Algorithmique et structures de données

CM 3 Listes doublement chaînées et listes circulaires

Plan du CM 3

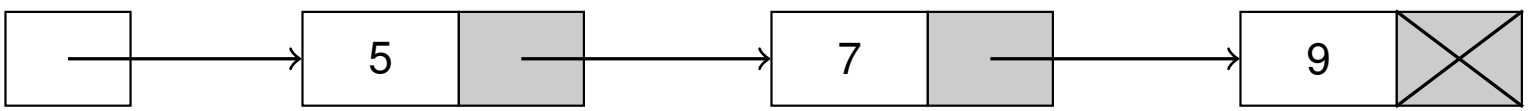
Plan du CM3

Rappel

structure récursive

```
structure noeud {
    valeur : entier // le type est fixé
    suivant : pointeur sur noeud
}
type liste = pointeur sur noeud
```

Exemple de liste



Opérations élémentaires

Les opérations élémentaires peuvent être regroupées en trois familles

- recherche d'un ou plusieurs éléments
- insertion d'un ou plusieurs éléments
- suppression d'un ou plusieurs éléments

Accès séquentiel

Pour ces trois familles d'algorithmes l'accès se fait obligatoirement de manière séquentiel.

Opérations élémentaires – Recherche

Exemples de recherche

- recherche de x dans une liste L
- recherche de toutes les occurrences de x dans la liste L
  - on peut retourner une liste
- recherche de tous les éléments vérifiant une condition
  - retourner les éléments pairs
  - retourner les éléments positifs
  - retourner un élément sur trois

Opérations élémentaires – Insertion

Exemples d'insertion

- insersion en début de liste
- insertion en fin de liste
- insertion avant un nœud de valeur y
- insertion après un nœud de valeur y

Considérez tous les cas

- cas où la liste vide
- cas où y n'apparaît pas dans la liste

Opérations élémentaires – Suppression

Exemples de suppression

- suppression en début de liste
- suppression en fin de liste
- suppression de tous les nœuds contenant la valeur x
- suppression de tous les nœuds vérifiant une condition

Limitations des listes chaînées

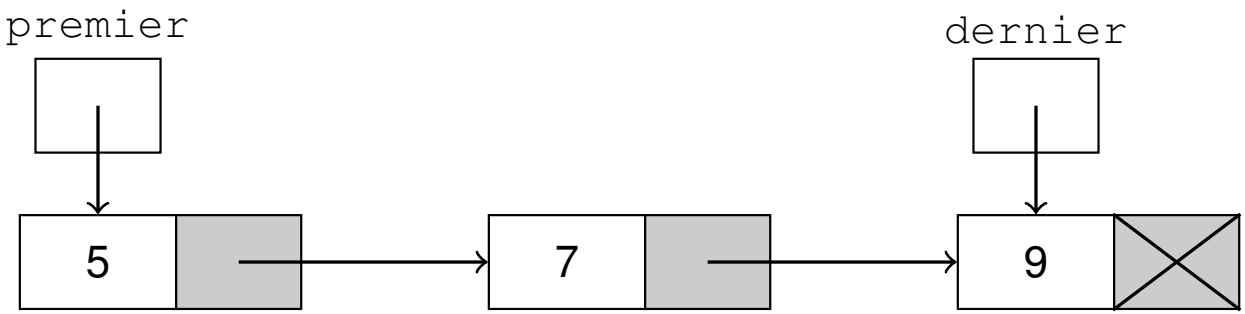
- un seul parcours possible
- on ne peut revenir en arrière
- certaines opérations sont coûteuses : insertion ou suppression en fin

Pour certaines utilisations

- il est possible de modifier la structure de liste chaînée

Plan du cours

Liste chaînée avec accès au premier et dernier nœud



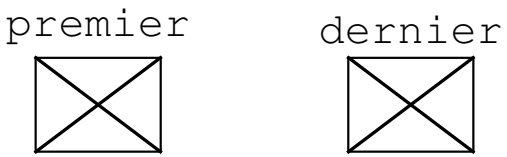
Structure de liste avec premier et dernier nœud

```
structure liste2{
   premier : pointeur sur noeud
   dernier : pointeur sur noeud
}
```

Liste chaînée avec accès au premier et dernier nœud

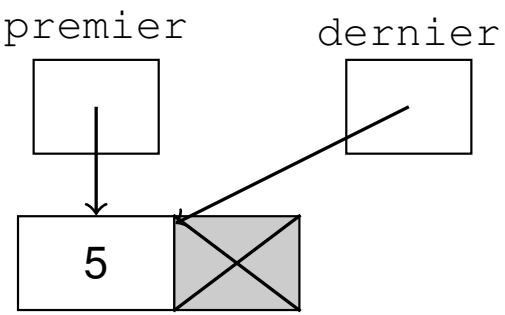
Liste vide

```
L : liste2
L.premier = None ; L.dernier = None
```



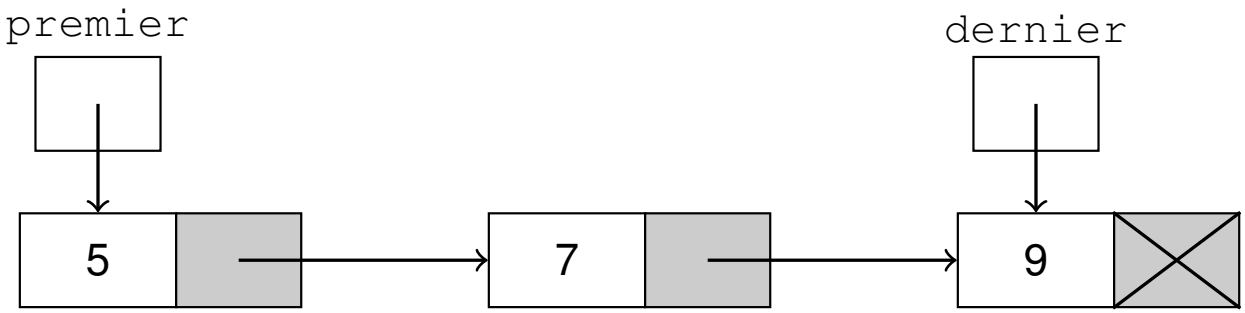
Liste avec un nœud

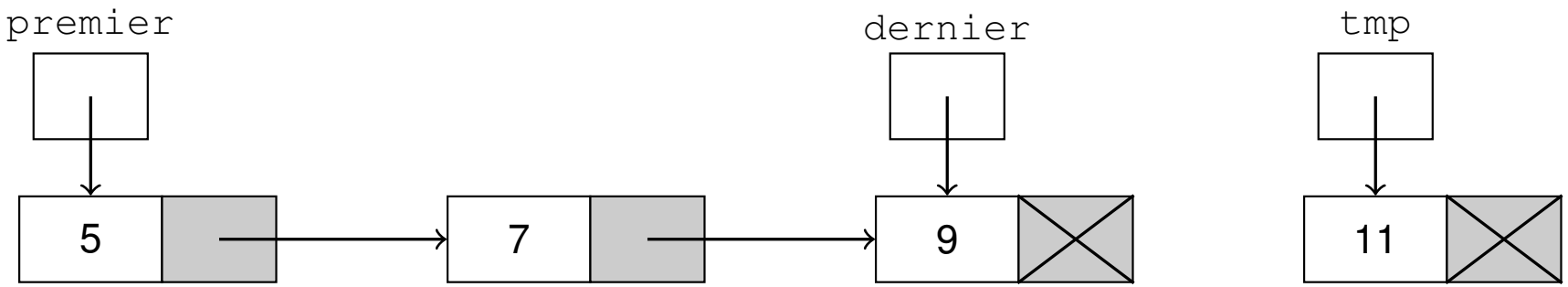
```
n : entier
L.premier = insertionDebut(L.premier,n)
L.dernier = L.premier
```



Liste chaînée avec accès au premier et dernier nœud

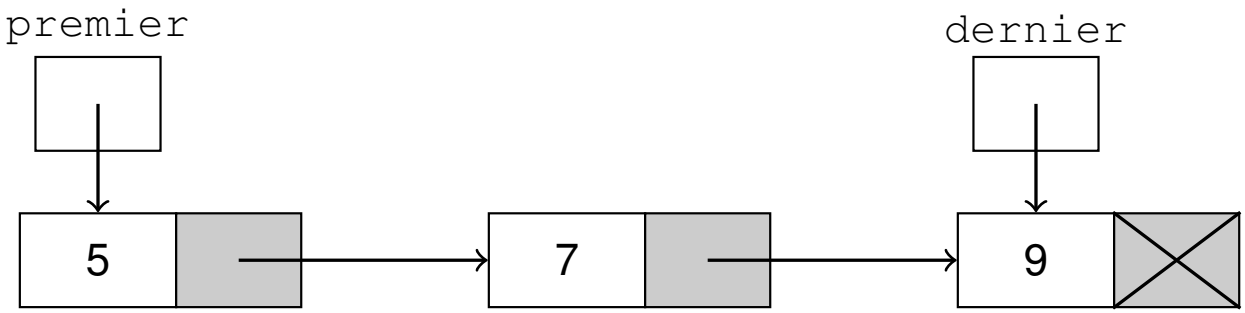
Insertion en fin de liste

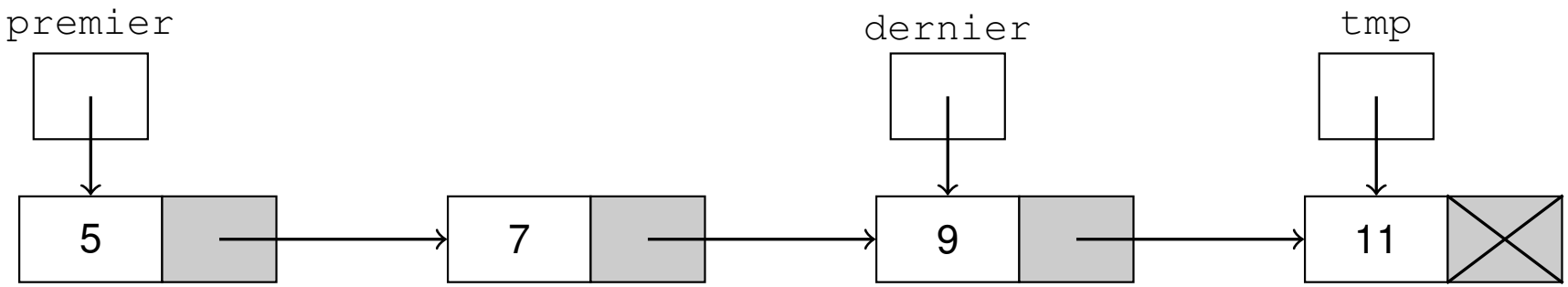




Liste chaînée avec accès au premier et dernier nœud

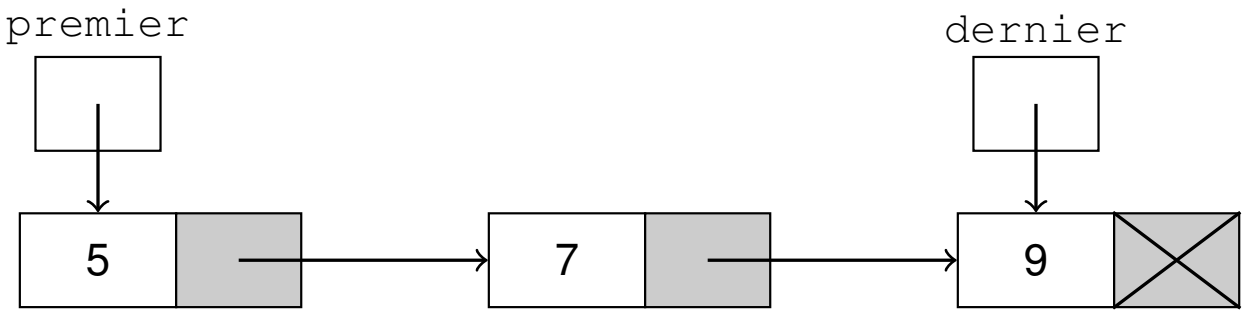
Insertion en fin de liste

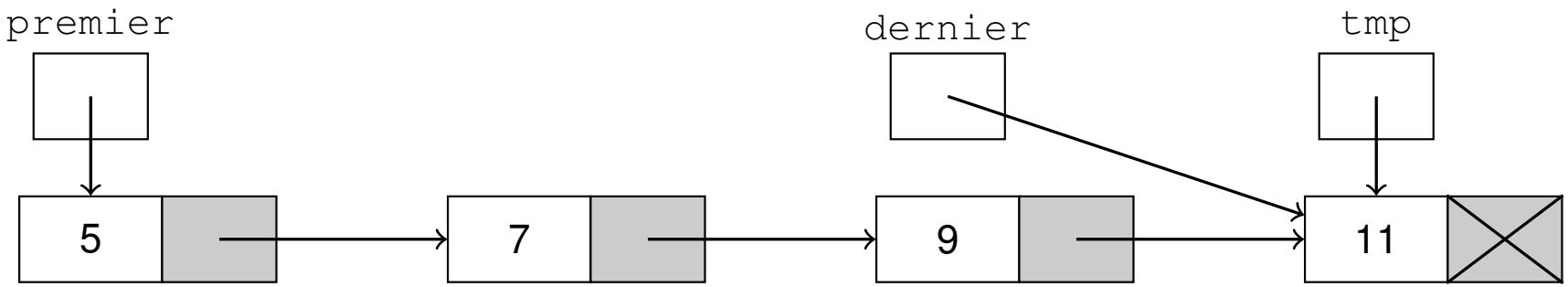




Liste chaînée avec accès au premier et dernier nœud

Insertion en fin de liste





Liste chaînée avec accès au premier et dernier nœud

Insertion en fin de liste

L'insertion se fait en temps constant (linéaire pour un seul pointeur)

```
insereEnFin(L : liste2, n : entier)
   tmp : pointeur sur noeud
   tmp = Nouveau(noeud);tmp->valeur = n;tmp->suivant = None    (1)
   si L.premier = None alors
      L.premier = tmp ; L.dernier = tmp
   sinon
        L.dernier->suivant = tmp                               (2)
        L.dernier = tmp                                        (3)
```

Remarque

Avec la structure de liste2, nous pouvons travailler directement sur la liste sans retourner la liste modifiée.

Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste

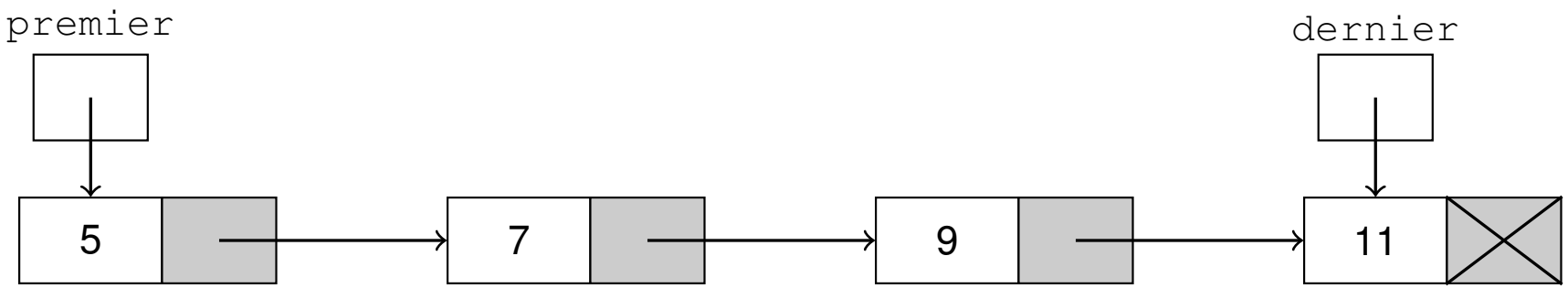
- la suppression se fait en temps linéaire
- car le parcours de la liste est inévitable

Algorithme

1. le pointeur tmp s'arrête au nœud juste avant le dernier nœud
2. on désalloue le dernier nœud
3. tmp devient le dernier nœud et L.dernier est égal à tmp

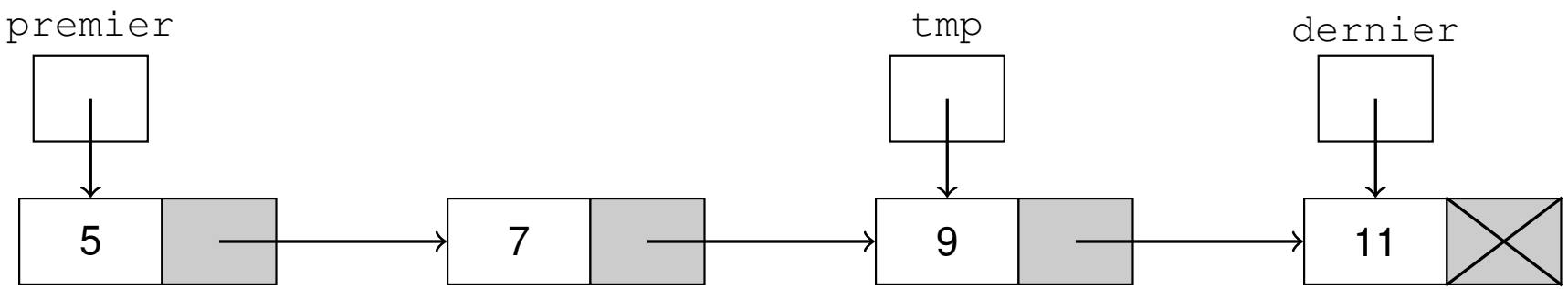
Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste



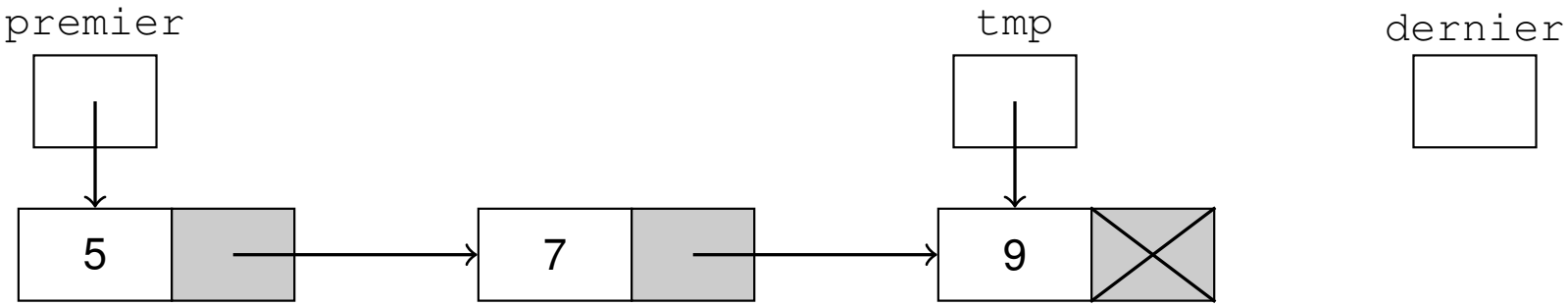
Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste (1)



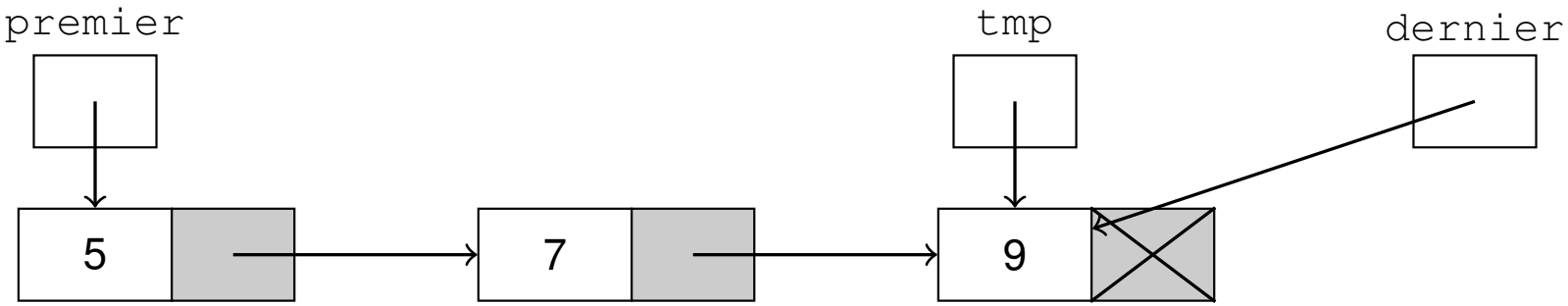
Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste (2)



Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste (3)



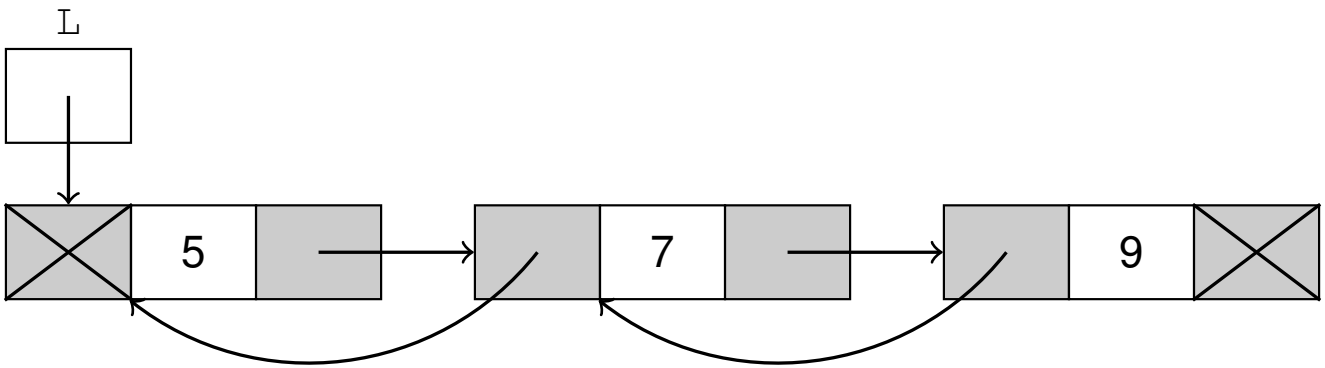
Liste chaînée avec accès au premier et dernier nœud

Suppression en fin de liste – algorithme

1. le pointeur tmp s'arrête au nœud juste avant le dernier nœud
2. on désalloue le dernier nœud
3. tmp devient le dernier nœud et L.dernier est égal à tmp

Suppression en fin de liste – Langage algorithmique

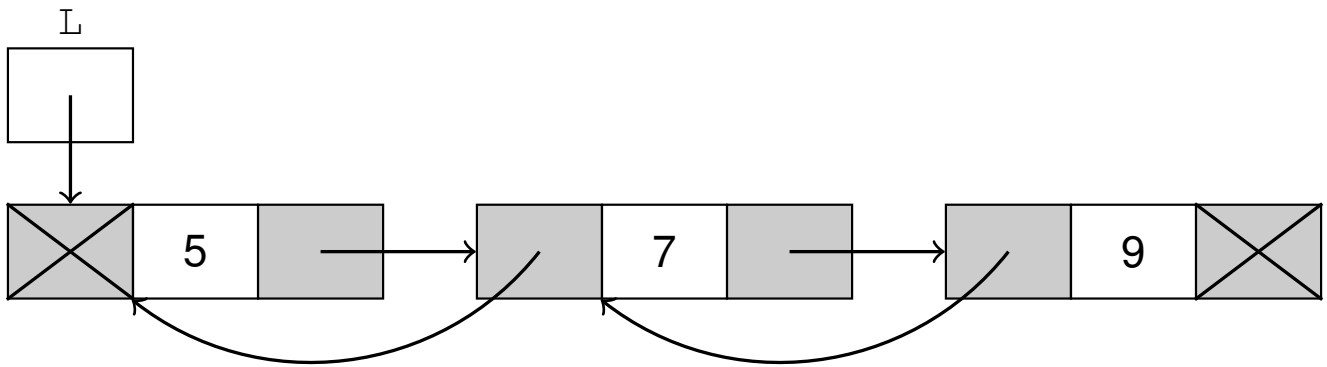
```
suppressionEnFin(L : liste2)
   si L.premier = L.dernier alors
      L.premier = None ; L.dernier = None
   tmp : pointeur sur noeud
   tant que tmp->suivant <> L.dernier faire                    (1)
      tmp = tmp->suivant
   désallouer tmp->suivant
   tmp->suivant = None                                         (2)
   L.dernier = tmp                                             (3)
```


L
5
7
9

Plan du CM3

Liste doublement chaînée

Exemple de liste doublement chaînée

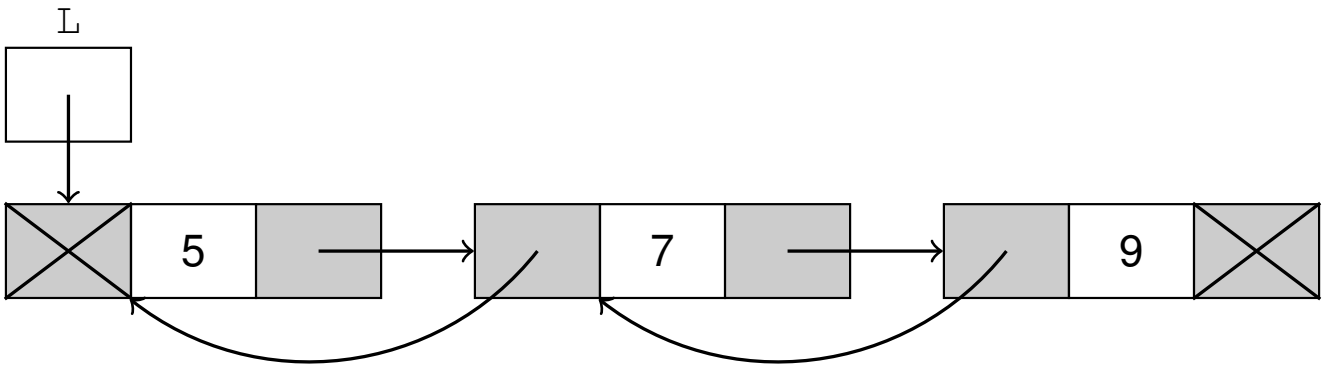


structure récursive nœudDouble

```
structure noeudDouble {
précédent : pointeur sur noeudDouble
valeur : entier
suivant : pointeur sur noeudDouble
}
type listeDouble = pointeur sur noeudDouble
```

Liste doublement chaînée

Exemple de liste doublement chaînée



Création de cette liste *à la main*

```
L : listedouble
L = Nouveau(noeudDouble) ; L->precedent = None ; L->valeur = 5
L->suivant =  Nouveau(noeudDouble)
L->suivant->precedent = L
L->suivant->valeur = 7
L->suivant->suivant = Nouveau(noeudDouble)
L->suivant->suivant->precedent = L->suivant
L->suivant->suivant->valeur = 9
L->suivant->suivant->suivant = None
```

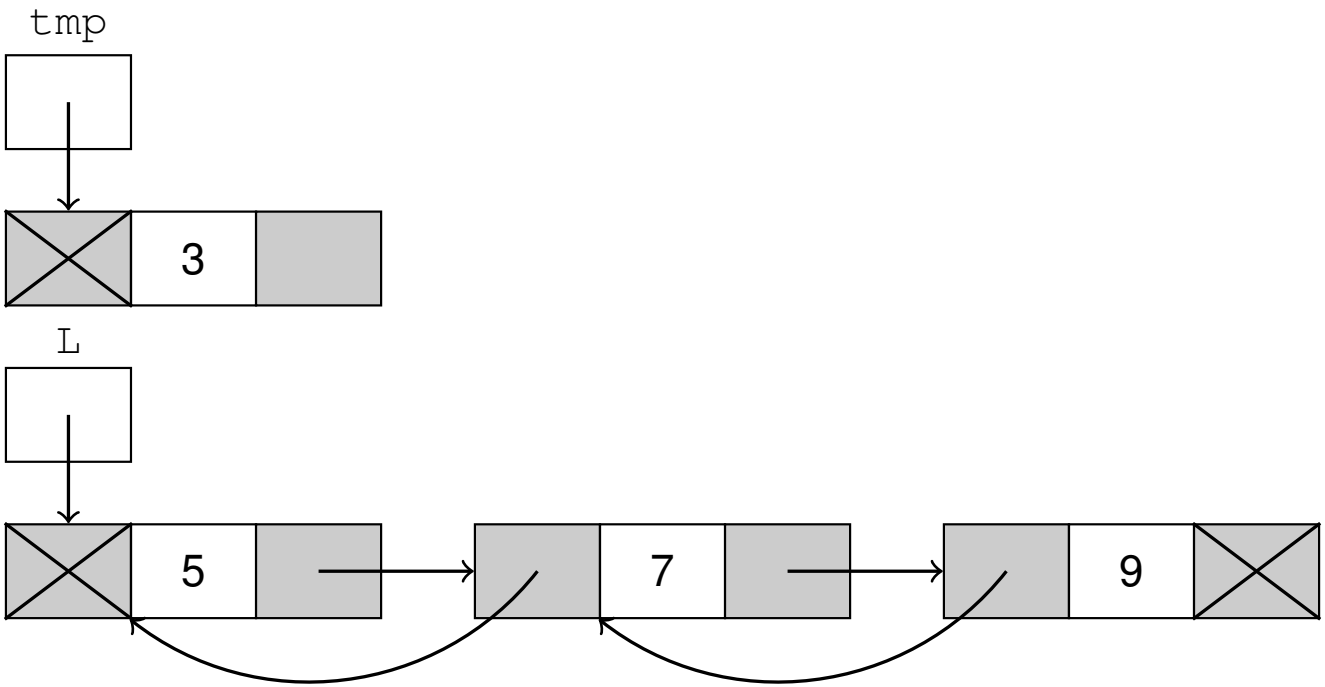
Liste doublement chaînée

Insertion au début

```
insertionDouble(L : listedouble, n : entier) : listedouble
   tmp : pointeur sur noeudDouble
   tmp = Nouveau(noeudDouble)
   tmp->precedent=None;tmp->suivant=None;tmp->valeur=n      (1)
   si L = None alors retourner tmp
   sinon
         tmp->suivant = L                                   (2)
         L->precedent = tmp                                 (3)
         retourner tmp                                      (4)
```

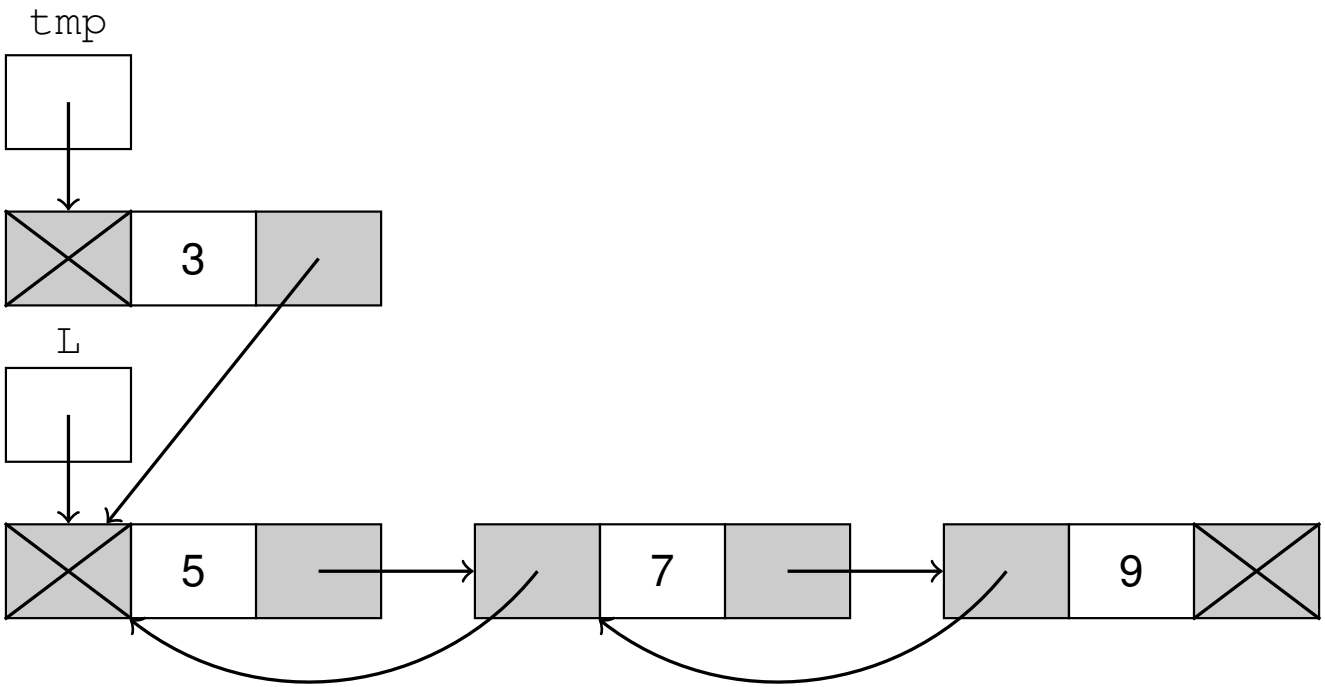
Liste doublement chaînée

Insertion au début – Étape 1



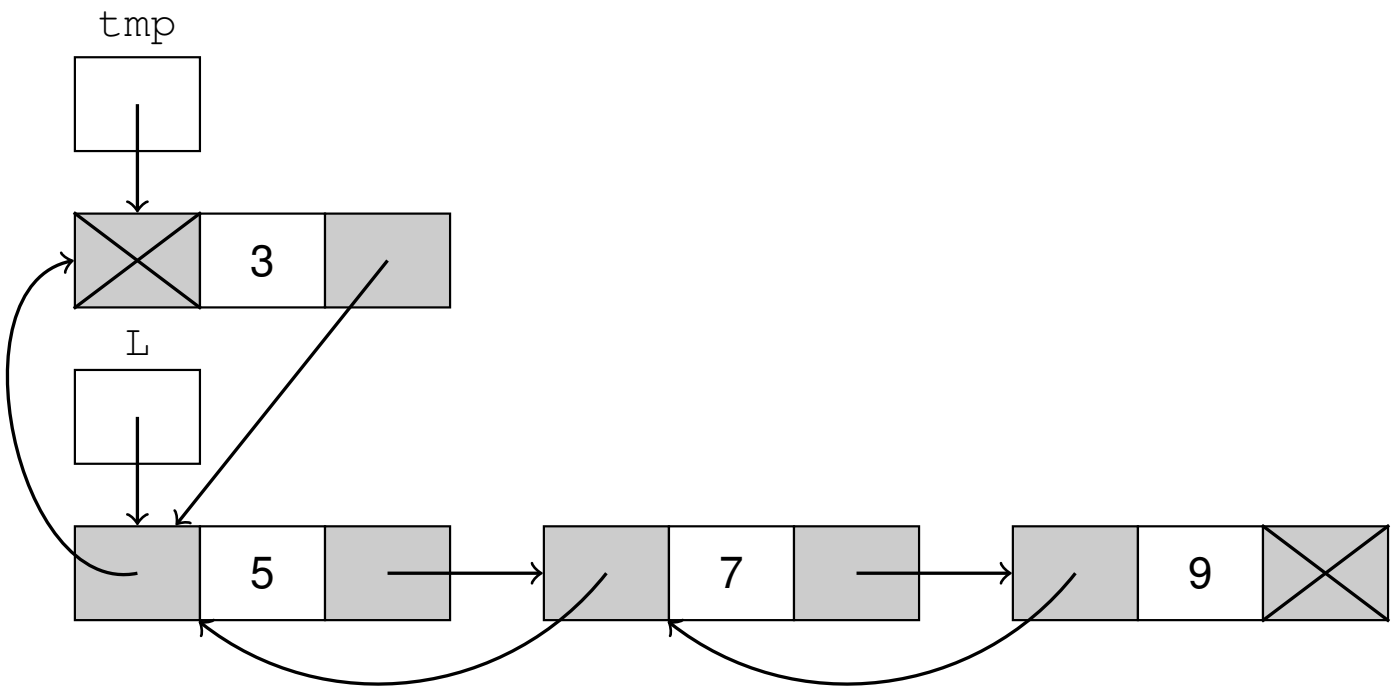
Liste doublement chaînée

Insertion au début – Étape 2



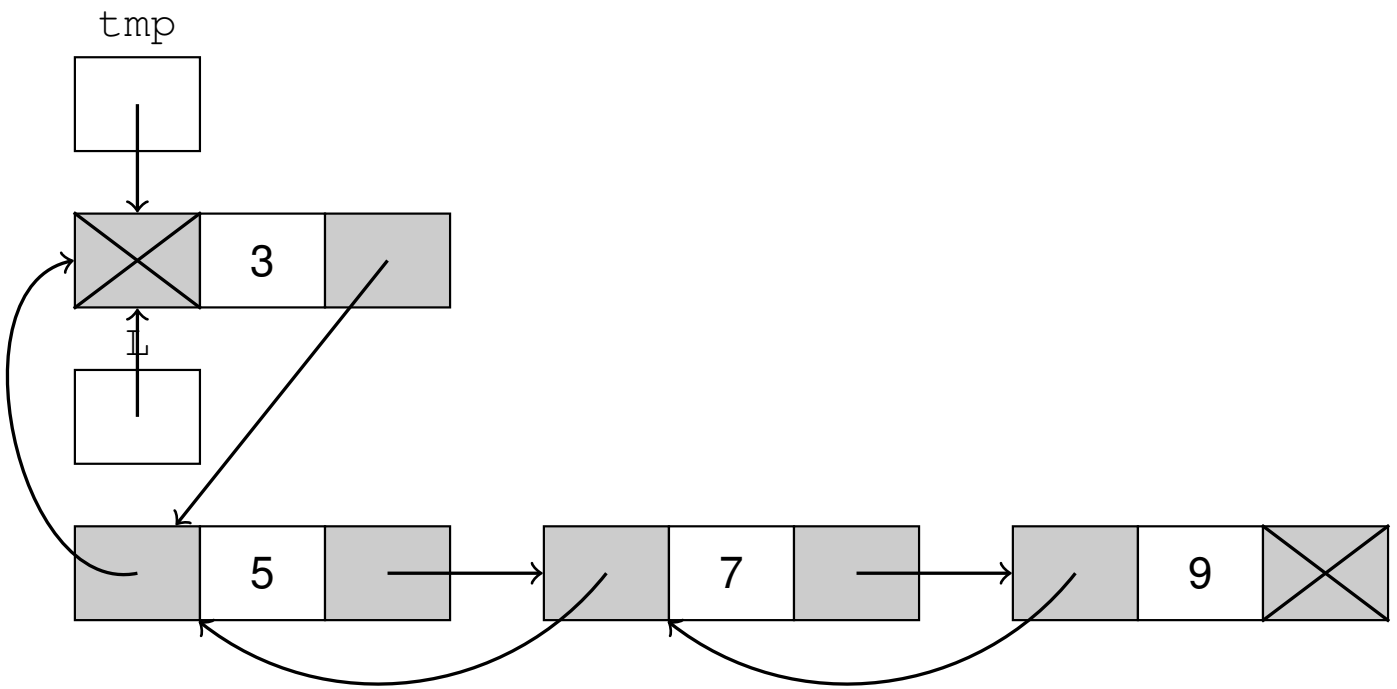
Liste doublement chaînée

Insertion au début – Étape 3



Liste doublement chaînée

Insertion au début – Étape 4

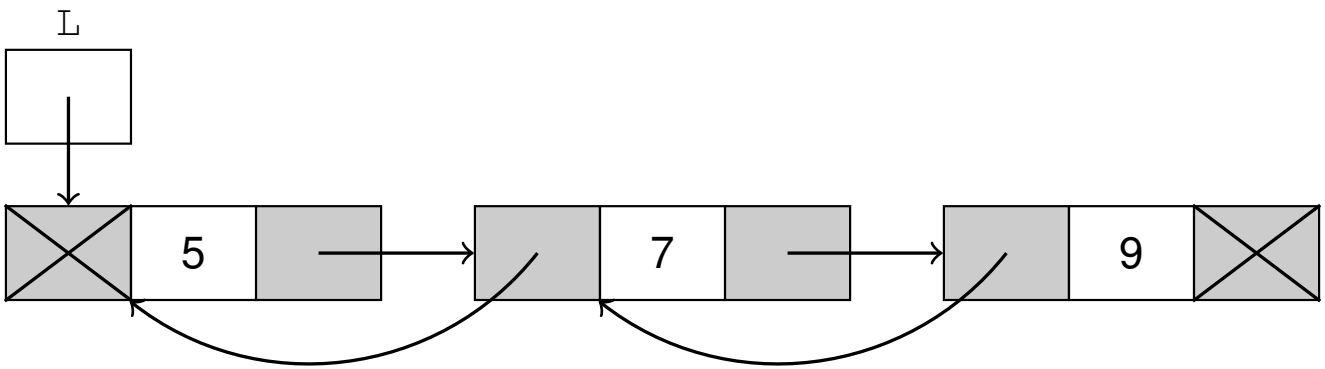


Liste doublement chaînée

Suppression au début

```
suppressionDouble(L : listedouble) : listedouble
   si L = None alors
      retourner None
   si L->suivant = None alors
      desallouer L
      retourner None
   sinon
       tmp : pointeur sur noeudDouble          (1)
       tmp = L->suivant                        (1)
       desallouer L                            (2)
       tmp->precedent = None                   (3)
       retourner tmp                           (4)
```

Exemple de liste

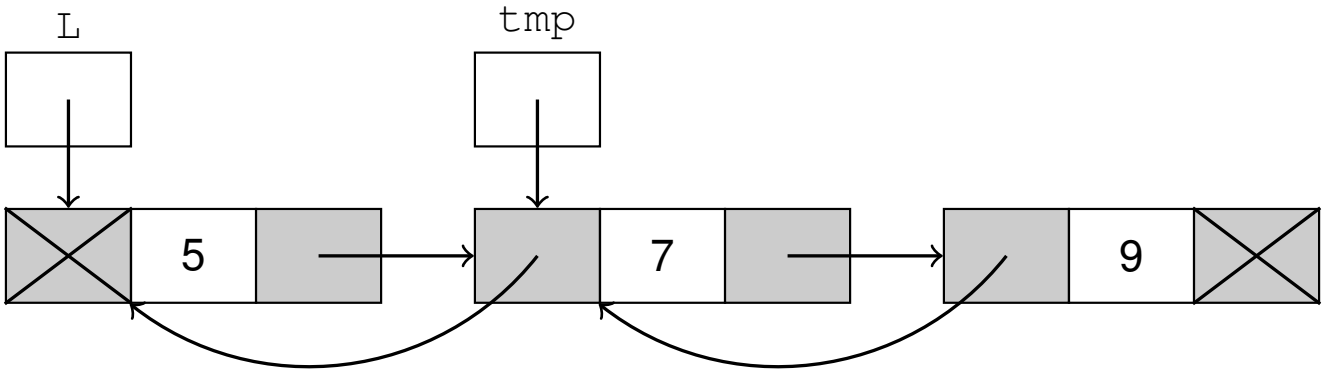


Liste doublement chaînée

Suppression au début

```
suppressionDouble(L : listedouble) : listedouble
   si L = None alors
      retourner None
   si L->suivant = None alors
      desallouer L
      retourner None
   sinon
       tmp : pointeur sur noeudDouble          (1)
       tmp = L->suivant                        (1)
       desallouer L                            (2)
       tmp->precedent = None                   (3)
       retourner tmp                           (4)
```

Exemple de liste – étape 1

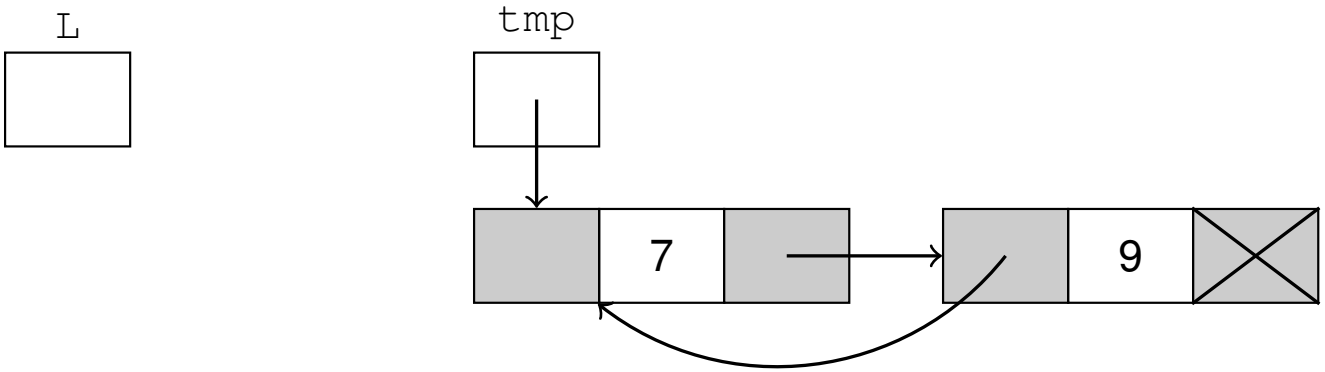


Liste doublement chaînée

Suppression au début

```
suppressionDouble(L : listedouble) : listedouble
   si L = None alors
      retourner None
   si L->suivant = None alors
      desallouer L
      retourner None
   sinon
       tmp : pointeur sur noeudDouble          (1)
       tmp = L->suivant                        (1)
       desallouer L                            (2)
       tmp->precedent = None                   (3)
       retourner tmp                           (4)
```

Exemple de liste – étape 2

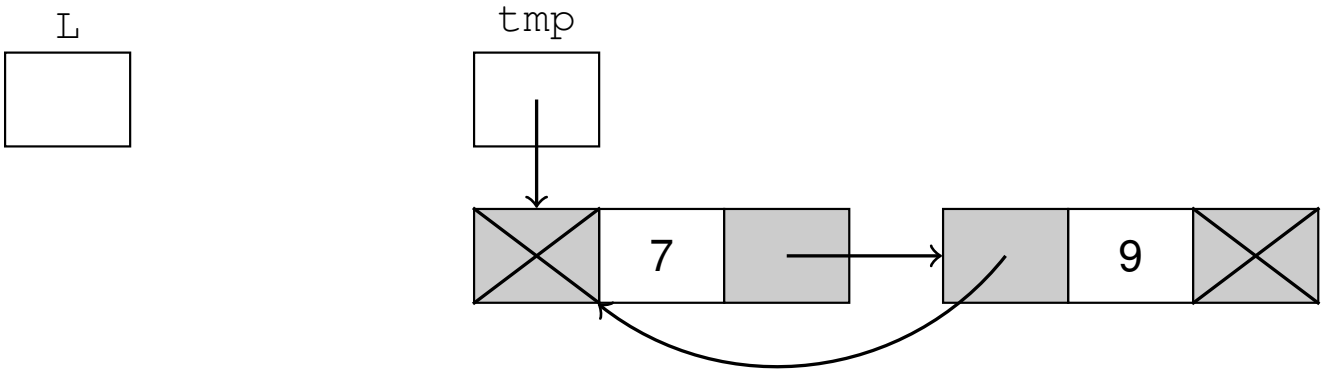


Liste doublement chaînée

Suppression au début

```
suppressionDouble(L : listedouble) : listedouble
   si L = None alors
      retourner None
   si L->suivant = None alors
      desallouer L
      retourner None
   sinon
       tmp : pointeur sur noeudDouble          (1)
       tmp = L->suivant                        (1)
       desallouer L                            (2)
       tmp->precedent = None                   (3)
       retourner tmp                           (4)
```

Exemple de liste – étape 3

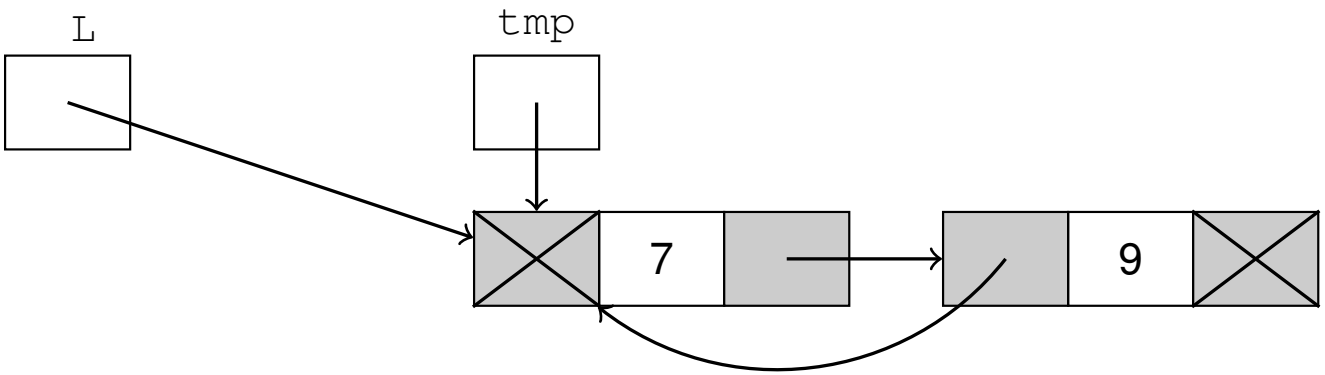


Liste doublement chaînée

Suppression au début

```
suppressionDouble(L : listedouble) : listedouble
   si L = None alors
      retourner None
   si L->suivant = None alors
      desallouer L
      retourner None
   sinon
       tmp : pointeur sur noeudDouble          (1)
       tmp = L->suivant                        (1)
       desallouer L                            (2)
       tmp->precedent = None                   (3)
       retourner tmp                           (4)
```

Exemple de liste – étape 4



Plan du CM3

Liste chaînée circulaire

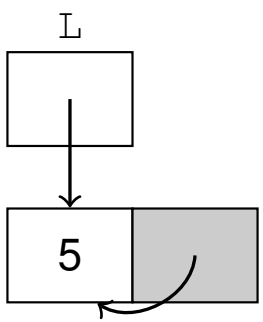
Structure de la liste

- le dernier nœud pointe sur le premier nœud
- le pointeur se fait sur le dernier nœud pour réaliser l'insertion et la suppression en temps constant.

Création d'une liste à un nœud

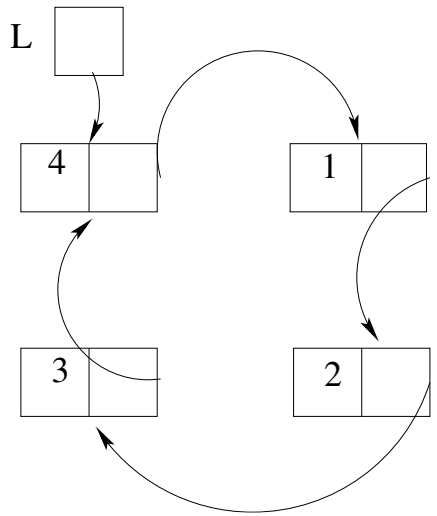
```
L : liste
n : entier
L = Nouveau(noeud)
L->valeur = n
L->suivant = L
```

Exemple de liste avec un seul nœud



Insertion

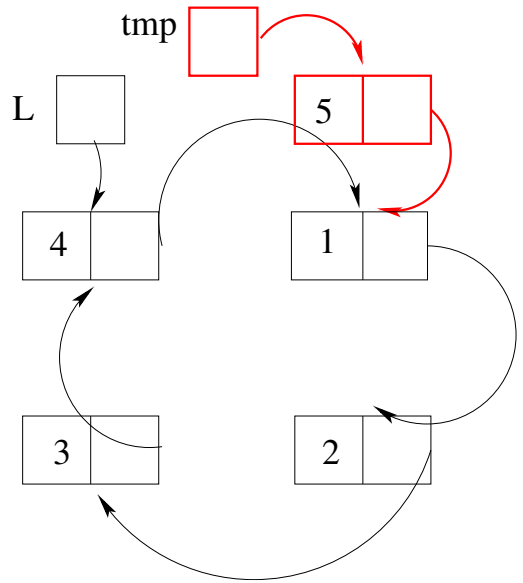
Insertion au début dans une liste circulaire




```
insertionCirculaire(L : liste, n  : entier) : liste
   si L = None alors
      L=Nouveau(noeud);L->valeur=n;L->suivant=L
   sinon tmp : pointeur sur noeud
          tmp=Nouveau(noeud);tmp->valeur=n;tmp->suivant=L->suivant
          L->suivant = tmp
   retourner tmp
```

Insertion

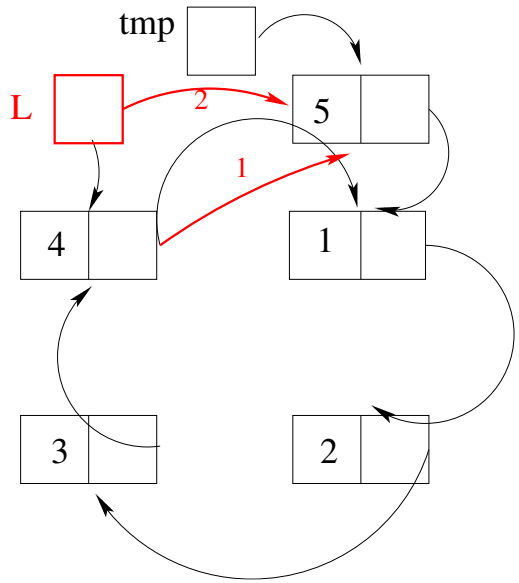
Insertion au début dans une liste circulaire




```
insertionCirculaire(L : liste, n  : entier) : liste
   si L = None alors
      L=Nouveau(noeud);L->valeur=n;L->suivant=L
   sinon  tmp : pointeur sur noeud
          tmp=Nouveau(noeud);tmp->valeur=n;tmp->suivant=L->suivant
          L->suivant = tmp
   retourner tmp
```

Insertion

Insertion en fin de liste dans une liste circulaire




```
insertionCirculaire(L : liste, n  : entier) : liste
   si L = None alors
      L=Nouveau(noeud);L->valeur=n;L->suivant=L
   sinon  tmp : pointeur sur noeud
          tmp=Nouveau(noeud);tmp->valeur=n;tmp->suivant=L->suivant
          L->suivant = tmp                          (1)
   retourner tmp                                    (2)
```

L'insertion s'effectue en temps constant

Suppression d'un nœud d'une liste circulaire

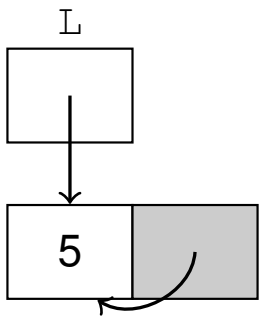
Suppression en début de liste

```
suppressionCirculaire(L : liste) : liste
   si L = None alors
      retourner None
   si L = L->suivant alors
      desallouer L
      retourner None
   L->suivant = L->suivant->suivant
   retourner L
```

La suppression en début de liste s'effectue en temps constant.

Suppression d'un nœud d'une liste circulaire

Exemple de liste avec un seul nœud



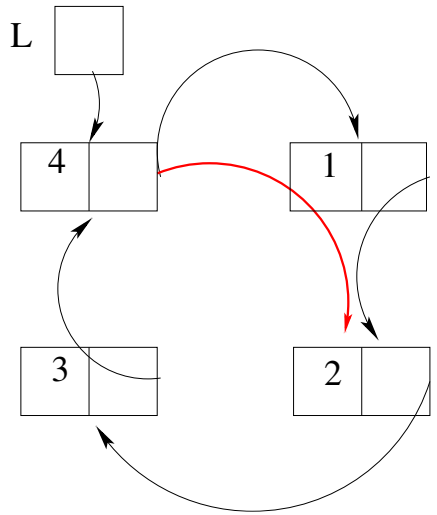
Suppression en début de liste

```
suppressionCirculaire(L : liste) : liste
    si L = None alors
        retourner None
    si L = L->suivant alors
        desallouer L
        retourner None
```

La liste est vide ou contient un seul nœud.

Suppression d'un nœud d'une liste circulaire

Exemple avec plusieurs nœuds



Suppression en début de liste

```
suppressionCirculaire(L : liste) : liste
    si L = None alors
        retourner None
    si L = L->suivant alors
        desallouer L
        retourner None
    L->suivant = L->suivant->suivant
    retourner L
```

La suppression en début de liste s'effectue en temps constant.

Suppression en fin de liste

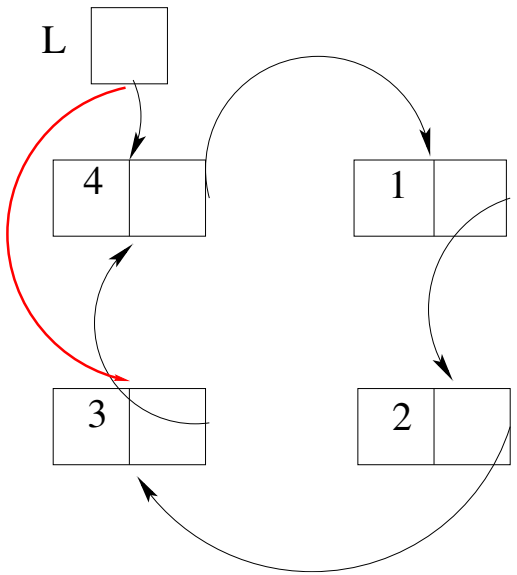
Pour une liste vide ou avec un nœud

Même instruction que précédemment :

```
si L = None alors
   retourner None
si L = L->suivant alors
   desallouer L
   retourner None
```

Suppression en fin de liste

Exemple avec plusieurs nœuds

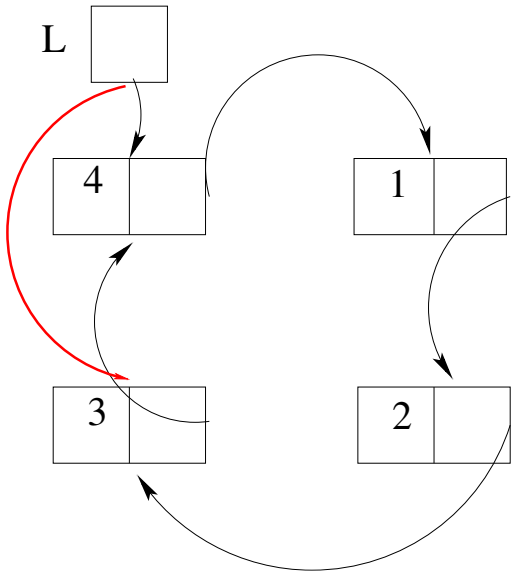


Suppression en fin de liste

La suppression en fin de liste s'effectue en temps linéaire (nombre de nœuds), car elle nécessite de parcourir complétement la liste.

Suppression en fin de liste

Exemple avec plusieurs nœuds




```
tmp : pointeur sur noeud ; tmp = L->suivant
tant que tmp->suivant <> L faire                    (1)
    tmp = tmp->suivant
tmp->suivant = L->suivant                           (2)
desallouer L
retourner tmp                                       (3)
```

Bilan

Choix de la structure de donnée

- il faut connaître les opérations qui seront les plus utilisées
- éviter les opérations coûteuses

Exemples d'utilisation

- piles
- files
- files avec ajout en début et en fin