction est dite récursive terminale s'il n'y a aucune instruction à exécuter après chacun des appels à elle-même qu'elle contient.

Exemple

Considérons l'algorithme Recter suivant.

```
Fonction Recter (d T : tableau [1 … n] d'entiers ; d i, n : entier) : entier;
{…}
Début
    Si i ≤ n alors
        TRAITER(T[i])
        Recter :=Recter(T, i + 1, n)
    Fin si
Fin ;
```

La fonction Recter est récursive terminale, car il n'existe aucune instruction à exécuter après l'appel récursif Recter(T, i + 1, n).

Règle de transformation

Pour les fonctions récursives terminales, il existe une règle générale permettant de les transformer en fonctions itératives. Cette règle est la suivante.

Soit Frec la fonction récursive terminale qui suit.

Fonction Frec(U) : type ;

Début

    Si C alors

        D

        Frec(τ(U))

    Sinon

        T

Fin ;

Où,

U est la liste des paramètres (arguments) ;

C est une condition qui porte sur $U' \subset U$;

D est le traitement de base qui dépend de U' ;

τ(U) est la transformation des paramètres ;

T est le traitement de terminaison qui dépend de U.

La fonction itérative, Fiter, correspondant à Frec est la suivante.

Fonction Fiter(U **-** **U'**) : type ;

Var U'

Début

    U':=I

    Tant que C faire

        D;

        U' := τ (U')

    Fin tant que

    T

Fin ;

Où I sont les valeurs données à U' lors du premier appel récursif.

Exemple

La fonction itérative correspondant à la fonction récursive terminale Recter est la suivante.

```
Fonction Reciter (d T : tableau [1 … n] d'entiers ; d n : entier) : entier;
{…}
Var i :N
i :=1
Début
    Tant que i ≤ n faire
        TRAITER(T[i])
        i:=i+1
    Fin Tant que
Fin ;
```