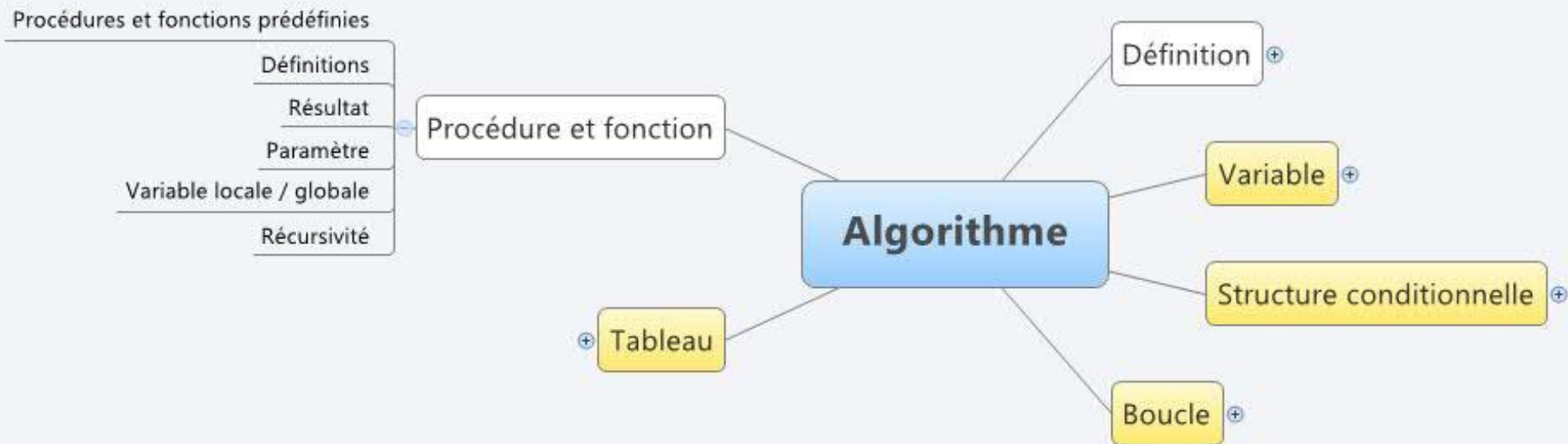


LES PROCÉDURES ET LES FONCTIONS

Plan du cours



Entrer dans un magasin de chaussures

Choisir des chaussures rouges

Enlever mes vieilles chaussures

Mettre les nouvelles chaussures

Marcher un peu

Me regarder dans un miroir

Enlever les nouvelles chaussures

Mettre mes vieilles chaussures

Choisir des chaussures vertes

Enlever mes vieilles chaussures

Mettre les nouvelles chaussures

Marcher un peu

Me regarder dans un miroir

Enlever les nouvelles chaussures

Mettre mes vieilles chaussures

Sortir du magasin

La même chose,
avec une procédure...

Entrer dans un magasin de chaussures

Essayer des chaussures (rouges)

Essayer des chaussures (vertes)

Sortir du magasin

Fonction Essayer des chaussures(couleur)

Enlever mes vieilles chaussures

Mettre les nouvelles chaussures \$couleur

Marcher un peu

Me regarder dans un miroir

Enlever les nouvelles chaussures

Mettre mes vieilles chaussures

Fin de fonction

Procédures et fonctions prédéfinies

- Tout langage de programmation propose un grand nombre de procédures et de fonctions prédéfinies
 - Pour nous simplifier la vie
 - Pour nous faire gagner du temps

Procédures et fonctions prédéfinies

- Fonctions mathématiques
 - sin, cos, moyenne, nombre aléatoire, ...
- Fonctions sur des chaînes de caractères
 - longueur, extraire une sous-chaîne, mettre en majuscules, ...
- Fonctions sur les tableaux
 - Longueur, tri, présence dans le tableau, agglomération de 2 tableaux, ...
- Fonctions diverses et variées
 - Accès aux bases de données
 - affichage graphique
 - manipulation d'images
 - manipulation de fichiers XML
 - ...

Définitions - Procédure

Une **procédure** est un ensemble d'instructions que l'on déclenche par un appel, en donnant des paramètres.

```
PROCEDURE <nom>([<paramètre> : <type>]*)  
  <liste d'instructions>  
FIN DE PROCEDURE
```

ECRIRE() est une procédure que nous avons déjà beaucoup utilisée...

Définitions - Fonction

Une **fonction** est un ensemble d'instructions que l'on déclenche par un appel, en donnant des paramètres, **et qui retourne un résultat d'un type défini.**

```
<type> FONCTION <nom>([<paramètre> : <type>]*)  
    <liste d'instructions>  
    RETOUNER <valeur>  
FIN DE FONCTION
```

LIRE() est une procédure que nous avons déjà beaucoup utilisée...

Résultat

- Une procédure ne renvoie **jamais** de résultat
- Une fonction retourne **toujours** un résultat
 - Le type du résultat est défini lors de la création de la fonction
 - Il faut obligatoirement au moins une occurrence du mot-clé *RETOURNER*
 - La fonction s'arrête dès qu'elle a effectuer un instruction de type *RETOURNER*

A ← 3

B ← MultiplierParDeux(A) # B vaut 6

```
Entier FONCTION MultiplierParDeux(nombre : Entier)
```

```
    RETOURNER nombre * 2
```

```
FIN DE FONCTION
```


Paramètres

- Les procédures et les fonctions peuvent accepter de très nombreux paramètres. Le nombre maximum de paramètres est propre à chaque langage
- L'ordre des paramètres est très important !!! Les valeurs données lors de l'appel sont attribuées aux paramètres dans l'ordre défini lors de la déclaration de la fonction ou de la procédure
- Il est obligatoire de donner une valeur à chaque paramètre. Sinon, il y a une erreur de compilation ou d'exécution

Variables globales & locales

(Exemple idiot, à ne pas reproduire... Il faut choisir des noms qui ont plus de sens, et éviter d'utiliser le même nom pour 2 choses différentes !!!)

A ← 10

B ← 20

C ← soustraction(B, A) # C vaut 10

```
Entier FONCTION soustraction(A : Entier,
                              B : Entier)
```

```
    RETOURNER A - B
```

```
FIN DE FONCTION
```

Contexte **global**

A = 10

B = 20

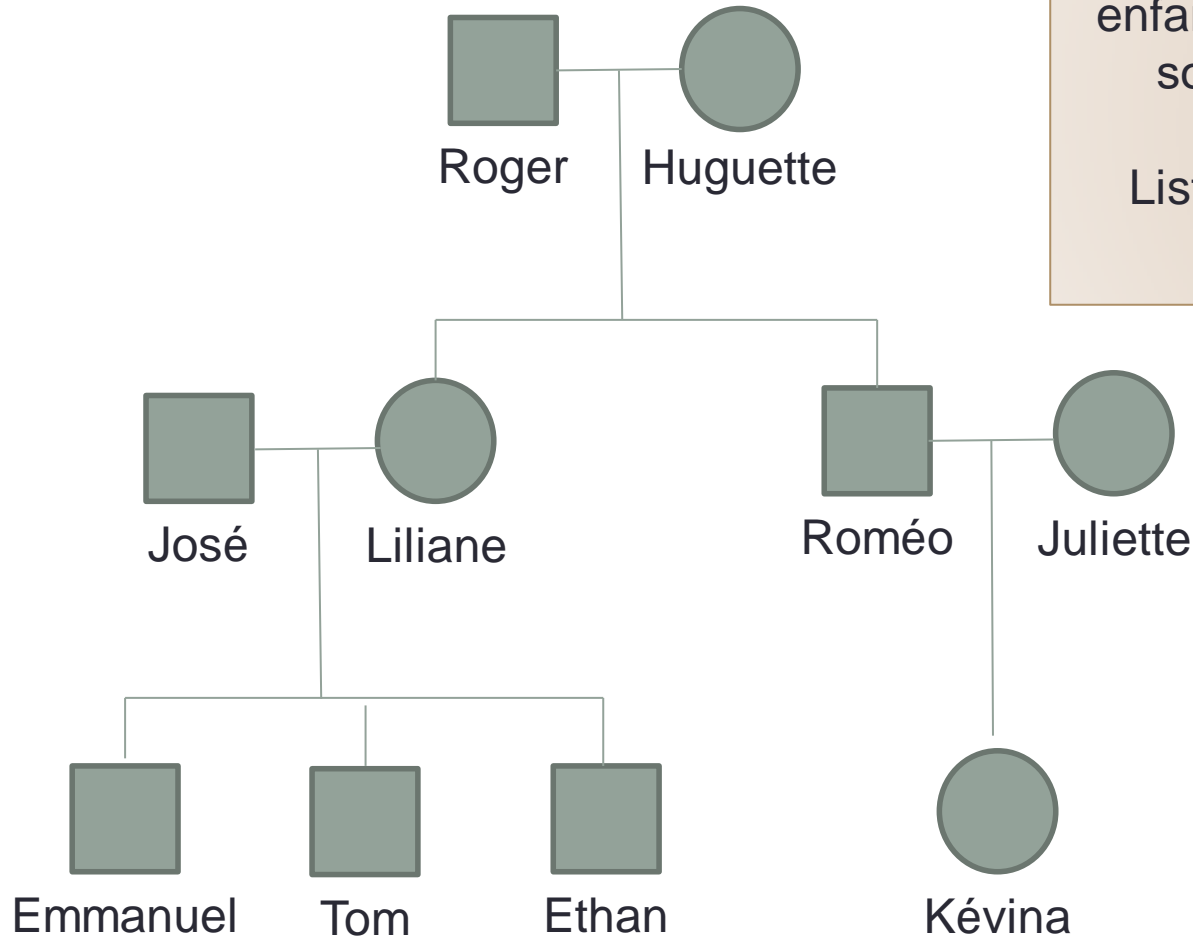
Deux contextes d'exécution différents, dans des zones mémoire différentes, où les noms A et B font référence à des variables différentes...

Contexte **local** à la fonction

A = 20

B = 10

Récurtivité - Exemple



Chaque être humain est un enfant, et chaque enfant peut à son tour devenir parent...

Lister les enfants de chacun

Récurtivité

- Un algorithme est dit « *récuratif* » s'il s'appelle lui-même.

```
Entier FONCTION factorielle (k : entier)
    SI k=0
    ALORS
        RETOURNER 1
    SINON
        RETOURNER k * factorielle (k-1)
    FINSI
FIN DE FONCTION
```

Récurtivité

- Rappel : en mathématiques, la factorielle se définit ainsi :

$$n! = \prod_{i=1}^n i = 1 \times 2 \times 3 \times \dots \times (n-1) \times n$$

$$0! = 1$$

$$1! = 1 * 0!$$

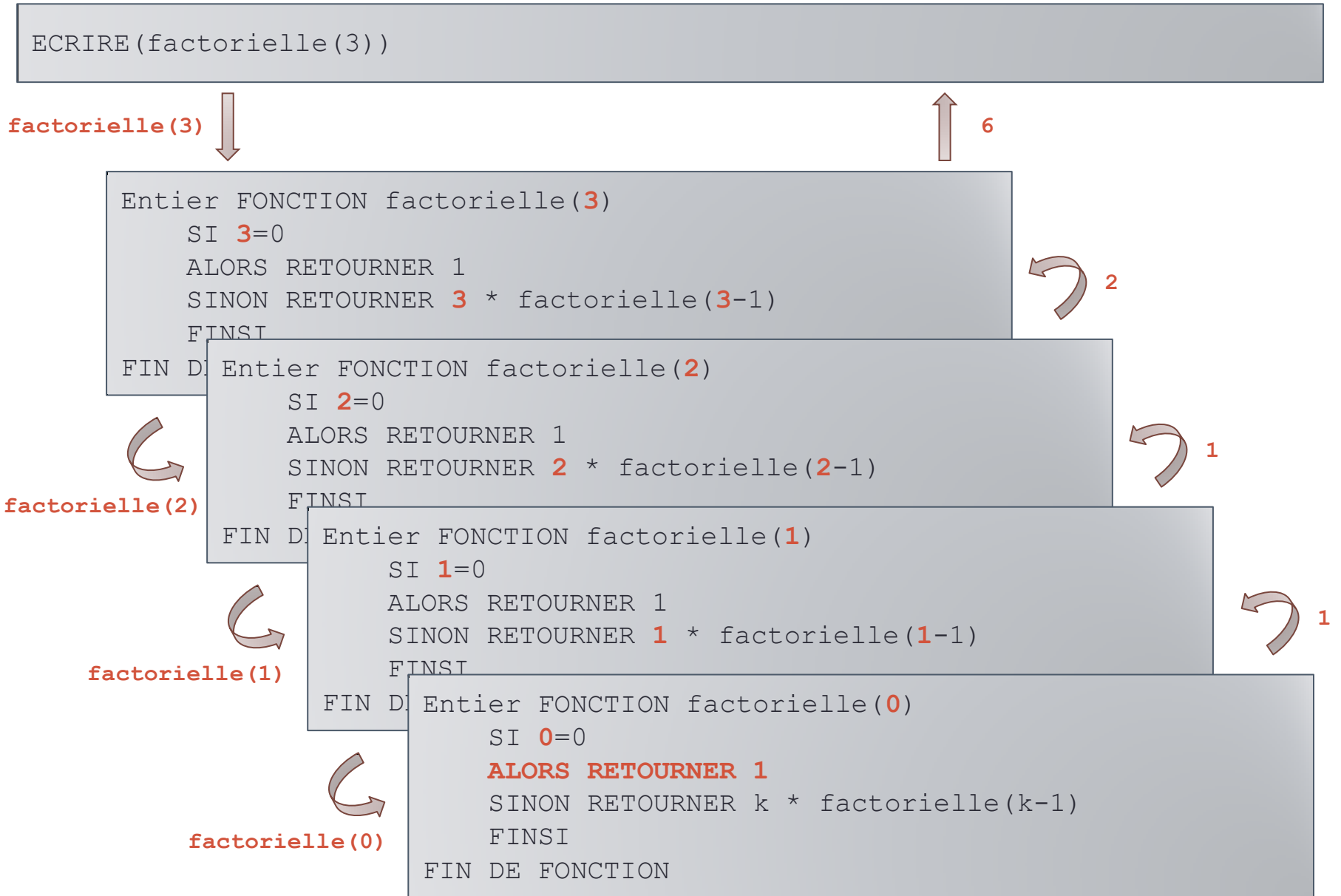
$$2! = 2 * 1! = 2 * 1 * 0!$$

$$3! = 3 * 2! = 3 * 2 * 1 * 0!$$

$$4! = 4 * 3! = 4 * 3 * 2 * 1 * 0!$$

$$n! = n * (n-1)!$$

```
Entier FONCTION factorielle(k : entier)
  SI k=0
    ALORS RETOURNER 1
  SINON RETOURNER k * factorielle(k-1)
  FINSI
FIN DE FONCTION
```



Récurtivité et boucles

- `compter(1, 10)`

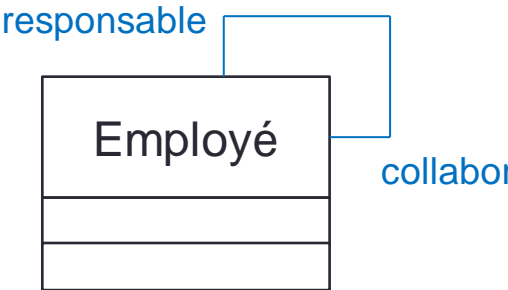
```
PROCEDURE compter(aPartirDe, jusquA)
POUR i DE aPartirDe A jusquA PAR PAS DE 1
    ECRIRE(i)
FINPOUR
FIN DE PROCEDURE
```

`compter(1, 10)`

- **Condition initiale**
- Il se passe quelque chose entre 2 appels
- **Condition finale**

- ```
PROCEDURE compter(aPartirDe, jusquA)
SI aPartirDe <= jusquA
 ECRIRE(aPartirDe)
 compter((aPartirDe + 1), jusquA)
FINSI
FIN DE PROCEDURE
```

# La récursivité : dans quels cas ?

- Il y a des cas, relativement rares, où un algorithme récursif est plus simple qu'un algorithme classique
  - Affichage de structures arborescentes
    - Position dans l'arborescence d'un site web
    - Données sous forme arborescente
    - Système de fichiers
  - Structures de données récursives :

```
graph LR; E[Employé] -- responsable --> E1[]; E1 -- collaborateur --> E
```
  - ...



# Exercices

- Exercices sur les fonctions : 5.1 à 5.x