Structure de Données
Liste

Marie Pelleau
marie.pelleau@unice.fr

Semestre 3

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**

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

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

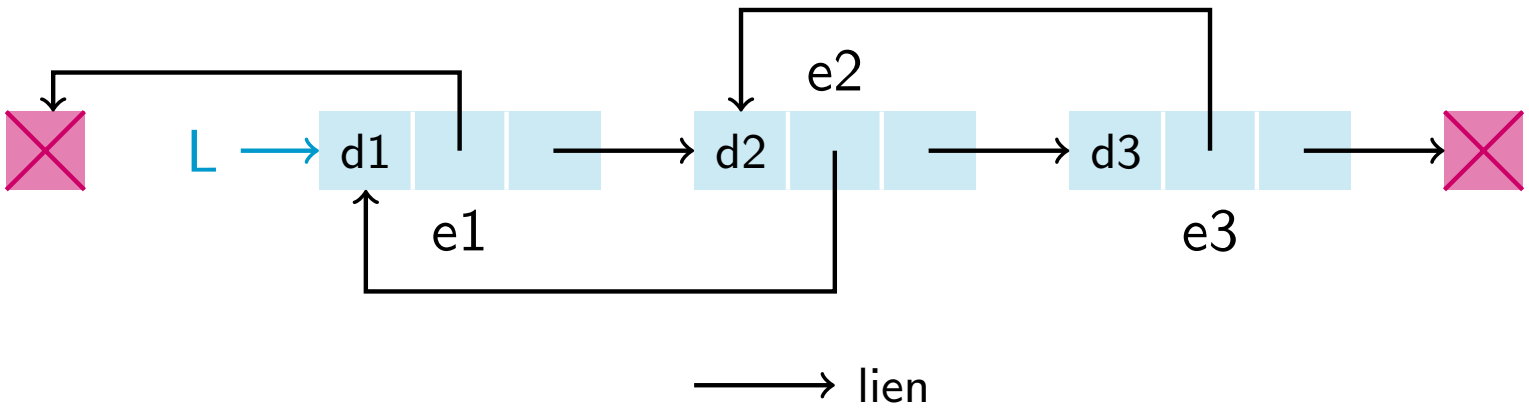
- premier(L) : désigne le premier élément de la liste
- nil : désigne l'absence d'élément

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

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

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, ...

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 : L : 5 7 4 9 6 5, on insère 5 7 4 elt 9 6 5

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 donnee(e) <= val) {
    e <- suivant (e)
  }
  if (e = nil) {
    // on insère en fin
  } sinon {
    // on insère avant e
  }
}
```

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 donnee(e) <= val) {
    dernier <- e
    e <- suivant (e)
  }
  if (e = nil) {
    // on insère en fin
    suivant(dernier) <- elt
    précédent(elt) <- dernier
    suivant(elt) <- nil
  } sinon {
    // on insère avant e
  }
}
```

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 donnee(e) <= val) {
    dernier <- e
    e <- suivant (e)
  }
  if (e = nil) {
    // on insère en fin
    suivant(dernier) <- elt
    précédent(elt) <- dernier
    suivant(elt) <- nil
  } sinon {
    // on insère avant e
    prec <- précédent(e)
    précédent(e) <- elt
    suivant(elt) <- e
    si (prec = nil) {
      premier(L) <- elt
    } sinon {
      suivant(prec) <- elt
    }
  }
}
```

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

Listes avec sentinelles

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 !