# Chapitre 1 : Récursivité

# Récursivité

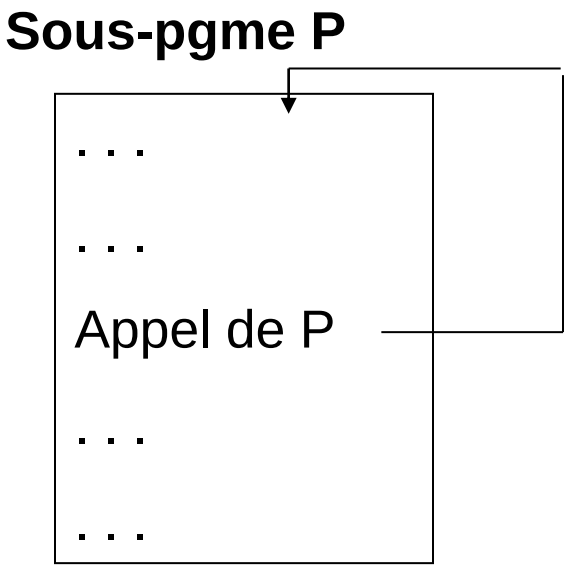
Déf 1: Un mécanisme(fct, poced…) est récursif s'il se contient lui-même en partie.

Exple: Placez vous entre 2 miroirs opposés

Déf 2: En pgtion, la récursivité est le fait qu'un sous-pgme s'appelle lui-même.

La récursivité permet de résoudre certains problèmes de manière plus rapide que leur résolution itérative. Cette dernière consommerait plus de temps et de structures de données intermédiaires.

```
Sous-pgme P;
Begin
   Instr;
   Appel de P;
   Instr;
End;
```



Exemple :

```
Function fact ( n : Integer) : Integer;
Begin
    IF n=1
        then fact :=1
        else fact :=n*fact(n-1);
End;
```

**Sans récursivité**, la fonction précédente s'écrit :

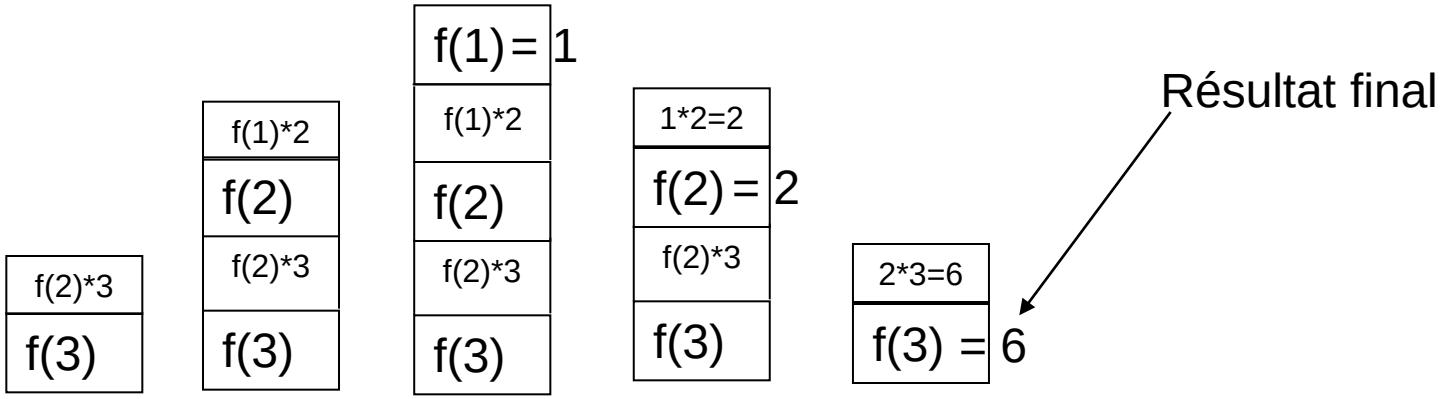
```
Function fact (n: integer) : integer;
Var i, f :integer;
Begin          f:=1;
               For i:=1 to n do        f:=f*i;
               fact:=f;
End;
```

# Déroulement d'un algo récursif

Simuler le fonctionnement du compilateur en utilisant une pile.

Exemple 1: Function fact(n:integer):Integer;

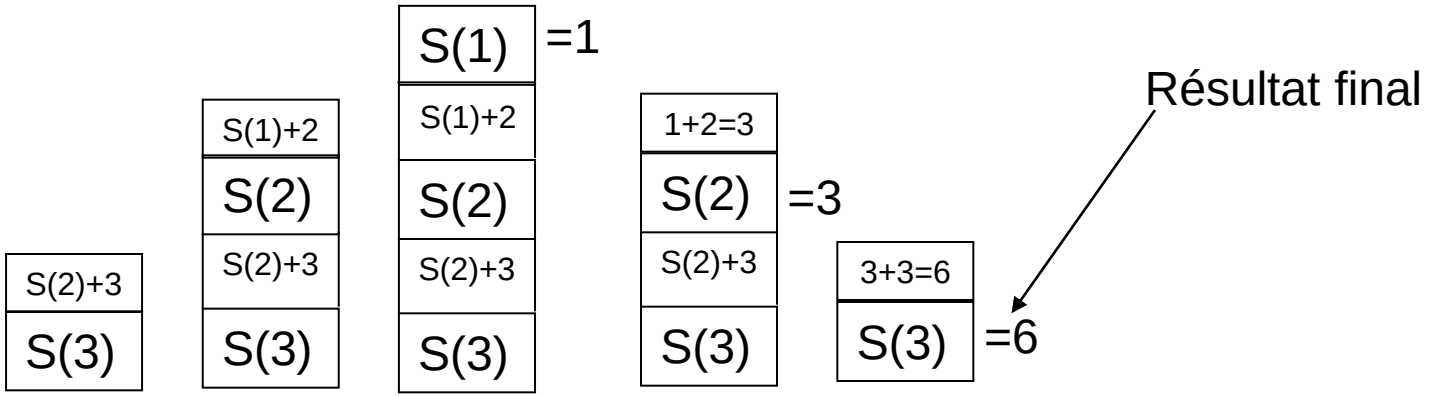
```
Begin
        IF n=1 then fact:=1
                else fact:=fact(n-1)*n;
End;
```



# Déroulement d'un algo récursif

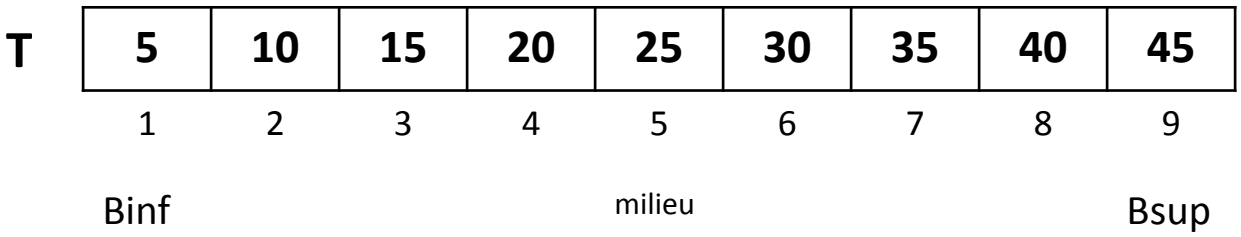
Exemple 2: Function som(n:integer):Integer;

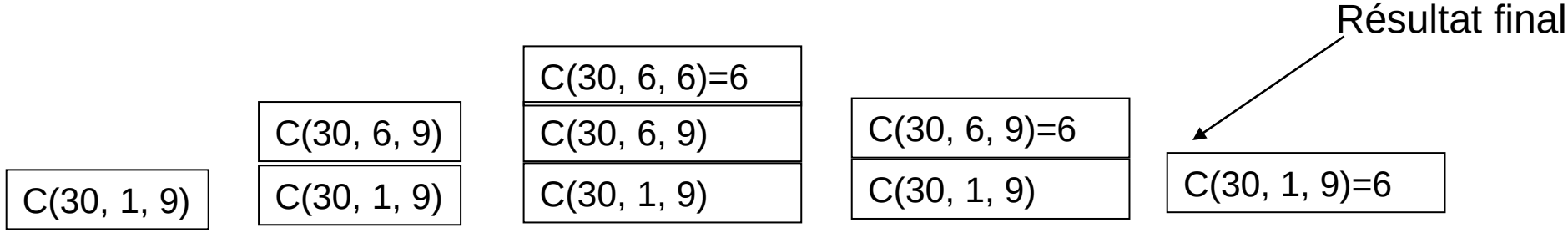
```
Begin
        IF n=1 then som:=1
                else som:=som(n-1)+n
End;
```



Exemple 3: **Recherche dichotomique d'un élément dans un tableau ordonné T**

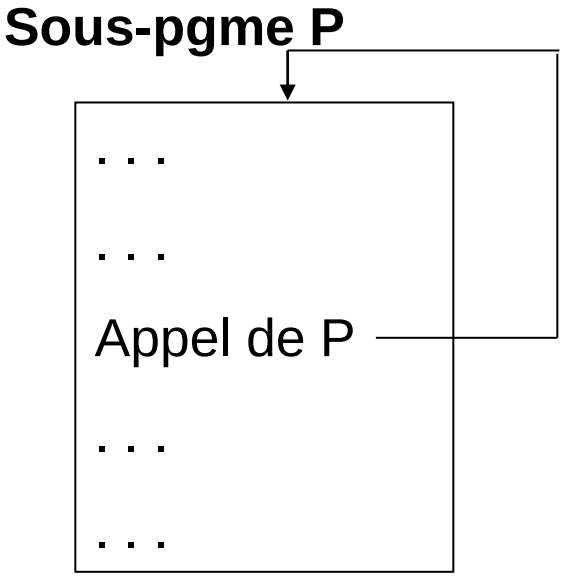
```
Function chercher(x:integer; Binf, Bsup:integer):Integer;
Var milieu:integer;
Begin       IF Binf>Bsup then chercher:=-1
                    else        Begin
                                milieu:=(Binf+Bsup) div 2;
                                if x=T[milieu] then chercher:=milieu
                                                else if x<T[milieu]
                                                        then  chercher:=chercher(x, Binf, milieu-1)
                                                        else chercher:=chercher(x, milieu+1, Bsup );

                                End;
    End;
```



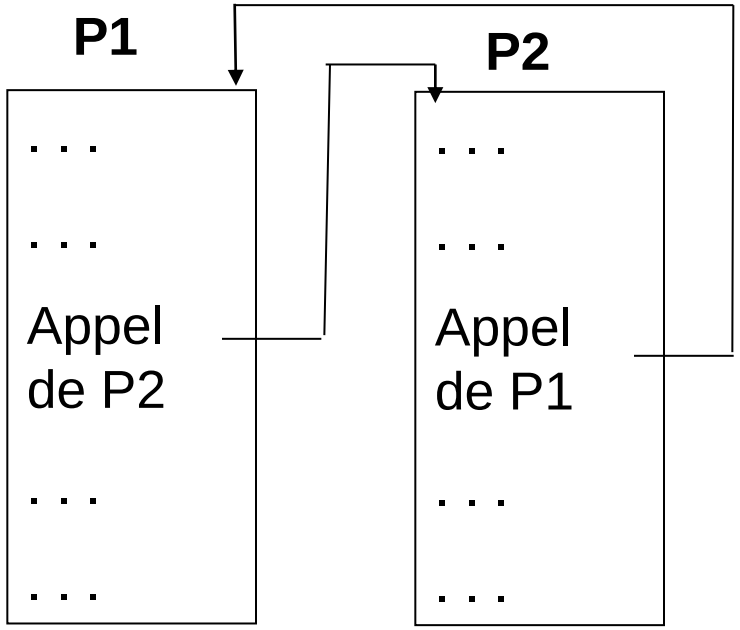
# 2 types de Récursivité

Récursivité simple: Un sous-pgme s'appelle lui-même directement.




Récursivité croisée:

2 sous-pgmes s'appellent mutuellement.



## Avantages

- Economie de l'écriture du code.
- Eviter la gestion des boucles.

## Inconvénients

- Grand espace mémoire à l'exécution
- Possibilité d'une infinité d'appels

# Construction d'un algorithme récursif

1- Chercher à décomposer le pbme en +eurs sous-pbmes de même type, mais de tailles inférieures. Chaque sous-pbme définit une action de l'algorithme.

   Exemple: Fact(n)=n*Fact(n-1)  Si n>=1

2- Déterminer l'action qui représente le cas trivial et donner sa solution.  Par exemple :

   Si n=0 alors Fact(n)=1.

3- En supposant connaître l'action pour le cas d'ordre (n-1), peut-on prévoir le cas suivant, càd d'ordre n?

   Si on connaît Fact(N-1), Alors :

   Fact(N)=N*Fact(N-1)

# Série de TD n°1 : Récursivité

## Exercice 01

Ecrire le programme récursif qui calcule la somme des **N** premiers entiers naturels.

## Exercice 02

Soit A un tableau d'entiers. Ecrire des programmes récursifs pour calculer :

**a)** le maximum de A

**b)** le minimum de A,

**c)** la somme des éléments de A,

**d)** le produit des éléments de A,

**e)** la moyenne des éléments de A.

## Exercice 03

Soit un tableau ordonné d'entiers naturels ; Ecrire un programme récursif qui recherche par **dichotomie** un élément dans ce tableau.