R3.02 : Développement efficace

Notion de liste

J-F. Kamp

Septembre 2024

Les listes chaînées

Définition d'une liste

Ensemble formé d'un nombre variable (éventuellement zéro) de données.

Tous les éléments sont rangés les uns derrière les autres.

Il existe un début *head* et une fin *end*.

On parcourt la liste avec un curseur (*current*) en passant d'un élément à son voisin.

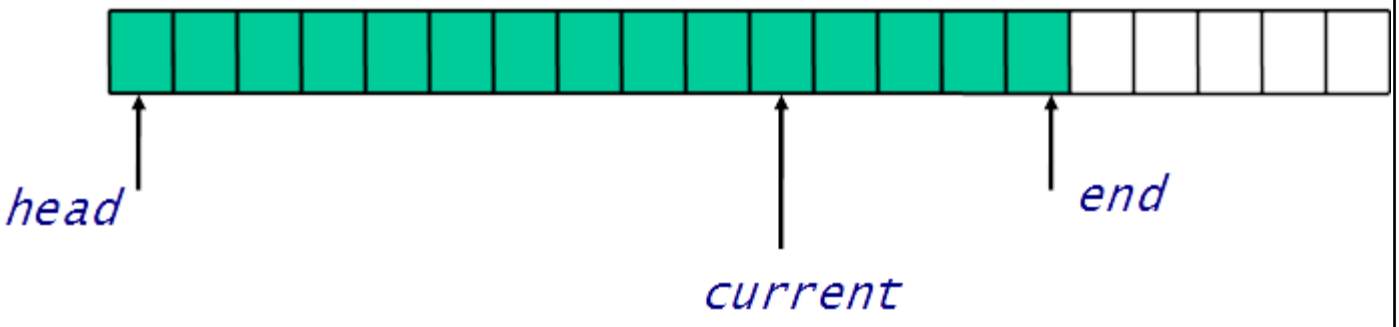
Définition d'une liste

On doit pouvoir effectuer les opérations suivantes :

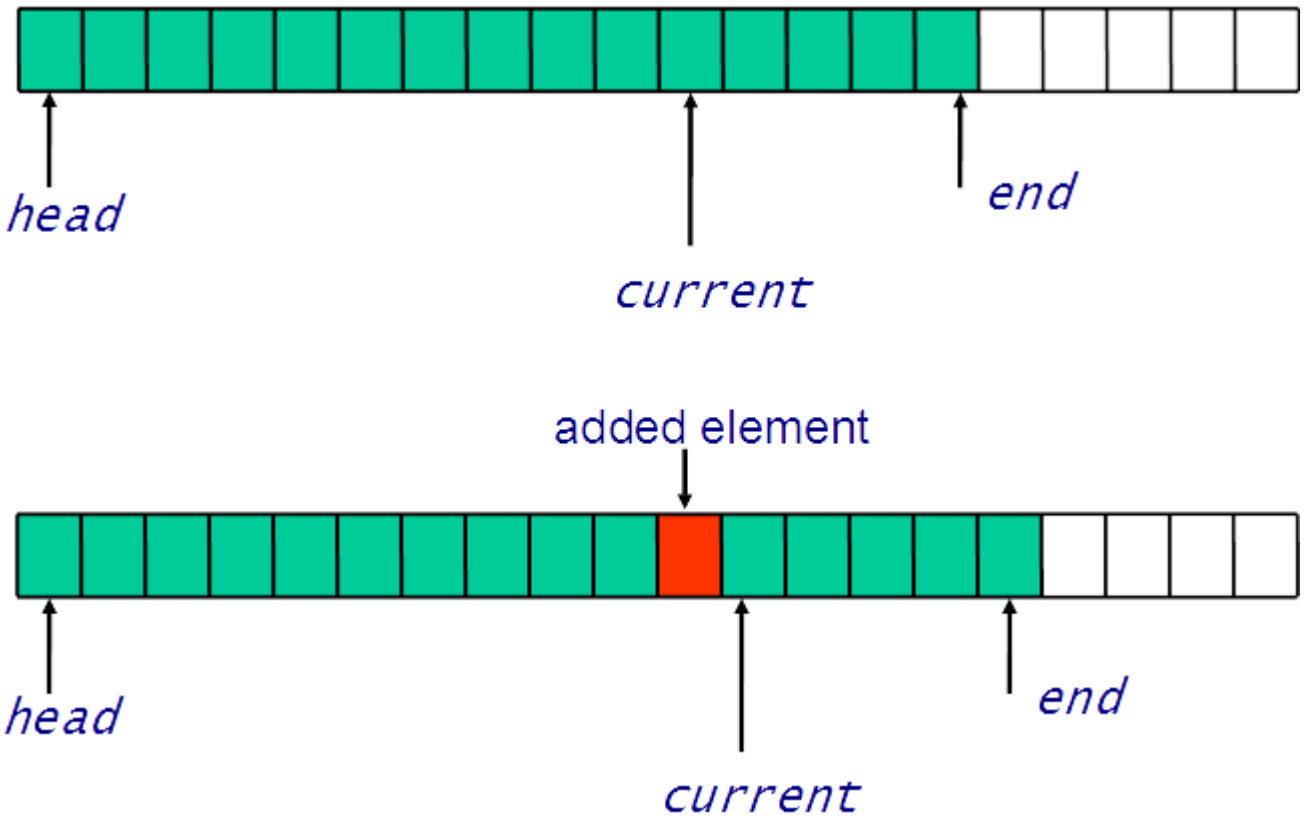
- ajout d'une nouvelle donnée en position *i*
- consultation de la donnée en position *i*
- suppression de la donnée en position *i*
- modification de la donnée en position *i*
- test si la liste est vide

Mise en œuvre par un tableau

- Tableau de *n* éléments d'un type défini.
- Une variable *head* : indice du premier élément.
- Une variable *end* : indice du dernier élément.
- Une variable *current* : indice de l'élément courant.
- A tout instant le tableau est rempli entre les indices *head* et *end*.



Insertion avant le current



Pour l'insertion : créer une place par décalage.

Pour suppression : éliminer une place par décalage.

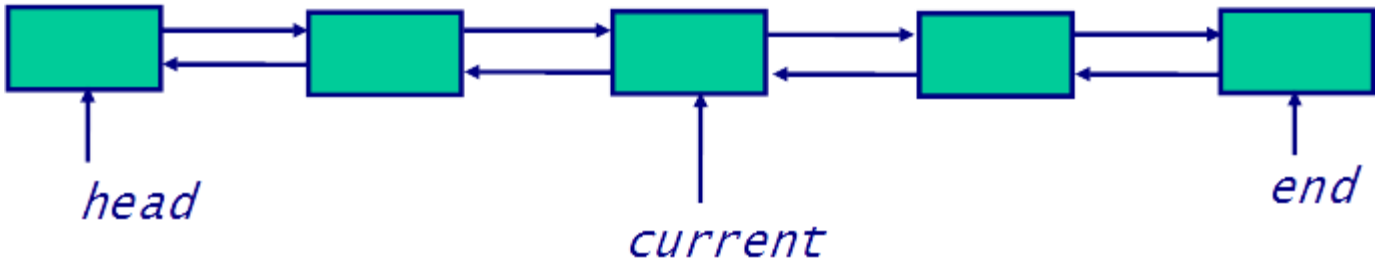
Mise en œuvre par un tableau : évaluation

- Avantages :
  - solution simple
  - accès rapide au *ième* élément (O(1))

- Inconvénients :
  insertions/suppressions coûteuses (O(n)) sauf en *end*

- Mécanisme adapté aux listes suivantes :
  - listes constantes ou subissant peu de modifications
  - listes modifiées essentiellement en début ou en fin

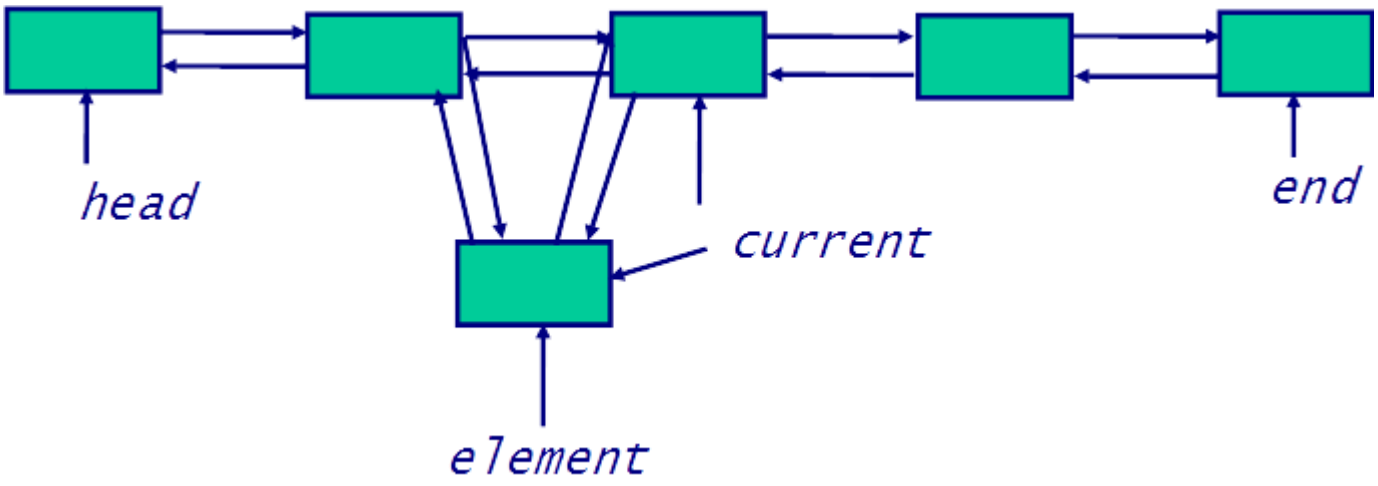
Mise en œuvre par une liste chaînée



Liste doublement chaînée :

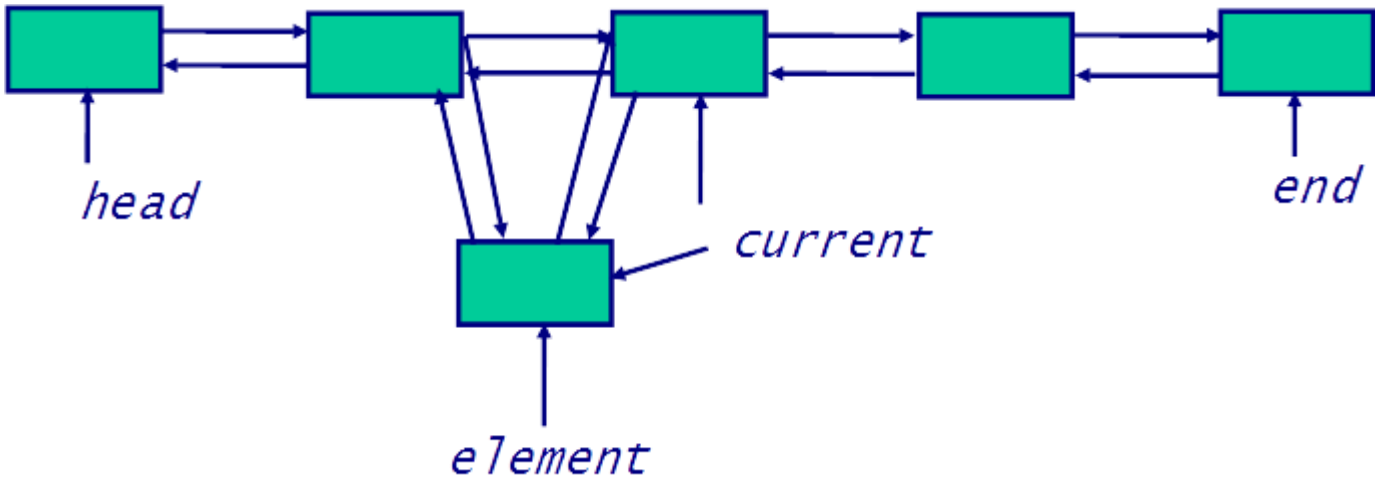
- chaque élément est un objet
- relié à un objet identique à sa gauche (*prev*) et à sa droite (*next*) par référence
- le premier élément *head* n'a pas de voisin à gauche (null)
- le dernier élément *end* n'a pas de voisin à droite (null)

Insertion à gauche de current



- création du nouvel objet (*element*)
- relier ce nouvel objet à ses 2 voisins l'un à gauche (*prev*) et l'autre à droite (*next*)
- relier les voisins gauche et droit à ce nouvel objet
- placer *current* sur le nouvel objet (*element*)

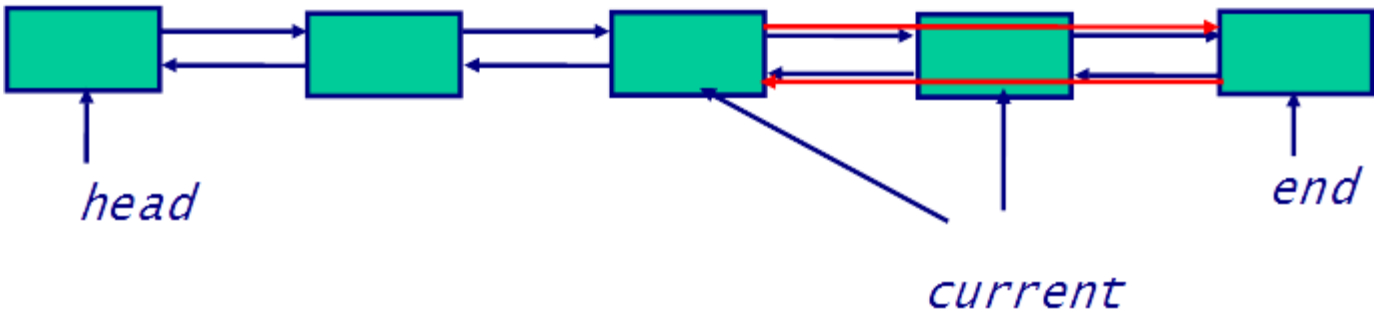
Insertion à gauche de current



Cas particuliers :

- l'insertion du tout premier élément
- l'insertion d'un élément si *current* est positionné sur *head*

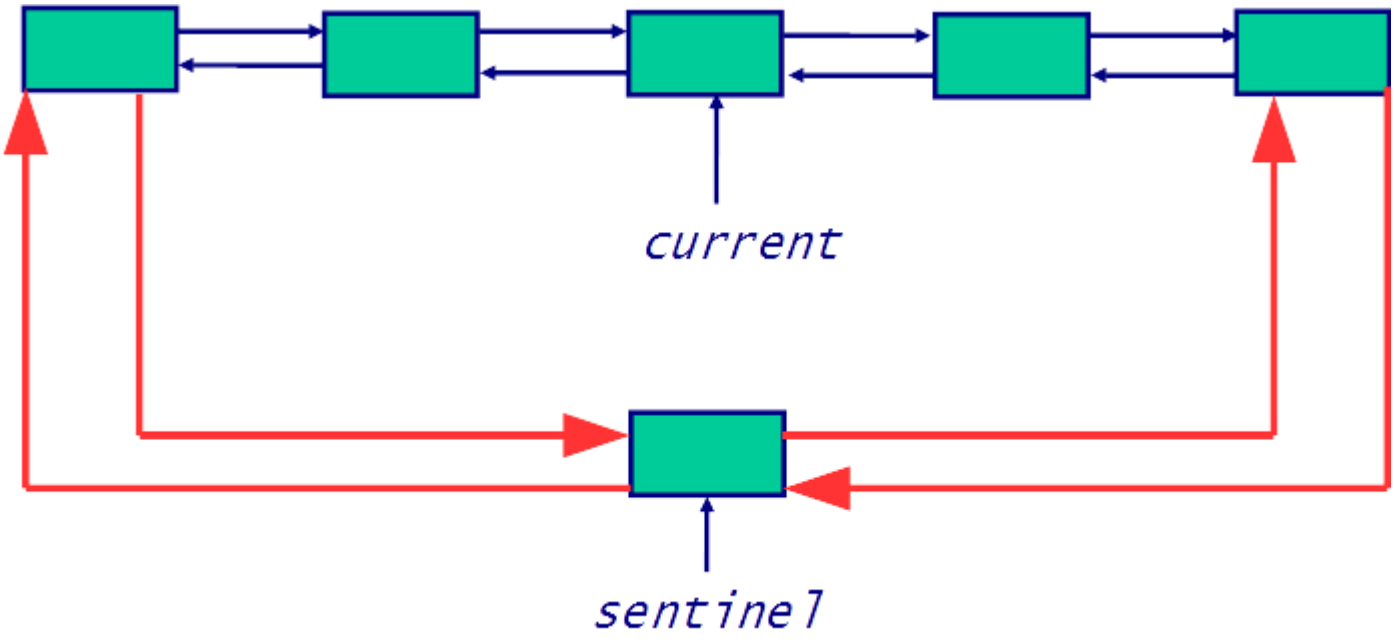
Suppression de l'élément current



Cas particuliers :

- la suppression du tout dernier élément
- la suppression de l'élément si *current* est positionné sur *head* ou sur *end*

En pratique : version avec sentinelle



Avantages :

- tous les cas particuliers disparaissent
- liste vide = sentinelle toute seule et *current* pointe dessus
- liste non vide : *current* ne se trouve jamais sur la sentinelle

Mise en œuvre par liste chaînée : évaluation

- Avantages :
  - insertion/suppression peu coûteuses lorsqu'on est positionné sur l'élément à modifier (O(1))
  - occupation mémoire dynamique
- Inconvénient :
  - accès coûteux au *ième* élément car il faut parcourir les *i* premiers éléments un par un (O(n), si chaîne de n éléments)