Listes chaînées

Enrica Duchi

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

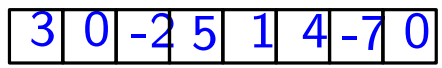
Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Pour le moment nous avons vu des tableaux.



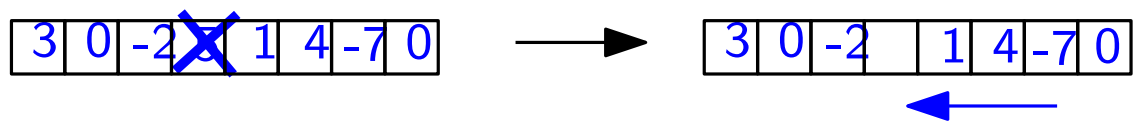
Avantage: accès en temps constant à un élément du tableau.

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Pour le moment nous avons vu des tableaux.



Avantage: accès en temps constant à un élément du tableau.

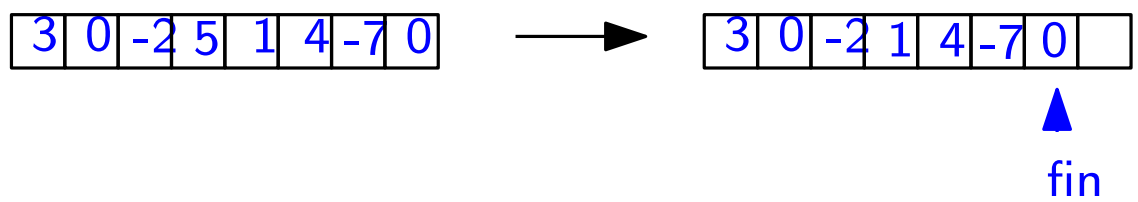
Désavantage: longueur du tableau fixée. Difficile de gérer l'élimination d'un élément.

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Pour le moment nous avons vu des tableaux.



Avantage: accès en temps constant à un élément du tableau.

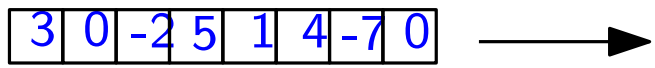
Désavantage: longueur du tableau fixée. Difficile de gérer l'élimination d'un élément.

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Pour le moment nous avons vu des tableaux.



Avantage: accès en temps constant à un élément du tableau.

Désavantage: longueur du tableau fixée. Difficile de gérer l'élimination d'un élément.

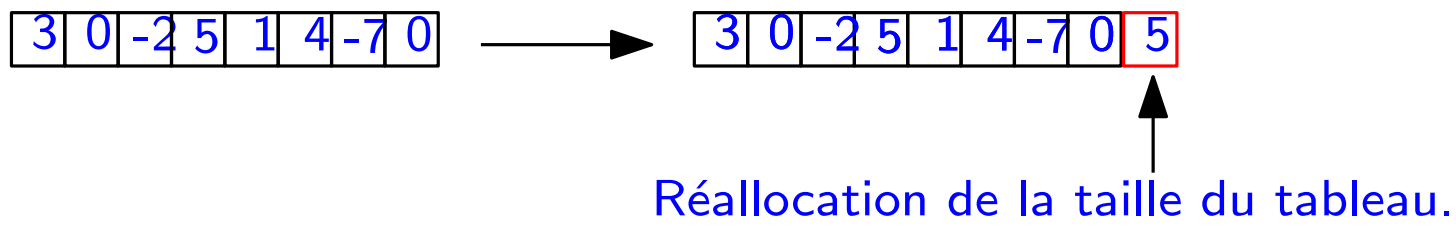
Difficile de gérer l'ajout d'un élément.

Listes chainées

Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Pour le moment nous avons vu des tableaux.



Avantage: accès en temps constant à un élément du tableau.

Désavantage: longueur du tableau fixée. Difficile de gérer l'élimination d'un élément.

Difficile de gérer l'ajout d'un élément.

Listes chainées

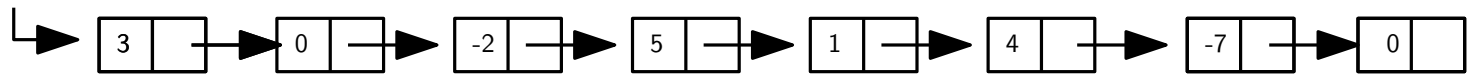
Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Représentation d'une liste par une liste chaînée.

Liste chainée: structure de données de même type de taille arbitraire, dont la représentation en mémoire est une ensemble de cellules avec un contenu et un pointeur vers la cellule suivante.

Tête de liste



Listes chainées

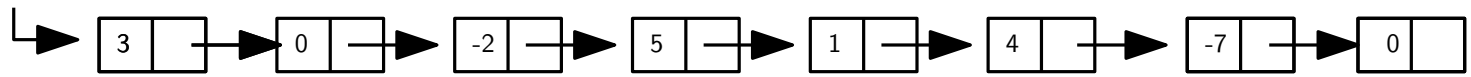
Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Représentation d'une liste par une liste chaînée.

Liste chainée: structure de données de même type de taille arbitraire, dont la représentation en mémoire est une ensemble de cellules avec un contenu et un pointeur vers la cellule suivante.

Tête de liste



Avantages: Taille flexible, ajout et suppression en tête de liste en temps constant.

Désavantages: l'accès au i-ème élément de la liste nécessite de parcourir les éléments qui le précèdent.

Listes chainées

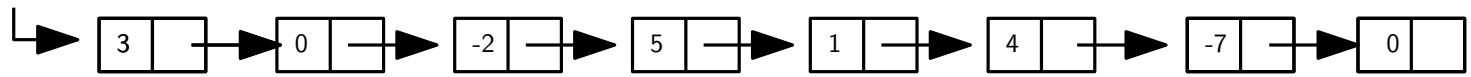
Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Représentation d'une liste par une liste chaînée.

Liste chainée: structure de données de même type de taille arbitraire, dont la représentation en mémoire est une ensemble de cellules avec un contenu et un pointeur vers la cellule suivante.

Tête de liste



Exemple:Suppression du cinquième élément de la liste

Listes chainées

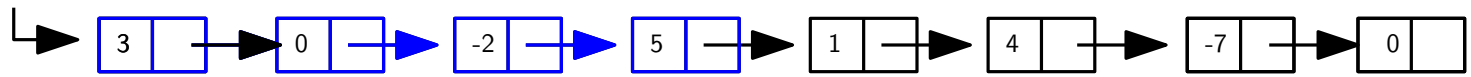
Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Représentation d'une liste par une liste chaînée.

Liste chainée: structure de données de même type de taille arbitraire, dont la représentation en mémoire est une ensemble de cellules avec un contenu et un pointeur vers la cellule suivante.

Tête de liste



Exemple:Suppression du cinquième élément de la liste

- parcourir la liste juqu'à la 4-ème cellule.

Listes chainées

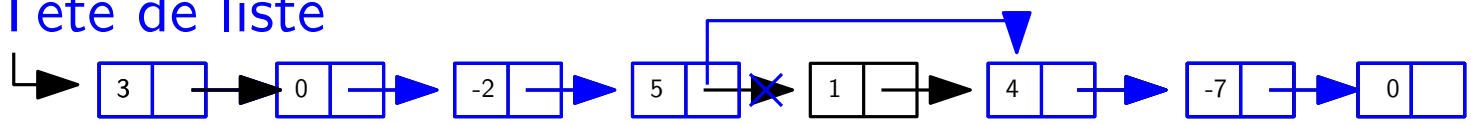
Nous avons besoin d'une structure de données pour représenter une collection ordonnée d'objets du même type.

Exemple: 3, 0, -2, 5, 1, 4, -7, 0 est une liste d'entiers.

Représentation d'une liste par une liste chaînée.

Liste chainée: structure de données de même type de taille arbitraire, dont la représentation en mémoire est une ensemble de cellules avec un contenu et un pointeur vers la cellule suivante.

Tête de liste



Exemple:Suppression du cinquième élément de la liste

- parcourir la liste juqu'à la 4-ème cellule.
- changer la cellule pointeé par la 4-ème cellule.

- Remarque: la 5-ème cellule n'est plus référencée par une autre cellule. Pour certains langages libérer la mémoire à la main. Pas besoin pour java.

Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
  c=L.head
  while c!= null{
     if c.key == x
        return c
     c=c.next
  }
  return c
}
```

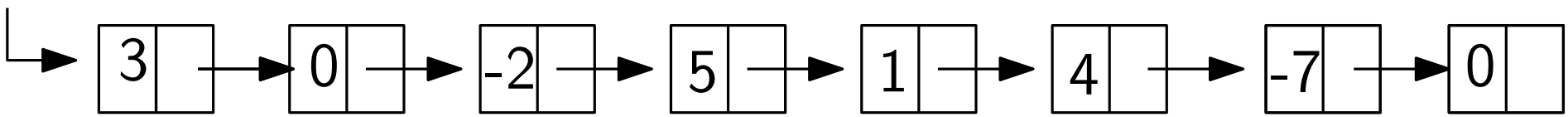
cellule=(c.key,c.next)

L.head=pointeur vers la première cellule de L.

Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
  c=L.head
  while c!= null{
     if c.key == x
        return c
     c=c.next
  }
  return c
}
```

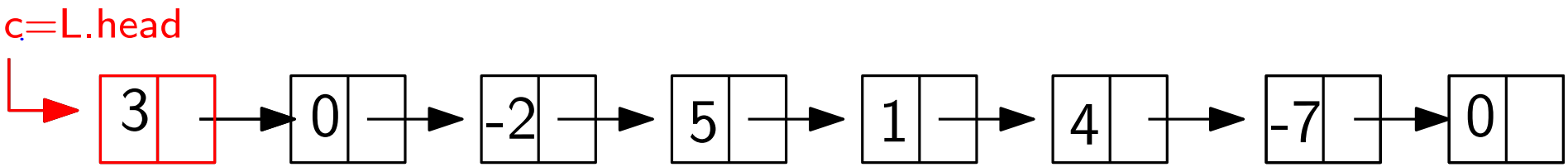
Exemple: search(L,5)



Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
  c=L.head
  while c!= null{
     if c.key == x
        return c
     c=c.next
  }
  return c
}
```

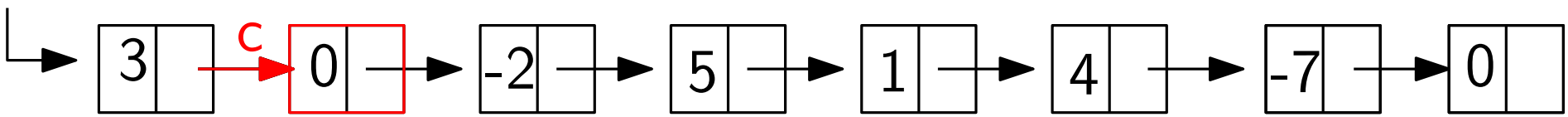
Exemple: search(L,5)



Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
   c=L.head
   while c!= null{
      if c.key == x
         return c
      c=c.next
   }
   return c
}
```

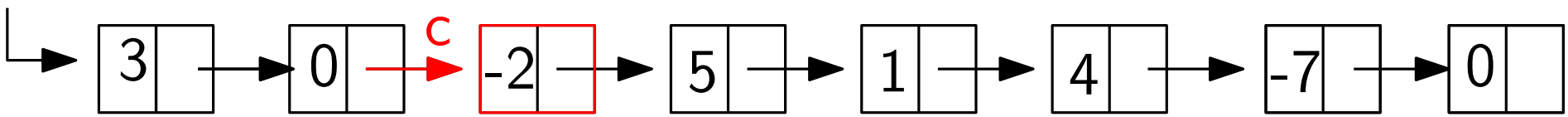
Exemple: search(L,5)



Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
   c=L.head
   while c!= null{
      if c.key == x
         return c
      c=c.next
   }
   return c
}
```

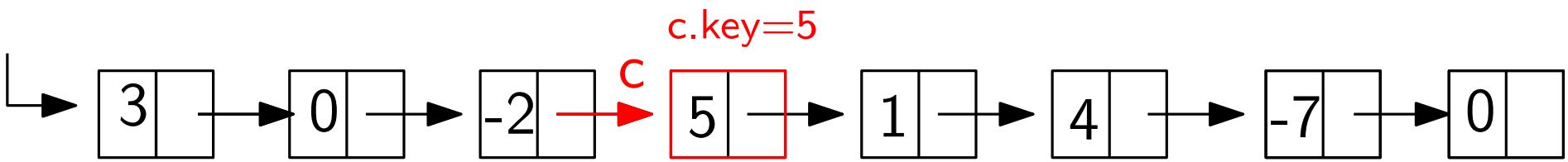
Exemple: search(L,5)



Recherche dans un élément x dans une liste L

```
search(L: list, x:  key){
   c=L.head
   while c!= null{
      if c.key == x
         return c
      c=c.next
   }
   return c
}
```

Exemple: search(L,5)



Recherche récursive d'un élément dans la liste L

```
search_Rec(L: list, x: key){
  return(search_Rec_aux(L.head,x)
}

search_Rec_aux(c: cell, x: key){
  if (c == null)
    return null
  if (c.key==x)
    return c
  else
    return search_Rec_aux(c.next,x)
}
```

cellule=(c.key,c.next)

L.head=pointeur vers la première cellule de L.

Insertion d'un élément x en tête de liste L

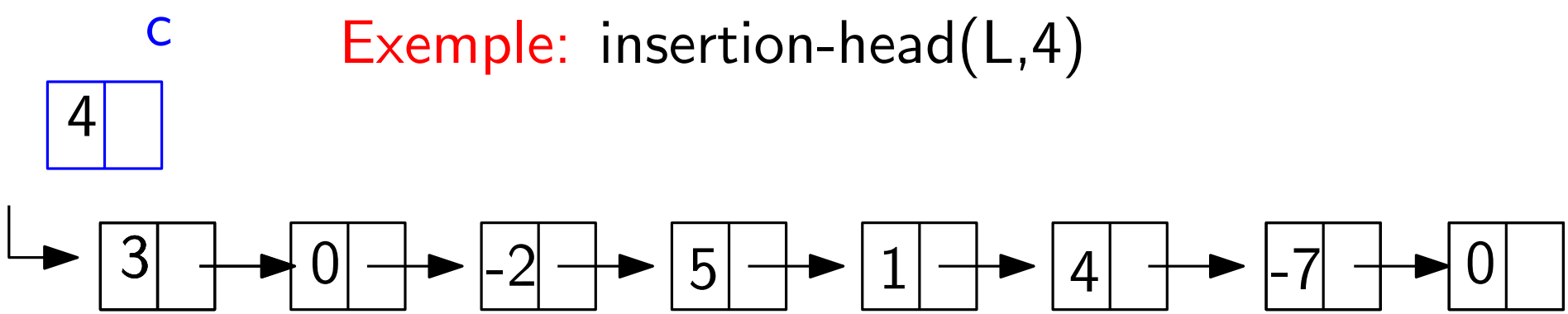
```
insertion_head(L: list, x: key){
  c= new cell()
  c.key=x
  c.next=L.head
  L.head=c
}
```

cellule=(c.key,c.next)

L.head=pointeur vers la première cellule de L.

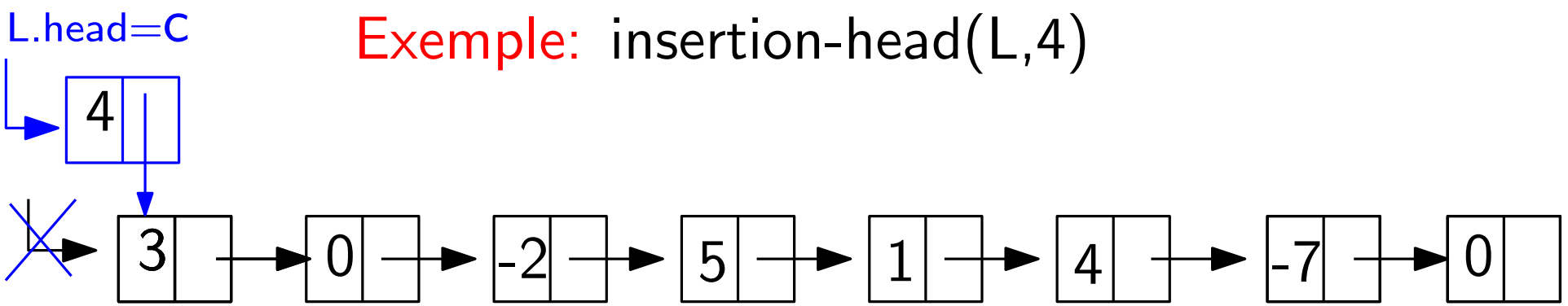
Insertion d'un élément x en tête de liste L

```
insertion_head(L: list, x: key){
  c= new cell()
  c.key=x
  c.next=L.head
  L.head=c
}
```



Insertion d'un élément x en tête de liste L

```
insertion_head(L: list, x: key){
   c= new cell()
   c.key=x
   c.next=L.head
   L.head=c
}
```



Exemple: insertion-head(L,4)

Suppression d'un élément en tête de liste L

```
remove_head(L: list){
   c=L.head
   if c != null{
      L.head = c.next
      }
   return c
}
```

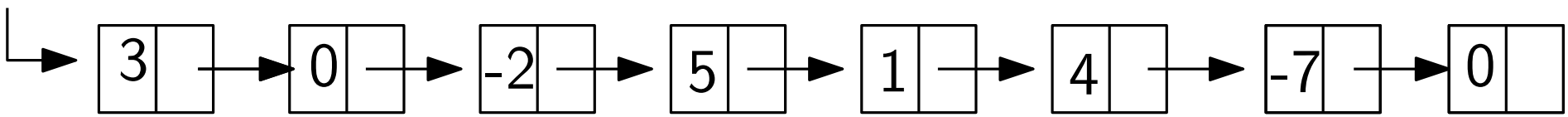
cellule=(c.key,c.next)

L.head=pointeur vers la première cellule de L.

Suppression d'un élément en tête de liste L

```
remove_head(L: list){
   c=L.head
   if c != null{
      L.head = c.next
      }
   return c
}
```

Exemple: remove-head(L)



Suppression d'un élément en tête de liste L

```
remove_head(L: list){
   c=L.head
   if c != null{
      L.head = c.next
      }
   return c
}
```

Exemple: remove-head(L)