

Méthode récursive et méthode itérative

1) Méthode récursive

On dit qu'une procédure est écrite de manière **récursive** lorsqu'elle fait appel à elle-même lors du calcul de ses valeurs. Par exemple, pour écrire récursivement la fonction puissance, on se contente d'indiquer que $x^n = x * x^{(n-1)}$, et on traite le cas initial $x^0 = 1$.

```
f:=proc(x,n) local resultat ;  
    if n=0 then resultat :=1 # traite le cas initial  
    else resultat := x*f(x,n-1) end if  
  
    resultat  
end proc;
```

L'appel de f pour évaluer f(10) entraîne l'appel de f pour f(9), etc. L'écriture est correcte si la suite d'appels aboutit à un « cas initial », où aucun appel récursif n'est effectué. Par exemple,
 $f(10) \gg f(9) \gg f(8) \gg f(7) \gg f(6) \gg \dots \gg f(0) := 1$

On parle de récursivité binaire (ou double) lorsque la procédure fait deux appels récursifs (cf. Fibonacci)
Exemple : la fonction d'Ackermann telle que $A(0,m)=m+1$ (cas de base); $A(n,0)=A(n-1,1)$; $A(n,m)=A(n-1,A(n,m-1))$;

```
ackermann := proc(n, m)  
    if n = 0 then m + 1  
    elif m = 0 then ackermann(n - 1, 1)  
    else ackermann(n - 1, ackermann(n, m - 1))  
    end if  
end proc ;
```

2) Méthode itérative

L'écriture **itérative** consiste à faire appel à une boucle :

```
g:=proc(x,n) local p;
```

```
    p:=1; #initialiser une variable avec  $x^0$   
  
    to n do p:=p*x end do; #on multiplie n fois par x  
  
    p  
end proc;
```

Invariant de boucle : tout algorithme itératif possède un invariant de boucle, c'est-à-dire une propriété qui reste vraie pendant toute l'exécution de l'algorithme, c'est-à-dire avant la boucle, pendant chaque itération et après la boucle. Dans la procédure g, l'invariant de boucle est $p = x^k$ (où k est l'indice de l'itération).

NB : Trouver l'invariant de boucle est un élément important pour *prouver* l'algorithme (hors programme).