Algorithmique

Structures Et Listes linéaires

Maria Malek

Département Informatique

Les structures - Définition

- Une structure est un type complexe défini par l'utilisateur, composé d'un ensemble de champs de différents types :

Les structures - Définition

- Une structure est un type complexe défini par l'utilisateur, composé d'un ensemble de champs de différents types :
- La syntaxe est :

  type NomType = structure

  - *NomChamps1 : T1*
  - *NomChamps2 : T2*
  - *...*
  - *NomChampsN : TN*

Les structures - Définition

- Une structure est un type complexe défini par l'utilisateur, composé d'un ensemble de champs de différents types :
- La syntaxe est :

  type NomType = structure

  - *NomChamps1 : T1*
  - *NomChamps2 : T2*
  - *...*
  - *NomChampsN : TN*
- où T1,T2,..,TN sont des types de bases ou définis

Les structures - Exemples - 1

- Structure à partir de types simples

  type eleve = structure

  - *nom : chaine*
  - *prenom : chaine*
  - *nSS : chaine*

- Structure contenant des types complexes : associer à chaque élève son numéro d'étudiant ainsi que l'ensemble des matières, etc.

  type etudiant = structure

  - *identite: eleve*
  - *numero : entier*
  - *annee : entier*
  - *matieres : tableau(N) de chaines*
  - *nbrMatieres : entier*

Les structures - Exemples - 2

- Structure récursif : associer à chaque élève son parrain :
  type eleve = structure
  - nom : chaine
  - prenom : chaine
  - nSS : chaine
  - parrain : eleve

Les structures - Déclaration& accès

- type NomType = structure
  - NomChamps1 : T1
  - NomChamps2 : T2
  - ...
  - NomChampsN : TN

Les structures - Déclaration& accès

- type NomType = structure
  - NomChamps1 : T1
  - NomChamps2 : T2
  - ...
  - NomChampsN : TN
- Déclaration : **vaiables : NomVariable: NomType**

Les structures - Déclaration& accès

- type NomType = structure
  - NomChamps1 : T1
  - NomChamps2 : T2
  - ...
  - NomChampsN : TN
- Déclaration : **vaiables : NomVariable: NomType**
- Accès à un champs : **NomVariable.NomChampsI**

Les structures - Exemple d’algorithme

- Afficher les matières d’un étudiant donné :

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

  procedure afficherMatieres (E e : etudiant)

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier

$i \leftarrow 1$

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier
$i \leftarrow 1$
tantque $i <= e.nbrMatieres$ **faire**

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier
$i \leftarrow 1$
tantque $i <= e.nbrMatieres$ **faire**
ecrireln(e.matieres(i))

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier
$i \leftarrow 1$
tantque $i <= e.nbrMatieres$ **faire**
ecrireln(e.matieres(i))
$i \leftarrow i + 1$

Les structures - Exemple d'algorithme

- Afficher les matières d'un étudiant donné :

procedure afficherMatieres (E e : etudiant)
variables i : entier
$i \leftarrow 1$
tantque $i <= e.nbrMatieres$ **faire**
ecrireln(e.matieres(i))
$i \leftarrow i + 1$
fin tantque

Structure particulière - La cellule

- Une structure paramétrée par un type donné T permettant de contenir une information de type T.

Structure particulière - La cellule

- Une structure paramétrée par un type donné T permettant de contenir une information de type T.
- elle est recursive : elle contient un ou plusieurs champs permettant d’indiquer un lien vers d’autres cellules

Structure particulière - La cellule

- Une structure paramétrée par un type donné T permettant de contenir une information de type T.
- elle est recursive : elle contient un ou plusieurs champs permettant d'indiquer un lien vers d'autres cellules
- La syntaxe d'une cellule :

  type NomCellule de T = structure

  - *ChampsInfo : T*
  - *ChampsCellule1 : NomCellule de T*
  - *ChampsCellule2 : NomCellule de T*
  - *...*
  - *ChampsCelluleN : NomCellule de T*

Exemples d’utilisation d’une cellule

- Représentation d’une liste linéaire de longueur variable :
  type celluleLineaire de T = structure
  - info : T
  - suivant : celluleLineaire de T

Exemples d'utilisation d'une cellule

- Représentation d'une liste linéaire de longueur variable :
  type celluleLineaire de T = structure
  - info : T
  - suivant : celluleLineaire de T
- Représentation d'un arbre binaire :
  type celluleBinaire de T = structure
  - info : T
  - filsGauche : celluleBinaire de T
  - filsDroite : celluleBinaire de T

Les Primitives pré-définies des cellules

- Deux primitives :

Les Primitives pré-définies des cellules

- Deux primitives :

  nouveau(L) cree une cellule L. Cette cellule va contenir un élément de type T. Les champs récursifs sont affectés à *nil* par default.

Les Primitives pré-définies des cellules

- Deux primitives :

  nouveau(L) cree une cellule L. Cette cellule va contenir un élément de type T. Les champs récursifs sont affectés à *nil* par default.

  laisser(L) détruit la cellule L

Les Primitives pré-définies des cellules

- Deux primitives :

  nouveau(L) cree une cellule L. Cette cellule va contenir un élément de type T. Les champs récursifs sont affectés à *nil* par default.

  laisser(L) détruit la cellule L

- la valeur *nil* affectée à une variable cellule signifie *qu'il a rien* dans la cellule.

Listes linéaires - Manipulation

- Création d’une cellule ou d’une suite de cellules.

Listes linéaires - Manipulation

- Création d’une cellule ou d’une suite de cellules.
- Accès à une cellule via un champ donné (algorithme de recherche).

Listes linéaires - Manipulation

- Création d’une cellule ou d’une suite de cellules.
- Accès à une cellule via un champ donné (algorithme de recherche).
- Insertion d’une cellule dans une liste linéaire.

Listes linéaires - Manipulation

- Création d’une cellule ou d’une suite de cellules.
- Accès à une cellule via un champ donné (algorithme de recherche).
- Insertion d’une cellule dans une liste linéaire.
- Suppression d’une cellule à partir d’une liste linéaire.

Listes linéaires - Création par lecture

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)
  $r.info \leftarrow x$

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)
  $r.info \leftarrow x$
  $r.suivant \leftarrow liste$

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)
  $r.info \leftarrow x$
  $r.suivant \leftarrow liste$
  $liste \leftarrow r$

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)
  $r.info \leftarrow x$
  $r.suivant \leftarrow liste$
  $liste \leftarrow r$
  $i \leftarrow i + 1$

Listes linéaires - Création par lecture

- **procedure** creerliste (E l: entier, S liste: celluleLineaire de T)
  variables r : celluleLineaire de T
  x: T
  i : entier
  $liste \leftarrow nil \quad i \leftarrow 1$
  tantque $i <= l$ **faire**
  nouveau(r)
  lire(x)
  $r.info \leftarrow x$
  $r.suivant \leftarrow liste$
  $liste \leftarrow r$
  $i \leftarrow i + 1$
  fin tantque

Listes linéaires - Parcours

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T

  $l \leftarrow liste$

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T

  $l \leftarrow liste$

  tantque $l <> nil$ **faire**

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T

  $l \leftarrow liste$

  tantque $l <> nil$ **faire**

  traiter(l)

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T
  $l \leftarrow liste$
  tantque $l <> nil$ **faire**
  traiter(l)
  $l \leftarrow l.suivant$

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T
  $l \leftarrow liste$
  tantque $l <> nil$ **faire**
  traiter(l)
  $l \leftarrow l.suivant$
  fin tantque

Listes linéaires - Parcours

- **procedure** parcours (E liste: celluleLineaire de T)
  variables l : celluleLineaire de T
  $l \leftarrow liste$
  tantque $l <> nil$ **faire**
  traiter(l)
  $l \leftarrow l.suivant$
  fin tantque

- traiter(l) étant une procédure donnée.

Listes linéaires - Accès par position

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)
  variables i : entier

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)
  variables i : entier
  l : celluleLineaire de T

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)
  variables i : entier
  l : celluleLineaire de T
  $l \leftarrow liste \quad i \leftarrow 1$

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)
  variables i : entier
  l : celluleLineaire de T
  $l \leftarrow liste \quad i \leftarrow 1$
  tantque $i < k$ ET $l <> nil$ **faire**

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

  variables i : entier

  l : celluleLineaire de T

  $l \leftarrow liste \quad i \leftarrow 1$

  tantque $i < k$ ET $l <> nil$ **faire**

  $i \leftarrow i + 1$

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

  variables i : entier

  l : celluleLineaire de T

  $l \leftarrow liste \quad i \leftarrow 1$

  tantque $i < k$ ET $l <> nil$ **faire**

  $i \leftarrow i + 1$

  $l \leftarrow l.suivant$

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

  variables i : entier

  l : celluleLineaire de T

  $l \leftarrow liste \quad i \leftarrow 1$

  tantque $i < k$ ET $l <> nil$ **faire**

  $i \leftarrow i + 1$

  $l \leftarrow l.suivant$

  fin tantque

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

  variables i : entier

  l : celluleLineaire de T

  $l \leftarrow liste \quad i \leftarrow 1$

  tantque $i < k$ ET $l <> nil$ **faire**

  $i \leftarrow i + 1$

  $l \leftarrow l.suivant$

  fin tantque

  $trouve \leftarrow ((i = k) ET (l <> nil))$

Listes linéaires - Accès par position

- **procedure** accesk (E liste: celluleLineaire de T, E k : entier, S r: celluleLineaire de T, S trouve:Booleen)

  variables i : entier

  l : celluleLineaire de T

  $l \leftarrow liste \quad i \leftarrow 1$

  tantque $i < k$ ET $l <> nil$ **faire**

  $i \leftarrow i + 1$

  $l \leftarrow l.suivant$

  fin tantque

  $trouve \leftarrow ((i = k) ET (l <> nil))$

  $r \leftarrow l$

Listes linéaires - Accès par valeur

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T, S r: celluleLineaire de T, S acces :booleen)

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**
  si $l.info = v$ **alors**

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**
  si $l.info = v$ **alors**
  $trouve \leftarrow VRAI$

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)
  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**
  si $l.info = v$ **alors**
  $trouve \leftarrow VRAI$
  sinon

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)

  variables trouve : booleen

  l : celluleLineaire de T

  $l \leftarrow liste \quad trouve \leftarrow FAUX$

  tantque non trouve ET $l <> nil$ **faire**

  si $l.info = v$ **alors**

  $trouve \leftarrow VRAI$

  sinon

  $l \leftarrow l.suivant$

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)

  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**
  si $l.info = v$ **alors**
  $trouve \leftarrow VRAI$
  sinon
  $l \leftarrow l.suivant$
  fin si

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)

  variables trouve : booleen

  l : celluleLineaire de T

  $l \leftarrow liste \quad trouve \leftarrow FAUX$

  tantque non trouve ET $l <> nil$ **faire**

  si $l.info = v$ **alors**

  $trouve \leftarrow VRAI$

  sinon

  $l \leftarrow l.suivant$

  fin si

  fin tantque

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)

  variables trouve : booleen
  l : celluleLineaire de T
  $l \leftarrow liste \quad trouve \leftarrow FAUX$
  tantque non trouve ET $l <> nil$ **faire**
  si $l.info = v$ **alors**
  $trouve \leftarrow VRAI$
  sinon
  $l \leftarrow l.suivant$
  fin si
  fin tantque
  $acces \leftarrow trouve$

Listes linéaires - Accès par valeur

- **procedure** accesV (E liste: celluleLineaire de T, E v : T,
  S r: celluleLineaire de T, S acces :booleen)

variables trouve : booleen

l : celluleLineaire de T

$l \leftarrow liste \quad trouve \leftarrow FAUX$

tantque non trouve ET $l <> nil$ **faire**

si $l.info = v$ **alors**

$trouve \leftarrow VRAI$

sinon

$l \leftarrow l.suivant$

fin si

fin tantque

$acces \leftarrow trouve$

$r \leftarrow l$

Listes linéaires - Insertion dans la tête

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)
  variables l: celluleLineaire de T

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)
  variables l: celluleLineaire de T
  $nouveau(l)$

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)
  variables l: celluleLineaire de T
  $nouveau(l)$
  $l.info \leftarrow v$

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)
  variables l: celluleLineaire de T
  $nouveau(l)$
  $l.info \leftarrow v$
  $l.suivant \leftarrow liste$

Listes linéaires - Insertion dans la tête

- **procedure** insererTete (ES liste: celluleLineaire de T, E v : T)
  variables l: celluleLineaire de T

  $nouveau(l)$

  $l.info \leftarrow v$

  $l.suivant \leftarrow liste$

  $liste \leftarrow l$

Listes linéaires - Insertion position K

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$
  sinon

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$
  sinon
  $accesk(liste, k-1, preced, possible)$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$
  sinon
  $accesk(liste, k-1, preced, possible)$
  si possible **alors**

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$
  sinon
  $accesk(liste, k-1, preced, possible)$
  si possible **alors**
  $nouveau(l)$ $l.info \leftarrow v$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)
  variables l,preced: celluleLineaire de T
  $possible \leftarrow FAUX$
  si k=1 **alors**
  insererTete(liste,v) $possible \leftarrow VRAI$
  sinon
  $accesk(liste, k-1, preced, possible)$
  si possible **alors**
  $nouveau(l)$ $l.info \leftarrow v$
  $l.suivant \leftarrow preced.suivant$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)

  variables l,preced: celluleLineaire de T

  $possible \leftarrow FAUX$

  si k=1 **alors**

  insererTete(liste,v) $possible \leftarrow VRAI$

  sinon

  $accesk(liste, k-1, preced, possible)$

  si possible **alors**

  $nouveau(l)$ $l.info \leftarrow v$

  $l.suivant \leftarrow preced.suivant$

  $preced.suivant \leftarrow l$

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)

  variables l,preced: celluleLineaire de T

  $possible \leftarrow FAUX$

  si k=1 **alors**

  insererTete(liste,v) $possible \leftarrow VRAI$

  sinon

  $accesk(liste, k-1, preced, possible)$

  si possible **alors**

  $nouveau(l)$ $l.info \leftarrow v$

  $l.suivant \leftarrow preced.suivant$

  $preced.suivant \leftarrow l$

  fin si

Listes linéaires - Insertion position K

- **procedure** insererK (ES liste: celluleLineaire de T, E k : entier, E v : T, S possible : booleen)

  variables l,preced: celluleLineaire de T

  $possible \leftarrow FAUX$

  si k=1 **alors**

  insererTete(liste,v) $possible \leftarrow VRAI$

  sinon

  $accesk(liste, k-1, preced, possible)$

  si possible **alors**

  $nouveau(l)$ $l.info \leftarrow v$

  $l.suivant \leftarrow preced.suivant$

  $preced.suivant \leftarrow l$

  fin si

  fin si

Listes linéaires - Suppression de la tête

Listes linéaires - Suppression de la tête

- **procedure** supptête (ES liste :celluleLineaire de T)

Listes linéaires - Suppression de la tête

- **procedure** supptête (ES liste :celluleLineaire de T)
  variables l: celluleLineaire de T

Listes linéaires - Suppression de la tête

- **procedure** supptête (ES liste :celluleLineaire de T)
  variables l: celluleLineaire de T

  $l \leftarrow liste$

Listes linéaires - Suppression de la tête

- **procedure** supptête (ES liste :celluleLineaire de T)
  variables l: celluleLineaire de T

  $l \leftarrow liste$

  $liste \leftarrow l.suivant$

Listes linéaires - Suppression de la tête

- **procedure** supptête (ES liste :celluleLineaire de T)
  variables l: celluleLineaire de T

  $l \leftarrow liste$

  $liste \leftarrow l.suivant$

  laisser(l)

Listes linéaires - Suppression position K

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T
  trouve:booleen

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T
  trouve:booleen
  si k=1 **alors**

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T
  trouve:booleen
  si k=1 **alors**
  $possible \leftarrow VRAI$ supptête(liste)

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T
  trouve:booleen
  si k=1 **alors**
  $possible \leftarrow VRAI$ supptête(liste)
  sinon

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

  $preced \leftarrow accesk(liste, k-1, preced, trouve)$

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

  $preced \leftarrow accesk(liste, k-1, preced, trouve)$

  si preced<>nil **alors**

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

  $preced \leftarrow accesk(liste, k-1, preced, trouve)$

  si preced<>nil **alors**

  $l \leftarrow preced.suivant$

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)
  variables l,preced: celluleLineaire de T
  trouve:booleen
  si k=1 **alors**
  $possible \leftarrow VRAI$ supptête(liste)
  sinon
  $possible \leftarrow FAUX$
  $preced \leftarrow accesk(liste, k-1, preced, trouve)$
  si preced<>nil **alors**
  $l \leftarrow preced.suivant$
  si l<>nil **alors**

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

  $preced \leftarrow accesk(liste, k-1, preced, trouve)$

  si preced<>nil **alors**

  $l \leftarrow preced.suivant$

  si l<>nil **alors**

  $possible \leftarrow VRAI$ $preced.suivant \leftarrow l.suivant$

Listes linéaires - Suppression position K

- **procedure** suppK (ES liste :celluleLineaire de T, E k : entier, S possible : booleen)

  variables l,preced: celluleLineaire de T

  trouve:booleen

  si k=1 **alors**

  $possible \leftarrow VRAI$ supptête(liste)

  sinon

  $possible \leftarrow FAUX$

  $preced \leftarrow accesk(liste, k-1, preced, trouve)$

  si preced<>nil **alors**

  $l \leftarrow preced.suivant$

  si l<>nil **alors**

  $possible \leftarrow VRAI$ $preced.suivant \leftarrow l.suivant$

  laisser(l)