Algorithmique

Chapitre 4 : Listes chaînées

Mise à jour

Insertion

Mise à jour dans une liste

- Insertion et suppression
- Facilité de mise en œuvre
  - modification d'un ou deux pointeurs
  - ni décalage (vecteurs)
  - ni recopie (fichiers)

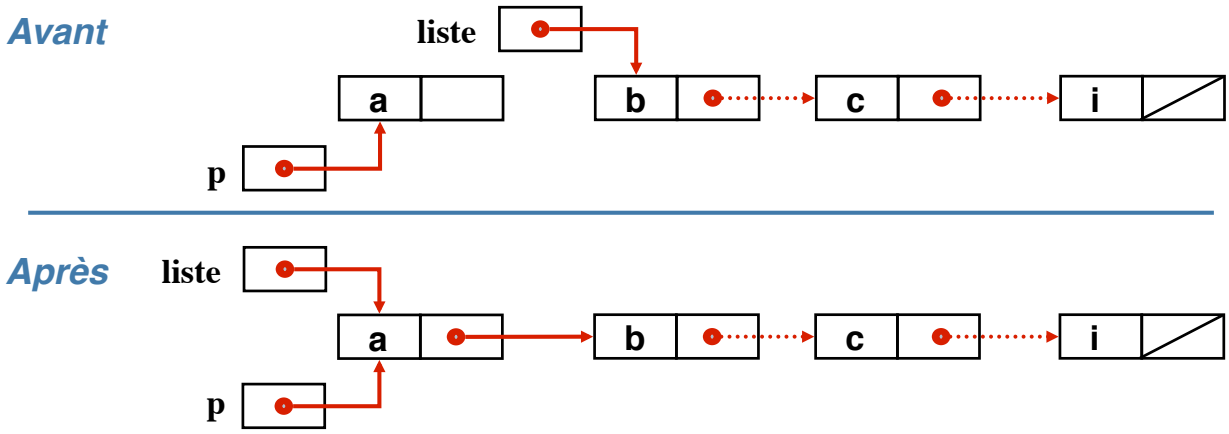
Insertion en tête de liste

- Il s'agit d'écrire

  procédure insertête (dr liste : pointeur ; d elem : t) ;

  spécification *{liste⁺ = la⁺}* ➔ *{liste⁺ = <elem>//la⁺}*

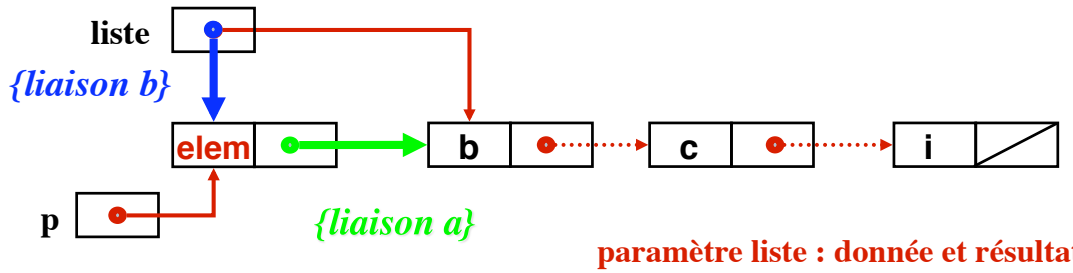
- On modifie l'adresse de la liste
- Exemple



Insertion en tête de liste

procédure insertête (dr liste : pointeur ; d elem : t) ;
spécification *{liste⁺ = la⁺}* ➔ *{liste⁺ = <elem>//la⁺}*

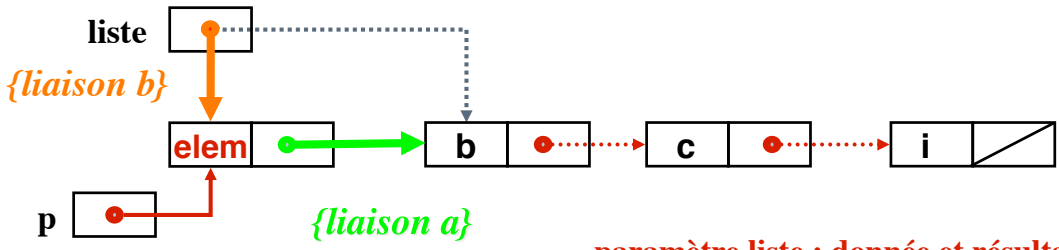
```
    p : pointeur ;
debproc
    nouveau(p);              {créer une cellule d'adresse p}
    p↑.info := elem ;
    p↑.suivant := liste ;            {liaison a}
    liste := p ;                     {liaison b}
finproc ;
```



paramètre liste : donnée et résultat
valable même si liste = nil

Insertion en tête de liste (ada)

```
procedure insertête (liste : in out Ta_ListEnt ;
                                    elem : in integer) is
  p : Ta_ListEnt ;
begin
  p := new Tr_ListEnt; --nouvelle cellule d'adresse p
  p.all.info := elem ;
  p.all.suivant := liste ;    --liaison a}
  liste := p ;                --liaison b}
end insertête ;
```

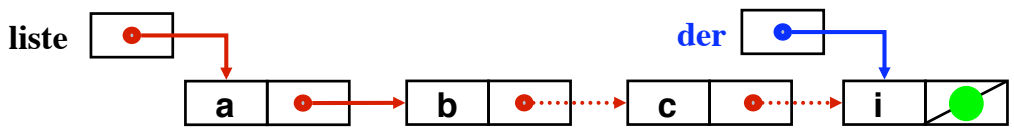


paramètre liste : donnée et résultat
valable même si liste = nil

Insertion en fin de liste

- Insérer en fin de liste = insérer en tête de la liste qui suit la dernière cellule

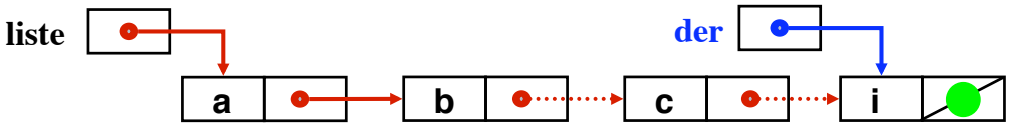
procédure inserfin (dr liste : pointeur ; d elem : t) ;
spécification *{liste⁺ = la⁺}* ➔ *{liste⁺=la⁺||<elem>}*
der : pointeur ;
debproc
si liste = nil alors
insertête (liste, elem) ;
sinon
der := dernier (liste) ;
insertête (der↑.suivant, elem) ;
finsi ;
finproc ;



Insertion en fin de liste (**ada**)

- Insérer en fin de liste = insérer en tête de la liste qui suit la dernière cellule

```
procedure inserfin (liste : in out Ta_ListEnt ;
                                    elem : in integer) is
  der : Ta_ListEnt;
begin
  if liste = null then
    insertête (liste, elem) ;
  else
    der := dernier (liste) ;
    insertête (der.all.suivant, elem) ;
  end if ;
end inserfin ;
```



Fonction dernier

fonction dernier (d liste : pointeur) : pointeur ;

spécification *{liste ≠ nil }* ➔ *{résultat = pointeur sur la dernière cellule de liste⁺}*

- Schéma récursif

➤ liste↑.suivant= nil

⇨ *la liste ne contient qu'une cellule*

⇨ résultat = liste ; ✳

➤ liste↑.suivant ≠ nil

⇨ résultat = dernier (liste↑.suivant) ;

Fonction dernier

fonction dernier (d liste : pointeur) : pointeur ;
spécification *{liste ≠ nil }* ➔ *{résultat = pointeur sur la dernière cellule de liste⁺}*

debfonc
  si liste↑.suivant = nil **alors**
    retour liste ;
  sinon
    retour dernier (liste↑.suivant) ;
  finsi ;
finfonc ;

Fonction dernier (**ada**)

```
function dernier (liste : in Ta_ListEnt)
                                  return Ta_ListEnt is
--spec {liste ≠ nil } ➔ {résultat = pointeur
                  sur la dernière cellule de liste⁺}
begin
  if liste.all.suivant = null then
    return liste ;
  else
    return dernier (liste.all.suivant) ;
  end if ;
end dernier ;
```

Inserfin récursive

Schéma récursif

- liste⁺ = <> ⇨ insertête (liste, elem) ; ✳
- liste⁺ ≠ <> ⇨ insertfin (liste↑.suivant, elem) ;

procédure inserfin (dr liste : pointeur ; d elem : t) ;
spécification *{liste⁺ = la⁺}* ➔ *{liste⁺=la⁺||<elem>}*
debproc
  si liste = nil **alors**
    insertête (liste, elem) ;
  sinon
    inserfin (liste↑.suivant, elem) ;
  finsi ;
finproc ;

Inserfin récursive (**ada**)

```
procedure inserfin (liste : in out Ta_ListEnt ;
                                    elem : in integer) is
--spec {liste⁺ = la⁺} ➔ {liste⁺=la⁺||<elem>}
begin
   if liste = null then
     insertête (liste, elem) ;
   else
     inserfin (liste.all.suivant, elem) ;
   end if ;
end inserfin  ;
```

Insertion à la k^{ième} place

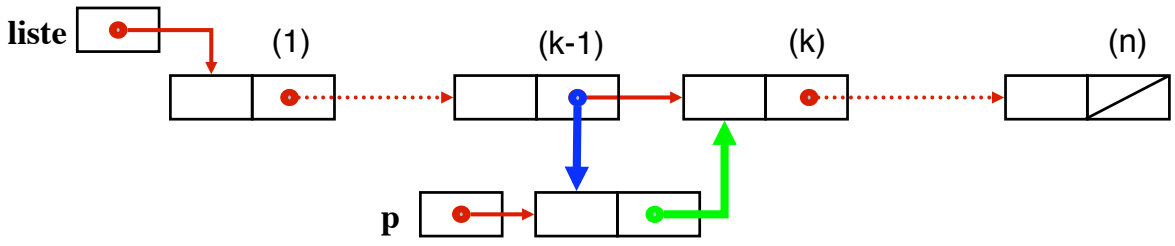
- On veut écrire

procédure insertk (dr liste : pointeur ; d k : entier ; d elem : t ;
r possible : booléen) ;

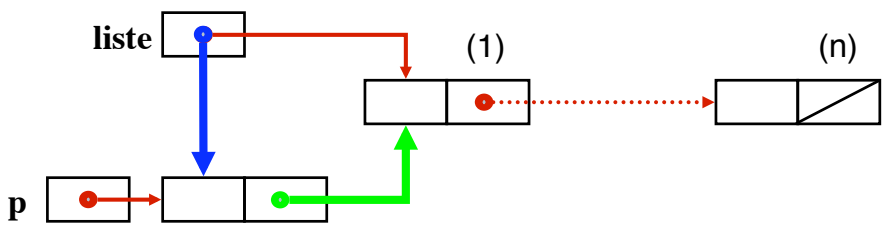
spécification *{}* ➔ *{(possible, elem inséré à la kième place)*
v (¬ possible, insertion impossible)}

Insertion à la k^{ième} place

- cas général



- k = 1 (insertion en tête)



Insertion à la k^{ième} place

- Schéma récursif

➤ k = 1 ⇨ insertête (liste, elem) ;
possible := vrai ; ✳

➤ k ≠ 1

➤➤ liste^{+} = <>
⇨ possible := faux ; ✳

➤➤ liste^{+} ≠ <>
⇨ insertk (liste↑.suivant, k-1, elem, possible) ;

Insertion à la k^{ième} place

procédure insertk (dr liste : pointeur ; d k : entier ; d elem : t ;
r possible : booléen) ;

spécification {} ➔ *{(possible, elem inséré à la kième place)*
v (¬ possible, insertion impossible)}

debproc
si k = 1 **alors** *{insertion en tête }*
insertête (liste, elem) ;
possible := vrai ;
sinonsi liste = nil **alors** *{insertion impossible}*
possible := faux ;
sinon *{insertion à la k-1^{ième} place de liste↑.suivant^{+}}*
insertk (liste↑.suivant, k-1, elem, possible) ;
finsi ;
finproc ;

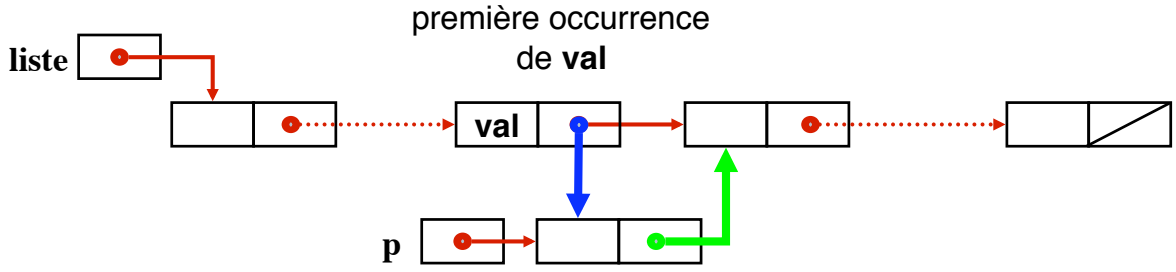
Insertion à la k^{ième} place (**ada**)

```
procedure insertk (liste : in out Ta_ListEnt ; k : in integer ;
                   elem : in integer ; possible : out boolean) is
--spec {} ➔ {(possible, elem inséré à la kième place)
                              ∨ (¬ possible, insertion impossible)}
begin
  if k = 1 then       --insertion en tête
    insertête (liste, elem) ;
    possible := true ;
  else if liste = null then  --insertion impossible
    possible := false ;
  else      --insertion à la k-1ième place de liste.suivant+
    insertk (liste↑.suivant, k-1, elem, possible) ;
  end if ;
end insertk ;
```


Insertion après la première occ. de val

procédure insert (d liste : pointeur ; d val, elem : t ; r possible : booléen) ;

spécification { } ➔ *{(possible, insertion de elem après la première occurrence de val dans liste) ∨ (¬ possible, insertion impossible)}*



Insertion après la première occ. de val

- Schéma récursif

➤ liste⁺ = <> ⇨ possible := faux ; ✳

➤ liste⁺ ≠ <>

➤➤ liste↑.info = val

⇨ insertête (liste↑.suivant, elem) ;
possible := vrai ; ✳

➤➤ liste↑.info ≠ val

⇨ insert (liste↑.suivant, val, elem, possible) ;

Insertion dans une liste triée

- Schéma récursif

➤ liste⁺ = <> ⇨ insertête (liste, elem) ; ✳

➤ liste⁺ ≠ <>

➤➤ liste↑.info < val

⇨ insertri (liste↑.suivant, elem) ;

➤➤ liste↑.info ≥ val

⇨ insertête (liste, elem) ; ✳

Insertion dans une liste triée

procédure insertri (dr liste : pointeur ; d elem : t) ;
spécification *{liste⁺ triée, liste⁺= la⁺|| lb⁺, la⁺< elem ≤ lb⁺}*
➔ *{liste⁺ triée, liste⁺= la⁺||<elem>|| lb⁺}*
debproc
si (liste = nil) **ou sinon** (liste↑.info ≥ elem) **alors**
insertête (liste, elem) ;
sinon *{liste↑.info < elem}*
insertri (liste↑.suivant, elem) ;
finsi ;
finproc ;

Insertion dans une liste triée (**ada**)

```
procedure insertri (liste : in out Ta_ListEnt ;
                                    elem : in integer) ;
--spec {liste⁺ triée, liste⁺=la⁺||lb⁺, la⁺< elem ≤ lb⁺}
          ➔ {liste⁺ triée, liste⁺= la⁺||<elem>|| lb⁺}
debproc
  if (liste = null) or else (liste.all.info >= elem)
   then
    insertête (liste, elem) ;
  else    --liste.all.info < elem
    insertri (liste.all.suivant, elem) ;
  end if ;
end insertri ;
```