Algorithmique & programmation

Chapitre 4 : Listes chaînées

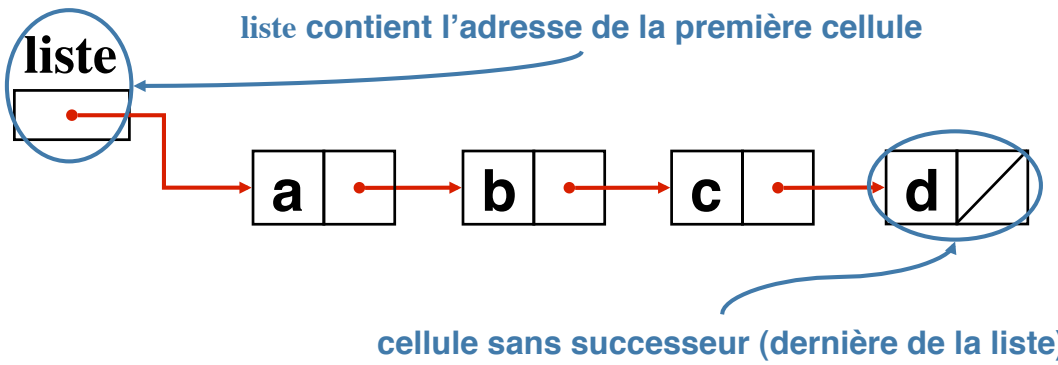
Définitions d'une liste chaînée

Création d'une liste chaînée

Liste linéaire chaînée

- Une liste linéaire chaînée est un ensemble de cellules chaînées entre elles
  - On connaît l'adresse de la première cellule qui est l'adresse de la liste (variable liste)

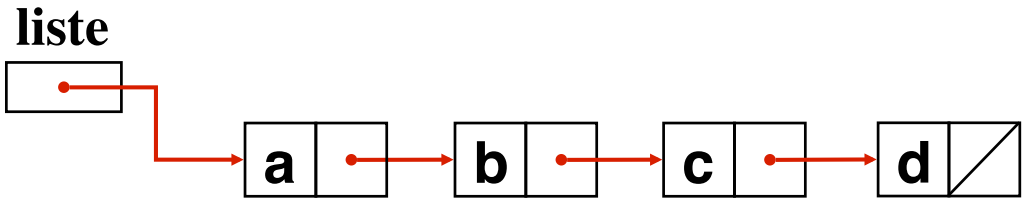
- Exemple



Notation

- **liste⁺**
  - suite des éléments contenus dans la liste dont l'adresse de la première cellule se trouve dans la variable liste

- Exemple



  - **liste⁺** = (a, b, c, d)

Notations

- Si **liste** = nil
  - **liste⁺** = <> [la liste est vide]

- Une liste contient autant de cellules que d'éléments

- Champ **suivant** d'une cellule
  = adresse de la cellule suivante

- Champ **suivant** de la dernière cellule
  = **nil**

Accès aux éléments d'une liste

- L'adresse de la première cellule, dans la variable **liste**, permet l'accès à tous les éléments
  - **liste↑.info**
  - **liste↑.suivant**
  - **liste↑.suivant↑.info**
  - etc.

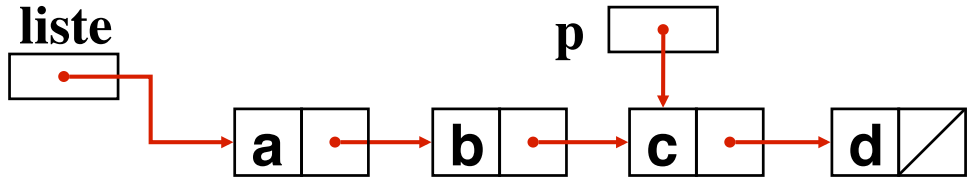
- **Rappel** : **liste↑.info** et **liste↑.suivant** ne sont pas définis si **liste = nil**

Notations

- Relations d'ordre
  - **a < liste⁺** *signifie* **a <** tous les éléments de **liste⁺**
  - on a aussi toutes les autres relation d'ordre

- Notion de **sous-liste**
  - Si **liste⁺** = (a, b, c, d)
  - alors
    - (b, c) est une sous-liste de **liste⁺**
    - (b, d) n'est pas une sous-liste de **liste⁺**

Notation

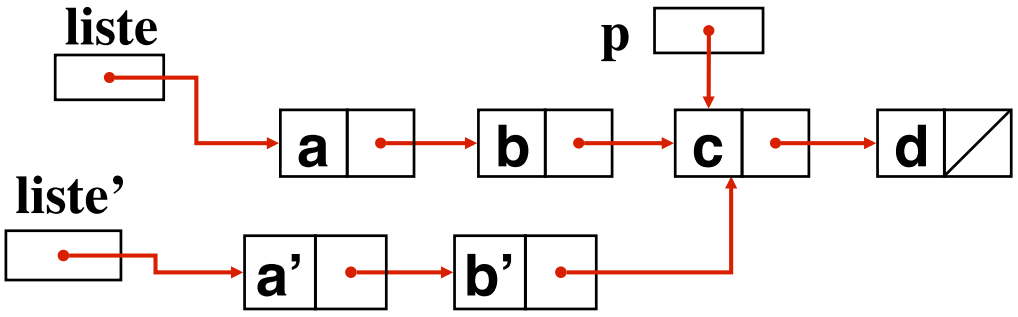


- p^+ = (c, d)
- p^-_{liste} = (a, b)
  - si p^-_{liste} n'est pas ambiguë, on écrit p^-

- $liste^+$ = (a, b, c, d)
- $liste^-$ = ()

En particulier
si liste = nil alors $liste^+$ = <>

Exemple d'ambiguïté pour p^-



- p^- est ambiguë, car on ne sait pas si c'est
  - p^-_{liste} = (a, b)

ou

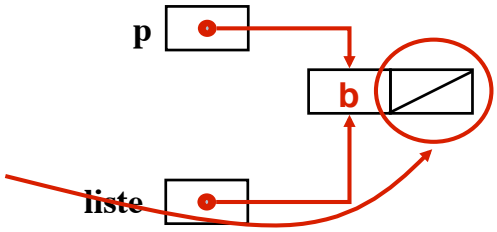
  - $p^-_{liste'}$ = (a', b')

Création d'une liste (algo)

- On veut créer la liste chaînée qui représente (a, b), à partir du dernier élément

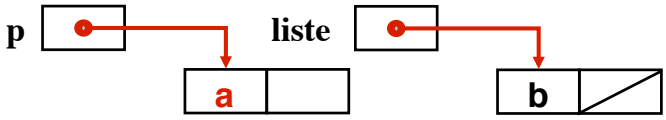
1. créer une liste vide d'adresse **liste**
   - **liste := nil ;**

2. créer une liste, d'adresse **liste**, contenant **b**
   - **nouveau (p) ;**
   - **p↑.info := b ;**
   - **p↑.suivant := liste ;**
   - **liste := p ;**

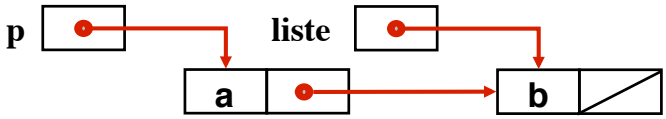


Création d'une liste (algo)

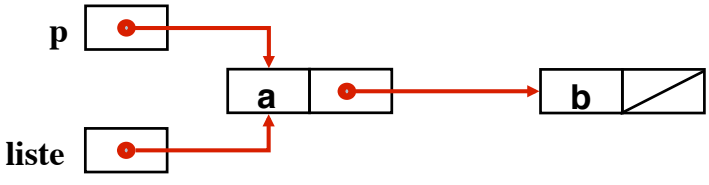
3. créer une liste d'adresse liste contenant (a, b)
   - **nouveau (p) ;**
   - **p↑.info := a ;**



   - **p↑.suivant := liste ;**



   - **liste := p ;**



Création d'une liste

- Comment a-t-on procédé ?
  - **On a fait des insertions en tête de liste !!**

Création d'une liste d'entiers (**ada**)

```
type Tr_ListEnt;
type Ta_ListEnt is access Tr_ListEnt;
type Tr_ListEnt is record
                     info    : Integer;
                     suivant: Ta_ListEnt;
end record;

list, p : Ta_ListEnt;  --création d'une liste (24,25)

begin
   list := null;    --création d'une liste vide
   p := new Tr_ListEnt;
   p.all.info := 25;
   p.all.suivant := list;
   list := p;    --insertion de 25 en tête
   p := new Tr_ListEnt;
   p.all.info := 24;
   p.all.suivant := list;
   list := p;    --insertion de 24 en tête
end …;
```

Fichier vers liste chaînée inverse

- On va généraliser la méthode précédente
  - Les éléments sont dans f
  - On va les lire successivement (du premier vers le dernier ➔ parcours complet)
  - À chaque lecture on va faire une insertion en tête de liste
  - On note qu'à chaque étape le dernier inséré est en tête

  ➔ La liste contiendra les éléments de f dans l'ordre inverse !

Fichier vers liste chaînée inverse (**algo**)

```
procédure créerliste (d f : fichier de t ; r liste : pointeur) ;
spécification { } ➔ {création de liste+ composée des éléments de f}
    p, liste1 : pointeur ; c : t ;
debproc
    liste1 := nil ;
    relire (f) ;
    tantque non fdf (f) faire
        nouveau (p) ;              {création d'une cellule}
        lire (f, c) ;              {lecture de l'info suivante}
        p↑.info := c ;             {champ information}
        p↑.suivant := liste1 ;     {chaînage}
        liste1 := p ;              {nouvelle liste}
    finfaire ;
    liste := liste1 ;              liste paramètre résultat
finproc ;                          liste interdite en partie droite d'affectation
```

Fichier vers liste chaînée inverse (**ada**)

```
procedure créerliste (F : in P_Entier_Io.File_Type ;
                                        liste : out Ta_ListEnt) is
--spec { } ➔ {création de liste+ composée des éléments de f}
  p, liste1 : Ta_ListEnt ; c : integer ;
begin
  liste1 := null ;
  reset (F) ;
  while not End_Of_File(F) loop
    p := new Tr_ListEnt ;    --création d'une cellule
    read (F, c) ;            --lecture de l'entier suivant
    p.all.info := c ;        --champ information
    p.all.suivant := liste1 ;  --chaînage
    liste1 := p ;              --nouvelle liste
  end loop;
  liste := liste1 ;
end créerliste;
```

liste paramètre résultat

liste interdite en partie droite d'affectation

Vecteur vers liste chaînée

- Peut-on conserver l'ordre des éléments du vecteur dans la liste ?
  - On sait faire une insertion en tête
  - Pour conserver l'ordre il faudrait insérer les éléments dans l'ordre inverse
  - Avec un vecteur c'est possible !!
    - On parcourt le vecteur de droite à gauche pour avoir une liste « à l'endroit »

Vecteur vers liste chaînée (**algo**)

procédure vectliste (d V[1..n] : vecteur ; **r liste : pointeur**) ;
spécification *{ }* ➔ *{création de liste⁺ composée des éléments de V}*
    i : entier ;
    p, **liste1** : pointeur ;
debproc
    liste1 := nil ;
    i := n ;
    tantque i > 1 **faire**
        nouveau (p) ;
        p↑.info := V[i] ;
        p↑.suivant := **liste1** ;
        liste1 := p ;
        i := i -1 ;
    finfaire ;
    liste := **liste1** ;
finproc ;

liste paramètre résultat
liste interdit en partie droite d'affectation

Vecteur vers liste chaînée (**ada**)

```
procedure vectliste (V : in vecteur ;
                                        liste : out Ta_ListEnt) is
--spec { } ➔ {création de liste+ composée des éléments de V}
  i : integer;
  p, liste1 : Ta_ListEnt ;
begin
  liste1 := null ;
  i := V'Last ;
  while i > V'First loop
    p := new Tr_ListEnt ;
    p.all.info := V[i] ;
    p.all.suivant := liste1 ;
    liste1 := p ;
    i := i - 1 ;
  end loop ;
  liste := liste1 ;
end vectliste ;
```

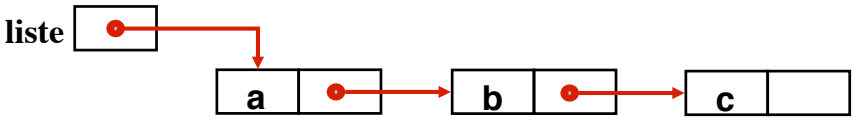
liste paramètre résultat
liste interdit en partie droite d'affectation

Fichier vers liste chaînée à l'endroit

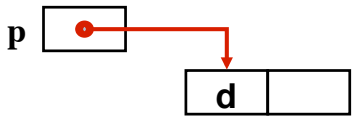
- Il faudrait pouvoir faire une insertion en fin de liste !
  - Créer la liste à partir de son premier élément
    - Traiter le premier élément de f
      - créer la première cellule de la liste
      - préserver son adresse
    - Pour chaque nouvel élément de f
      - créer une nouvelle cellule
      - l'insérer à la fin de la liste
      - ⇒ garder un pointeur sur le dernier élément de la liste

Fichier vers liste chaînée à l'endroit

- Exemple
  - Situation à l'étape i

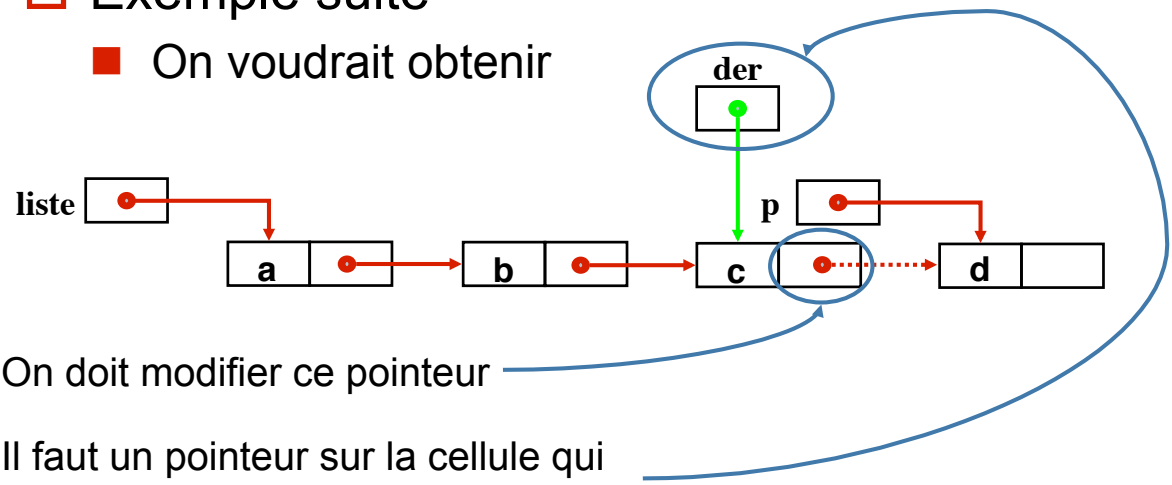


  - On veut insérer d
    - Créer une nouvelle cellule, y ranger d



Fichier vers liste chaînée à l'endroit

- Exemple suite
  - On voudrait obtenir



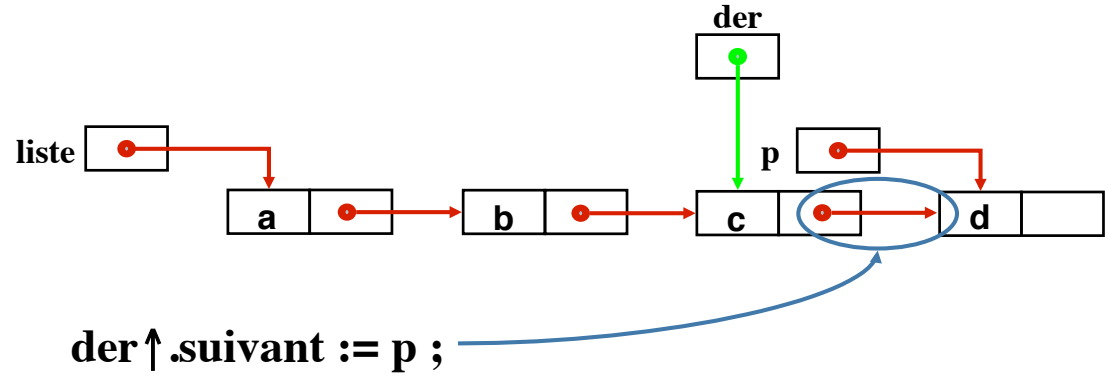
On doit modifier ce pointeur

Il faut un pointeur sur la cellule qui contient le dernier (avant insertion)

Le pointeur à modifier est : **der↑.suivant**

Fichier vers liste chaînée à l'endroit

- Exemple suite
  - Modification de **der↑.suivant**

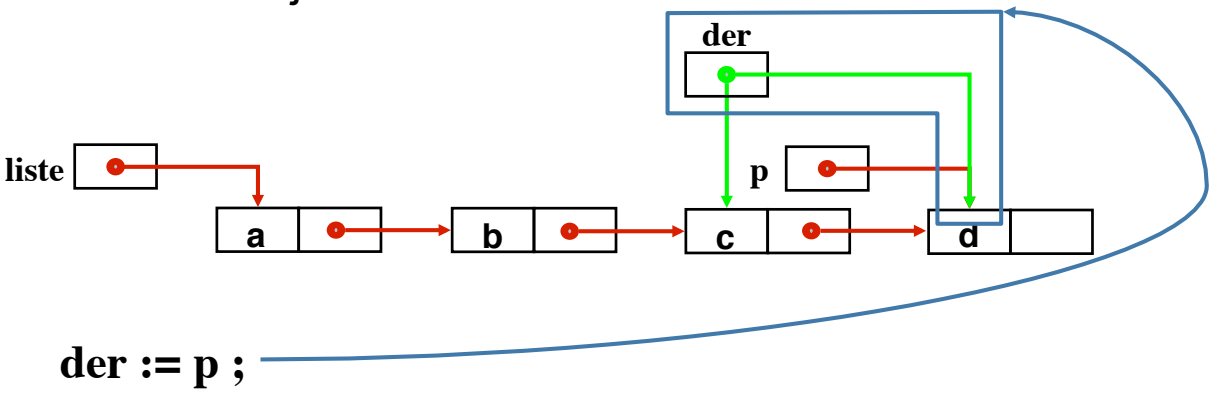


der↑.suivant := p ;

Il faut maintenant mettre à jour **der**

Fichier vers liste chaînée à l'endroit

- Exemple suite
  - Mise à jour de **der**



der := p ;

Fichier vers liste chaînée à l'endroit (algo)

```
procédure créerliste (d f : fichier de t ; r liste : pointeur) ;
spécification { } ➔ {création de liste+, à l'endroit, composée des éléments du fichier f}
      p, der : pointeur ; c : t ;
debproc
      relire (f) ;
      si fdf (f) alors
            liste := nil ;
      sinon
            lire(f, c) ;
            nouveau(der) ; der↑.info:= c ;      {der = adresse de la dernière cellule}
            liste := der ;
            tantque non fdf (f) faire
                  lire (f, c) ;
                  nouveau (p) ;                 {création d'une nouvelle cellule}
                  p↑.info := c ;
                  der↑.suivant := p ;           {ajout en fin de liste}
                  der := p ;                    {adresse de la dernière cellule}
            finfaire ;
            der↑.suivant := nil ;               {fin de liste}
      finsi ;
finproc ;
```

Fichier vers liste chaînée à l'endroit (ada)

```
procedure créerliste (F : in P_Entier_Io.File_Type ;
                                                liste : out Ta_ListEnt) is
--spec { } ➔ {création de liste+, à l'endroit, avec les éléments de f}
  p, der : Ta_ListEnt ; c : integer ;
begin
  reset (F) ;
  if End_Of_File(F) then
    liste := null ;
  else
    read (F, c) ;
    der :=  new Tr_ListEnt ;     --der = adresse de la dernière cellule
    der.all.info:= c ;
    liste := der ;               --liste = le premier dernier
    while not End_Of_File(F) loop
      read (F, c) ;
      p := new Tr_ListEnt ;      --création d'une nouvelle cellule
      p.all.info := c ;
      der.all.suivant := p ;     --ajout en fin de liste
      der := p ;                 --adresse de la dernière cellule
    end loop ;
    der.all.suivant := null ;    --fin de liste
  end if ;
end créerliste ;
```