Structures de données, IMA S6

Variantes des listes chaînées

N. Devésa, Polytech'Lille et A. Miné, ÉNS

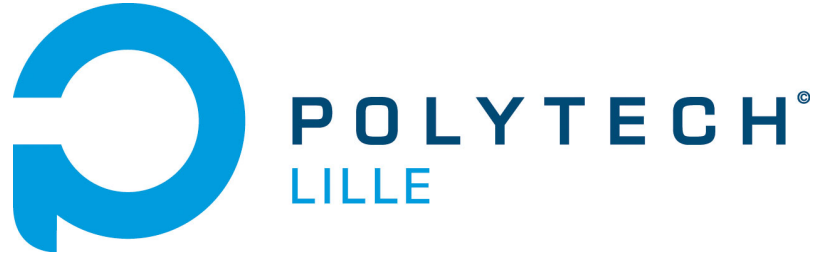
Laure Gonnord

http://laure.gonnord.org/pro/teaching/

Laure.Gonnord@polytech-lille.fr

Université Lille 1 - Polytech Lille

Février 2011



Listes avec sentinelle

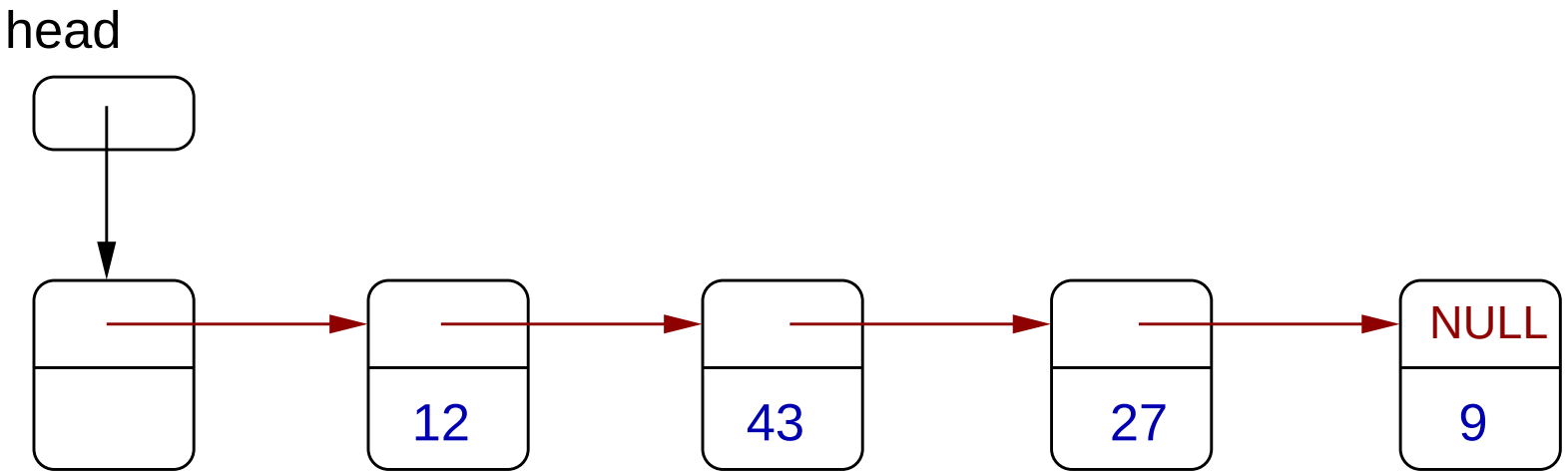
Idée : ajout d'une **cellule sentinelle** en tête de liste :

- `head->next` pointe sur la première cellule de la liste,
- `head->data` n'est pas utilisé,
- toutes les fonctions prennent en argument un **pointeur vers la sentinelle**.

Avantage : simplifie la gestion des cas limites :

- cas où la liste est vide,
  (la liste contient toujours au moins une cellule)
- cas où le premier élément de la liste est modifié.
  (`head->next` est modifié, pas de tête de liste à retourner)

Illustration d'une liste avec sentinelle



Une cellule est toujours définie comme ceci :

Structure *Cellule*
    valeur : Entier
    suivant : **pointeur de** *Cellule*
FStruct

Opérations sur les listes avec sentinelle - 1

Fonction *creerListe() :***pointeur de** *Cellule*
| **L**: pc :**pointeur de** *Cellule*
| pc ← (**pointeur de** *Cellule*) allouer()
| pc↑.suivant ← NULL
| **Retourner** *pc*
FFonction

Action *insereTeteListe(hd,elem)*
| **D**: hd :**pointeur de** *Cellule*
| **D**: elem :Int
|
|
|
|
|
FAction

Opérations sur les listes avec sentinelle - 2

Action *insereFinListe(hd,elem)*
D: hd :**pointeur de** *Cellule*
D: elem :Int

FAction

Concaténation : cf **TD**.

Listes doublement chaînées

Liste doublement chaînée = on maintient :

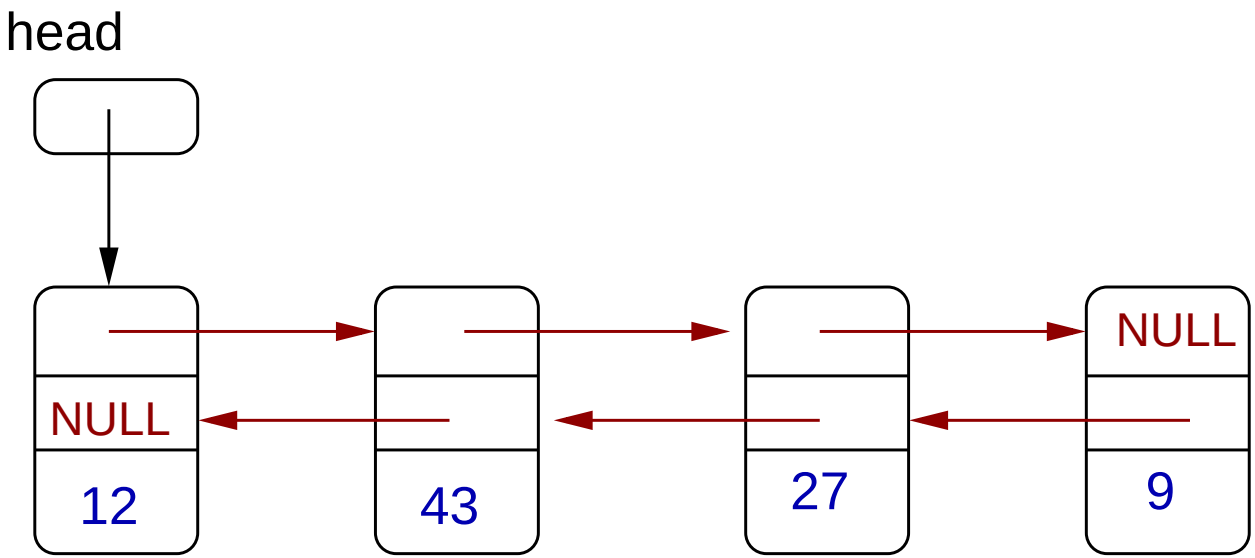
- un pointeur vers la cellule suivante,
- un pointeur vers la cellule précédente.

Déclaration de type cellule :

Structure *Cellule*
| valeur : Entier
| suiv : **pointeur de** *Cellule*
| prec : **pointeur de** *Cellule*
FStruct

```
struct cell {
   struct cell* next;
   struct cell* prev;
   int          data;
};
```

Illustration d'une liste doublement chaînée

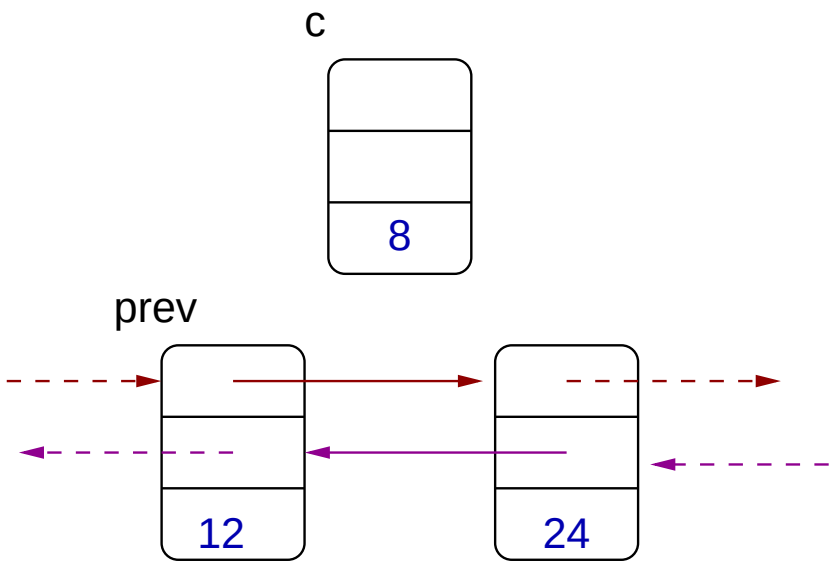


- `head` pointe vers la première cellule,
- `c->next=NULL` pour la dernière cellule, sinon `c->next->prev` existe
- `c->prev=NULL` pour la première cellule, sinon `c->prev->next` existe

Insertion dans une liste doublement chaînée - 1

Spécification :

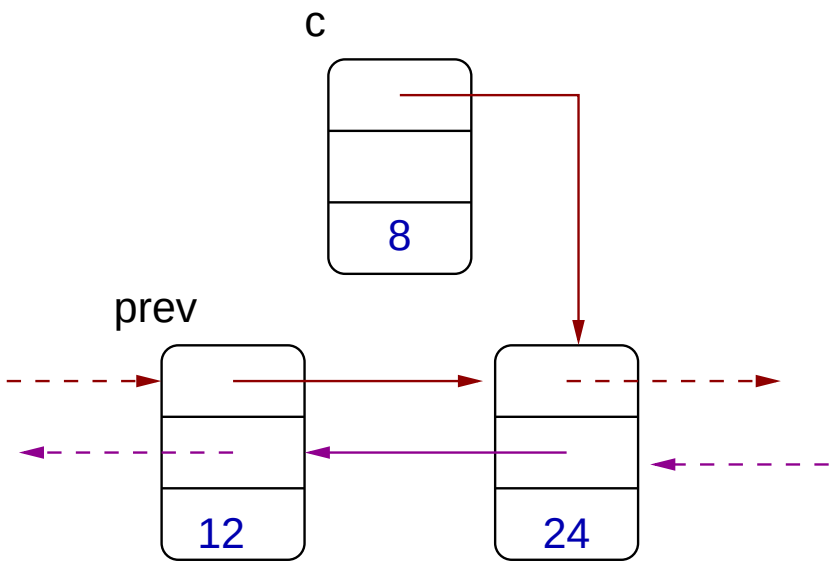
Insère une cellule avec l'élément `elem`(ici 8) après `prev`.



Insertion dans une liste doublement chaînée - 1

Spécification :

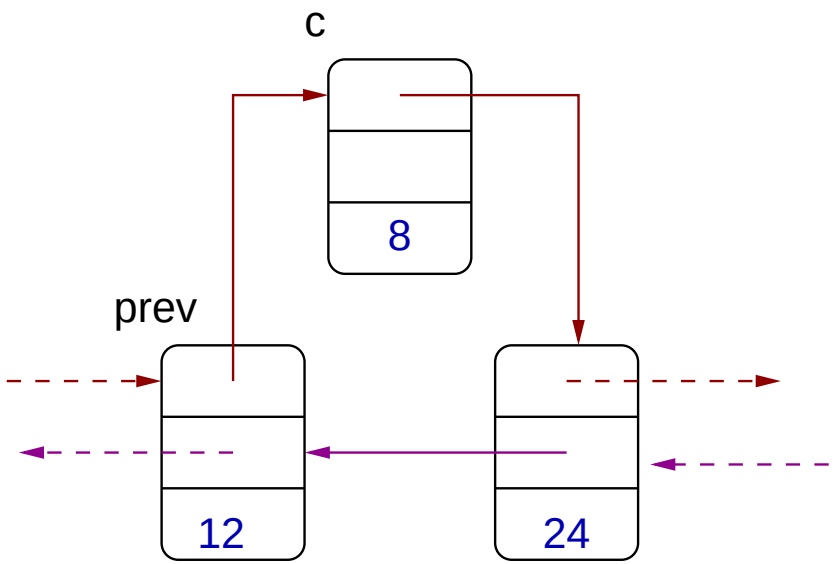
Insère une cellule avec l'élément `elem`(ici 8) après `prev`.



Insertion dans une liste doublement chaînée - 1

Spécification :

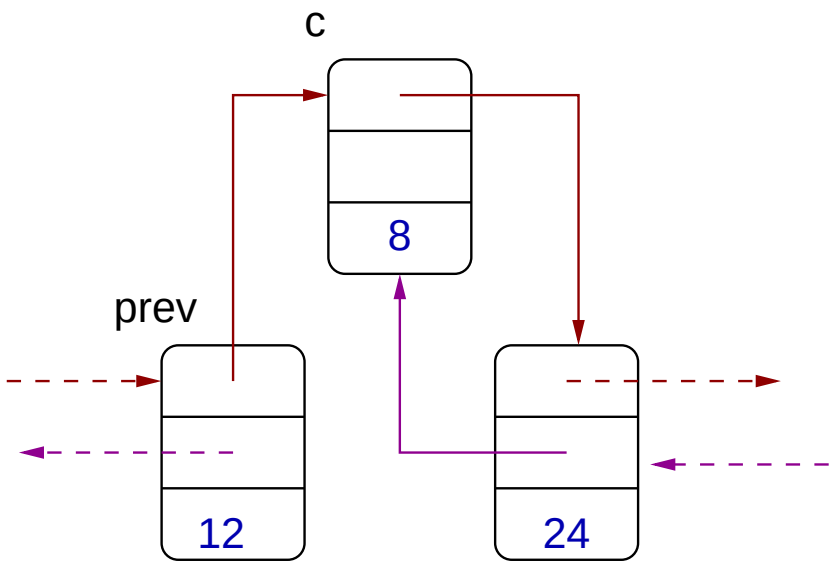
Insère une cellule avec l'élément `elem`(ici 8) après `prev`.



Insertion dans une liste doublement chaînée - 1

Spécification :

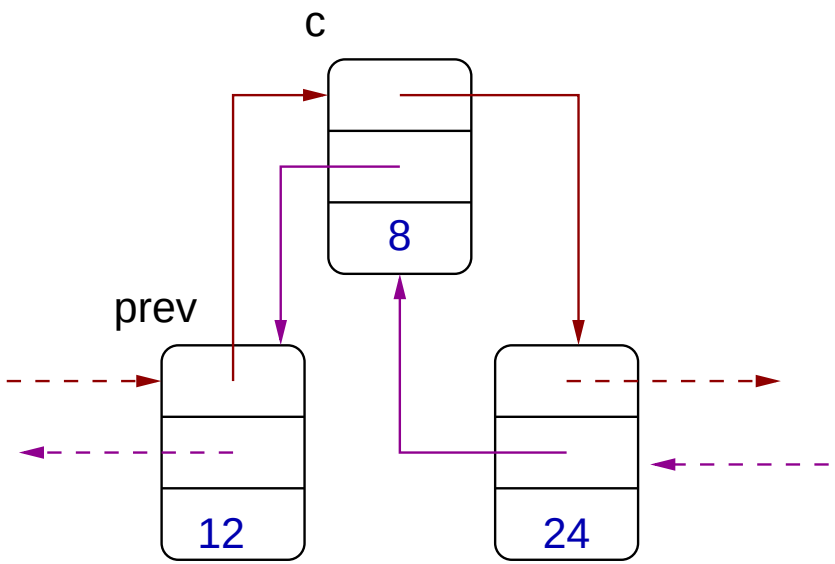
Insère une cellule avec l'élément `elem`(ici 8) après `prev`.



Insertion dans une liste doublement chaînée - 1

Spécification :

Insère une cellule avec l'élément `elem`(ici 8) après `prev`.



Insertion dans une liste doublement chaînée - 2

- coût constant, sans compter le calcul de `prev`,
- ne marche pas pour insérer en première position,
- pour permettre l'insertion en dernière position, **on change quoi ?**

Insertion dans une liste doublement chaînée -3

Spécification :

Insère une cellule avec l'élément `elem`(ici 8) avant `next`.

```
void insere_avant(struct cell* next, int elem) {
  struct cell *c = malloc(...);
  c->data = elem;
  c->prev = next->prev;
  next->prev = c;
  c->prev->next = c;
  c->next = next;
}
```

Remarques :

- on a simplement inversé les mots `prev` et `next`,
- ne marche pas pour insérer en dernière position,
- facilement modifiable pour permettre l'insertion en première position.

Comparaison des types de listes

Listes doublement chaînées *vs.* listes simplement chaînées :

- on doit maintenir deux pointeurs par cellule au lieu d'un :
  - chaque opération est légèrement plus coûteuse,
  - plus complexe, risques de bug accrus,
- certaines opérations sont plus faciles.
  (*e.g.*, parcours en arrière, insertion avant une cellule)

Schéma classique en informatique : l'ajout de redondance permet des oui calculs plus rapides, au prix d'une maintenance plus complexe.

Variantes : listes avec sentinelles, listes circulaires, listes avec un pointeur de fin, . . .

- Coût des listes avec pointeur de fin ?