

La récursivité consiste à ce qu'une fonction s'appelle elle-même pendant sa propre exécution.

Exemple avec un peu de math :

Soit:

$$S(n) = 0 + 1 + 2 + 3 + ... + n$$

la somme des premiers entiers naturels

Cette fonction est définie ci-dessus d'une façon non récursive.

On peut aussi l'écrire de la manière suivante :

$$S(n) = (0 + 1 + 2 + 3 + + (n-1)) + n$$

 $S(n) = S(n-1) + n$
 $et S(0) = 0$

Si
$$n = 0$$
 alors $S(n) = 0$
Si $n > 0$ alors $S(n) = S(n-1) + n$

Cette deuxième définition est dite **récursive**, car, dans la deuxième partie de la définition, la fonction S figure à gauche et à droite de l'égalité. C'est en ce sens que l'on dit que la fonction s'invoque ellemême dans sa définition.

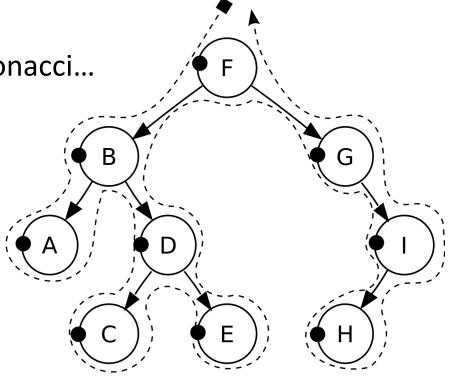
```
#include <stdio.h>
                                             Récursivité
int sommeEntiers(int);
int main() {
   printf("%d", sommeEntiers(3));
int sommeEntiers(int x) {
                           C'est la condition de
   if (x == 0) {
                             fin de récursivité
       return 0;
   else {
       return sommeEntiers(x - 1) + x;
                                    La fonction s'appelle
```

```
1^{er} appel (x = 3 > 0 : appel récursif)
   2^{\text{ème}} appel (x = 2 > 0 : appel récursif)
      3^{\text{ème}} appel (x = 1 > 0 : appel récursif)
         4^{\text{ème}} appel (x = 0)
             Retourne 0 : fin des appels récursifs...
             On commence à dépiler les valeurs de retour
          Fin du 4ème appel - retourne 0
      Fin du 3ème appel - retourne 1 + 0
   Fin du 2ème appel retourne 2 + 1
Fin du 1er appel retourne 3 + 3
Résultat final: 6
```

• Usage:

• Calculs de suites numériques : puissance, Fibonacci...

• Parcours d'arbres



- Limites
 - Certains langages limitent la profondeur de la pile d'appel.