Structure de Données
Liste

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

Semestre 3

Notes

Liste

- Une **liste chaînée** désigne une structure de données représentant une collection ordonnée et de taille arbitraire d'éléments
- L'accès aux éléments d'une liste se fait de manière séquentielle
  - chaque élément permet l'accès au suivant (contrairement au cas du tableau dans lequel l'accès se fait de manière absolue, par adressage direct de chaque cellule dudit tableau)
- **Un élément contient un accès vers une donnée**

Notes

Liste

Le principe de la liste chaînée est que chaque élément possède, en plus de la donnée, des pointeurs vers les éléments qui lui sont logiquement adjacents dans la liste

Opérations/syntaxe

- premier(L) : désigne le premier élément de la liste
- nil : désigne l'absence d'élément

Liste simplement chaînée

- donnée(elt) : désigne la donnée associée à l'élément elt
- suivant(elt) : désigne l'élément suivant elt

Notes

Liste

Le principe de la liste chaînée est que chaque élément possède, en plus de la donnée, des pointeurs vers les éléments qui lui sont logiquement adjacents dans la liste

Opérations/syntaxe

- premier(L) : désigne le premier élément de la liste
- nil : désigne l'absence d'élément

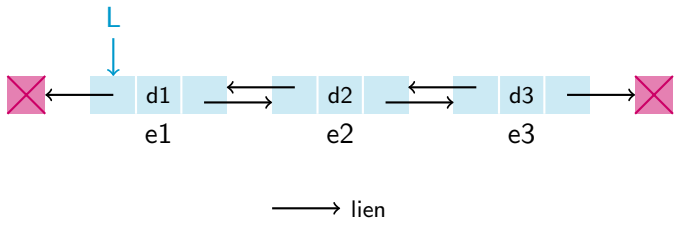
Liste doublement chaînée

- donnée(elt) désigne la donnée associée à l'élément elt
- suivant(elt) désigne l'élément suivant elt
- précédent(elt) désigne l'élément précédant elt

Notes

Liste doublement chaînée

Représentation



- premier(L) = e1
- donnée(e1) = d1, suivant(e1) = e2, précédent(e1) = nil
- donnée(e2) = d2, suivant(e2) = e3, précédent(e2) = e1
- donnée(e3) = d3, suivant(e3) = nil, précédent(e3) = e2

Notes

Liste

Trois opérations principales

- Parcours de la liste
- Ajout d'un élément
- Suppression d'un élément

À partir de là d'autres opérations vont être obtenues : recherche d'une donnée, remplacement, concaténation de liste, fusion de listes, ...

Notes

Insertion sous condition d'un élément

- Liste L doublement chaînée
- On veut insérer l'élément `elt` dans la liste avant le premier élément de la liste qui est associée à une donnée > 8

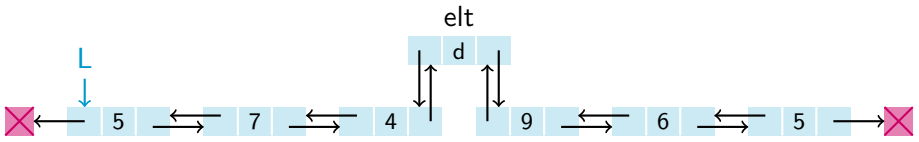
Exemple

Notes

Insertion sous condition d'un élément

- Liste L doublement chaînée
- On veut insérer l'élément `elt` dans la liste avant le premier élément de la liste qui est associée à une donnée > 8

Exemple



Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  tant que (e ≠ nil et donnée(e) <= val) {
    e <- suivant (e)
  }
  si (e = nil) {
    // on insère en fin
  } sinon {
    // on insère avant e
  }
}
```

Il faut connaître le dernier

Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  dernier <- e
  tant que (e ≠ nil et donnée(e) <= val) {
    dernier <- e
    e <- suivant (e)
  }
  si (e = nil) {
    // on insère en fin
    précédent(elt) <- dernier
    suivant(elt) <- nil
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
  }
}
```

Il faut tester avec le premier

Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  dernier <- e
  tant que (e ≠ nil et donnée(e) <= val) {
    dernier <- e
    e <- suivant (e)
  }
  si (e = nil) {
    // on insère en fin
    précédent(elt) <- dernier
    suivant(elt) <- nil
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(elt) <- prec
    suivant(elt) <- e
    précédent(e) <- elt
    si (prec = nil) {
      premier(L) <- elt
    } sinon {
      suivant(prec) <- elt
    }
  }
}
```


Notes

Listes avec sentinelles

- On introduit deux éléments "bidon", appelé sentinelles
  $\Rightarrow$ À la fois comme premier et comme dernier
- Ces éléments sont cachés
  - Le vrai premier est le suivant de la sentinelle
  - Le vrai dernier est le précédent de la sentinelle
- Cela évite les problèmes avec les tests avec la valeur nil, puisqu'il y a toujours un suivant ou un précédant pour les éléments visibles dans la liste

Notes

Listes avec sentinelles

Insertion **avant** e de elt

```
suivant(elt) <- e
précédent(elt) <- précédent(e)
suivant(précédent(e)) <- elt
précédent(e) <- elt
```

Insertion **après** e de elt

```
suivant(elt) <- suivant(e)
précédent(elt) <- e
précédent(suivant(e)) <- elt
suivant(e) <- elt
```

Marche toujours ! Plus besoin de tests !

Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  dernier <- e
  tant que (e ≠ sentinelle(L) et donnée(e) <= val) {
    dernier <- e
    e <- suivant (e)
  }
  si (e = sentinelle(L)) {
    // on insère en fin
    précédent(elt) <- dernier
    suivant(elt) <- sentinelle(L)
    précédent(sentinelle(L)) <- elt
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(elt) <- prec
    suivant(elt) <- e
    précédent(e) <- elt
    si (prec = sentinelle(L)) {
      premier(L) <- elt
    } sinon {
      suivant(prec) <- elt
    }
  }
}
```

Comme le dernier est le précédent de la sentinelle, on peut remplacer partout dernier par précédent(sentinelle (L))

Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  tant que (e ≠ sentinelle(L) et donnée(e) <= val) {
    e <- suivant (e)
  }
  si (e = sentinelle(L)) {
    // on insère en fin
    dernier <- précédent(sentinelle(L))
    précédent(elt) <- dernier
    suivant(elt) <- sentinelle(L)
    précédent(sentinelle(L)) <- elt
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(elt) <- prec
    suivant(elt) <- e
    précédent(e) <- elt
    si (prec = sentinelle(L)) {
      premier(L) <- elt
    } sinon {
      suivant(prec) <- elt
    }
  }
}
```

Comme le premier est le suivant de la sentinelle, on peut remplacer premier par suivant(sentinelle (L))

Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  tant que (e ≠ sentinelle(L) et donnée(e) <= val) {
    e <- suivant (e)
  }
  si (e = sentinelle(L)) {
    // on insère en fin
    dernier <- précédent(sentinelle(L))
    précédent(elt) <- dernier
    suivant(elt) <- sentinelle(L)
    précédent(sentinelle(L)) <- elt
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(elt) <- prec
    suivant(elt) <- e
    précédent(e) <- elt
    si (prec = sentinelle(L)) {
      suivant(sentinelle(L)) <- elt
    } sinon {
      suivant(prec) <- elt
    }
  }
}
```


Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  tant que (e ≠ sentinelle(L) et donnée(e) <= val) {
    e <- suivant (e)
  }
  si (e = sentinelle(L)) {
    // on insère en fin
    dernier <- précédent(sentinelle(L))
    précédent(elt) <- dernier
    suivant(elt) <- sentinelle(L)
    précédent(sentinelle(L)) <- elt
    suivant(dernier) <- elt
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(elt) <- prec
    suivant(elt) <- e
    précédent(e) <- elt
    suivant(prec) <- elt
  }
}
```


Notes

Insertion sous condition d'un élément

```
inserer(L,elt,val) {
  // on suppose que L n'est pas vide
  e <- premier(L)
  // on cherche la position
  tant que (e ≠ sentinelle(L) et donnée(e) <= val) {
    e <- suivant (e)
  }
  // on insère avant e
  prec <- précédent(e)
  précédent(elt) <- prec
  suivant(elt) <- e
  précédent(e) <- elt
  suivant(prec) <- elt
}
```


Notes