

## Fonctions récursives (5.1)

## Récurtivité

- Il est parfois difficile de définir un objet
- Il est parfois plus simple de définir cet objet en fonction de lui-même
- Ce procédé s'appelle la **récurtivité**
- On peut utiliser la récurtivité pour définir des ensembles, des suites, des fonctions

## Récurtion en programmation

- Quand une méthode (fonction) s'appelle elle-même
- Un exemple classique: **La fonction factorielle**

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$$

- Définition récurtive:

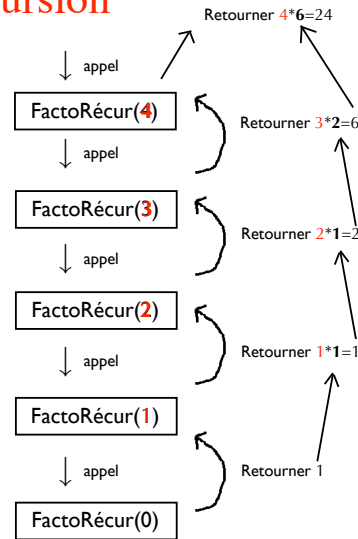
$$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot f(n-1) & \text{sinon} \end{cases}$$

## Éléments essentiels d'une méthode récurtive

- Un (ou plusieurs) cas de base
  - ▣ Les valeurs d'entrées pour lesquelles on ne fait aucun appel récurtif sont appelées les **cas de base**
- Appels récurtifs
  - ▣ Appels de la méthode courante
  - ▣ Chaque suite d'appels récurtifs **doit** essentiellement se terminer sur un cas de base

## Trace d'une récursion

- Une boîte pour chaque appel récursif
- Une flèche pour chaque appel
- Une flèche pour chaque retour d'une valeur



## Un autre exemple de récursion linéaire

### Algorithme SommeLinaire(A,n)

Entrées: Une liste d'entiers A et un entier  $n \geq 1$ , tel que A contient au moins n éléments

Sortie: La somme des n premiers entiers de A

Si  $n=1$  alors

retourner  $A[0]$

Sinon

retourner SommeLinaire(A,n-1)+  $A[n-1]$

## Bien définir la récursion

- Lorsqu'on crée des méthodes récursives, il est important de les définir de façon à faciliter la récursion
- Cela implique parfois qu'il faut donner des paramètres additionnels en entrée à la méthode
- Pour définir une fonction récursive qui inverse les éléments d'une liste, il est plus facile de définir une méthode InverseListe(A,i,j) qui prend en entrée deux indices i et j en plus de la liste a inverser.

## Inverser les éléments d'une liste

### Algorithme InverseListe(A,i,j)

Entrées: Une liste d'entiers A et des indices i et j

Sortie: La liste ayant les éléments de l'indice i à j dans l'ordre inverse

Si  $i < j$  alors

échanger  $A[i]$  et  $A[j]$

InverseListe(A,i+1,j-1)

Retourner A

## “Tail recursion”

- Lorsqu’une méthode récursive exécute son appel récursif comme dernière étape
- La méthode InverseListe en est un exemple
- Ces méthodes peuvent facilement être converties en des méthodes non récursives (ce qui sauve de l’espace mémoire)

## Récursion Binaire

- Quand une méthode fait deux appels récursifs

- Exemple:

Algorithme SommeBinaire(A,i,n)

Entrées: Une liste d’entiers A et des entiers i et n

Sortie: La somme de n entiers dans A, où l’on commence à additionner à partir de l’indice i

Si  $n = 1$  alors

retourner  $A[i]$

Sinon

retourner SommeBinaire[A,i,n/2] +

SommeBinaire[A,i+n/2,n/2]

## Calculer les nombres de Fibonacci

- Les nombres de Fibonacci sont définis récursivement

$$F_0 = 0$$

$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2} \quad \text{pour } i > 1$$

- Les calculer avec un algorithme récursif (essai 1)

Algorithme FibBinaire(k)

Entrées: Une entier  $k \geq 0$

Sortie:  $F_k$

Si  $k = 0$  ou  $1$  alors

retourner  $k$

sinon

retourner FibBinaire(k-1) + FibBinaire(k-2)

## Un meilleur algorithme

Algorithme FibLineaire(k)

Entrées: Une entier  $k \geq 0$

Sortie:  $(F_k, F_{k-1})$

Si  $k = 1$  alors

retourner  $(k, 0)$

sinon

$(i, j) := \text{FibLineaire}(k-1)$

retourner  $(i+j, i)$

➔ Complexité en temps:  $O(k)$