

Récurtivité

Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 3 novembre 2025

Définition

Définition 1

Un **algorithme récursif** résout un problème en calculant des solutions d'instances plus petites du même problème.



Définition 2

Une **fonction ou procédure récursive** s'appelle elle-même.

Calcul de la factorielle : itératif

Définition de la factorielle d'un nombre

En mathématiques, la **factorielle** d'un entier naturel n est le **produit des nombres entiers strictement positifs inférieurs ou égaux à n** .

Spécification du calcul de la factorielle

- Donnée d'entrée : n , entier
- Donnée de sortie : r , entier
- Pré-condition : $n \geq 0$
- Post-condition : $r = n! = 1 \times 2 \times \dots \times (n - 1) \times n$

Calcul de la factorielle : itératif

```
FONCTION factorielle (n : entier) : entier
```

```
    VARIABLE resultat : entier
```

```
    VARIABLE i : entier
```

```
    resultat ← 1
```

```
    POUR i DE 1 A n PAR PAS DE 1
```

```
        resultat ← resultat * i
```

```
    FIN POUR
```

```
    RETOURNER resultat
```

```
FIN FONCTION
```

Calcul de la factorielle : récursif

Définition de la factorielle d'un nombre

En mathématiques, la **factorielle** d'un entier naturel n est le **produit des nombres entiers strictement positifs inférieurs ou égaux à n** .

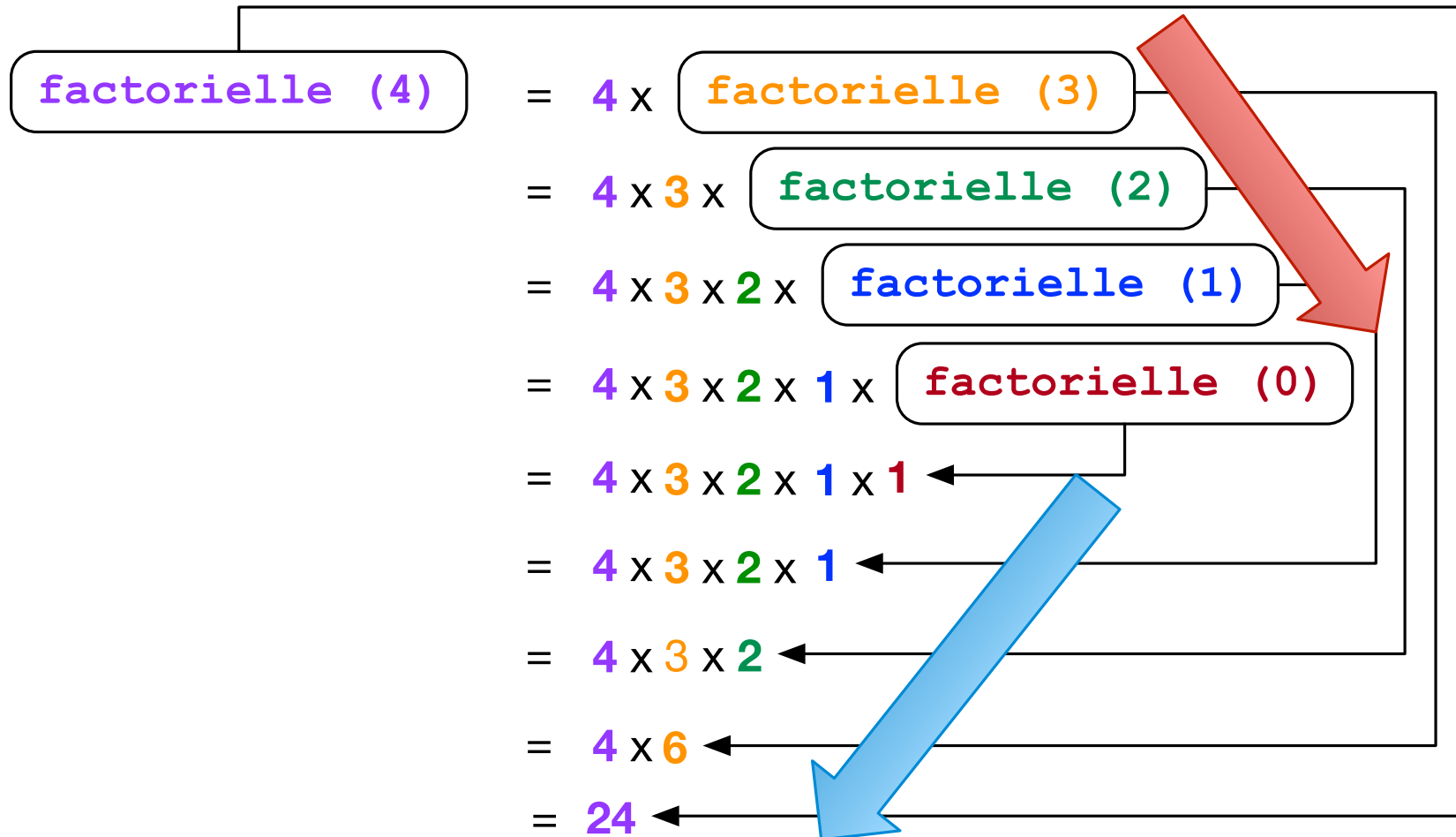
Spécification du calcul de la factorielle

- Donnée d'entrée : n , entier
- Donnée de sortie : r , entier
- Pré-condition : $n \geq 0$
- Post-condition :
 - si $n = 0$ $r = 1$
 - sinon $r = n \times \text{factorielle}(n - 1)$

Calcul de la factorielle : récursif

```
FONCTION factorielle (n : entier) : entier  
  
    SI n = 0 ALORS  
        RETOURNER 1  
  
    SINON  
        RETOURNER n * factorielle(n-1)  
  
    FIN SI  
  
FIN FONCTION
```

Calcul de la factorielle : récursif



Réversivité : règles

Règle 1

Tout algorithme réversif doit avoir **une ou plusieurs conditions de terminaison.**

- Arrêt de la réversivité et retour (d'un résultat si fonction)

Règle 2

Tout appel réversif doit **réduire la complexité du calcul** et **se rapprocher des conditions de terminaison**

Calcul de la factorielle : récursif

```
FONCTION factorielle (n : entier) : entier
```

Règle 1

```
SI n = 0 ALORS  
    RETOURNER 1
```

Condition de terminaison

Arrêt de la récursivité

```
SINON
```

```
    RETOURNER n * factorielle(n-1)
```

Règle 2

Appel récursif

Réduction de
la complexité

```
FIN SI
```

```
FIN FONCTION
```

Types de récursivité

- **Simple** :
 - La fonction (ou procédure) récursive s'appelle une seule fois
- **Multiple** :
 - La fonction (ou procédure) récursive s'appelle successivement fois
- **Croisée** :
 - 2 fonctions (ou procédures) récursives s'appellent l'une / l'autre
- **Imbriquée** :
 - Un des paramètres d'un appel récursif est un appel récursif

Réversivité terminale ou non terminale

Définition 1

Une fonction ou procédure est dite **réursive terminale** si elle **n'effectue aucun traitement après un appel réursif**, sinon elle est dite **réursive non terminale**.



Définition 2

La **déréursion** consiste à **transformer une fonction ou procédure réursive en équivalent itératif**.

Calcul de la factorielle : récursif

```
FONCTION factorielle (n : entier) : entier  
  
    SI n = 0 ALORS  
        RETOURNER 1  
  
    SINON  
        RETOURNER n * factorielle(n-1)  
  
    FIN SI  
  
FIN FONCTION
```



- Récursivité terminale ou non terminale ?

Dérécursivation d'une fonction procédure récursive

- Une fonction/procédure **récursive terminale** se transforme facilement en **TANT QUE**
- Une fonction/procédure **récursive non terminale** se transforme aussi en **itération**, mais c'est moins simple . . .



Suite de Fibonacci (récursif)

Enoncé du problème

La suite de Fibonacci est une suite de nombres entiers dont chaque terme successif représente la somme des deux termes précédents, et qui commence par 0 puis 1. On veut calculer le terme de rang n .

Spécification du problème



Signature de la fonction



Suite de Fibonacci (récursif)

Enoncé du problème

La suite de Fibonacci est une suite de nombres entiers dont chaque terme successif représente la somme des deux termes précédents, et qui commence par 0 puis 1. On veut calculer le terme de rang n .

Spécification du problème

- Donnée d'entrée : n , entier (indice du terme)
- Donnée de sortie : entier (terme d'indice n)
- Pré-condition : $n \geq 0$
- Post-condition :
 - Si $n = 0$ ou $n = 1$ alors $\text{fibonacci}(n) = n$
 - Sinon $\text{fibonacci}(n) = \text{fibonacci}(n-2) + \text{fibonacci}(n-1)$

Signature de la fonction

- **fibonacci** ($n : \text{entier}$) : entier

Suite de Fibonacci (récursif)

```

FUNCTION fibonacci (n : entier) : entier

    SI n = 0 OU n = 1 ALORS
        RETOURNER n
    FIN SI

    RETOURNER fibonacci(n-2) + fibonacci(n-1)

FIN FUNCTION

```

- Version récursive non terminale ...

Longueur de la suite de Syracuse (récursif)

Enoncé du problème

Une suite de Syracuse est une suite d'entiers naturels $(u_0 \dots u_i)$ définie de la manière suivante : on part d'un nombre entier strictement positif ; s'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on lui ajoute 1. Il existe un indice i tel que $u_i = 1$. On cherche l'indice i .

Spécification du problème



Signature de la fonction



Longueur de la suite de Syracuse (récursif)

Enoncé du problème

Une suite de Syracuse est une suite d'entiers naturels $(u_0 \dots u_i)$ définie de la manière suivante : on part d'un nombre entier strictement positif ; s'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on lui ajoute 1. Il existe un indice i tel que $u_i = 1$. On cherche l'indice i .

Spécification du problème

- Donnée d'entrée : i , entier (indice de départ)
- Donnée d'entrée : n , entier (terme de départ)
- Donnée de sortie : entier (indice du 1er terme valant 1)
- Pré-condition : $i \geq 0, n \geq 1$
- Post-condition :
 - Soit tranforme la transformation de syracuse
 - Si $n = 1$ alors $\text{syracuse}(i, n) = i$
 - Sinon $\text{syracuse}(i, n) = \text{syracuse}(i+1, \text{transforme}(n))$

Signature de la fonction

Longueur de la suite de Syracuse (récuratif)

```
FONCTION syracuse (i : entier, n : entier) : entier
```

```
    VARIABLE np1 : entier
```

```
    SI n = 1 ALORS
```

```
        RETOURNER i
```

```
    FIN SI
```

```
    SI (n mod 2 = 0) ALORS
```

```
        np1  $\leftarrow$  n div 2
```

```
    SINON
```

```
        np1  $\leftarrow$  (3 * n) + 1
```

```
    FIN SI
```

```
    RETOURNER syracuse(i+1, np1)
```

```
FIN FONCTION
```

Longueur de la suite de Syracuse (récursif)

- Remarque : le premier paramètre n'est pas utile pour l'appel initial
 - Implicitement égal à 0
- On peut fournir une fonction spécifique pour cet appel initial

```
FONCTION longueur_syracuse (n : entier) : entier
```

```
    RETOURNER syracuse(0, n)
```

```
FIN FONCTION
```

Inversion des chiffres d'un nombre décimal (récuratif)

Enoncé du problème

On cherche un entier i dont les chiffres sont dans l'ordre inverse d'un entier n (ex : $123 \rightarrow 321$).

Spécification du problème



Signature de la fonction



Inversion des chiffres d'un nombre décimal (récuratif)

Enoncé du problème

On cherche un entier i dont les chiffres sont dans l'ordre inverse d'un entier n (ex : $123 \rightarrow 321$).

Spécification du problème

- Donnée d'entrée : r , entier (chiffres restant à inverser)
- Donnée d'entrée : i , entier (chiffres déjà inversés)
- Donnée de sortie : entier (nombre inversé)
- Pré-condition : $i \geq 0$
- Post-condition :
 - Si $r = 0$ alors $\text{inverse}(r, n) = n$
 - Sinon $\text{inverse}(r, n) = \text{inverse}(r \text{ div } 10, (10 * i) + (r \text{ mod } 10))$

Signature de la fonction

- **inverseRec** ($r : \text{entier}, i : \text{entier}$) : entier

Inversion des chiffres d'un nombre décimal (récuratif)

```
FUNCTION inverseRec (r : entier, i : entier) : entier

    SI r = 0 ALORS
        RETOURNER i
    FIN SI

    RETOURNER inverseRec(r div 10, (i * 10) + (r mod 10))

FIN FUNCTION
```

```
FUNCTION inverse (r : entier) : entier

    RETOURNER inverseRec(r, 0)

FIN FUNCTION
```
