Chapitre 13 – Les listes

Structures séquentielles.

Une liste sur un ensemble E est une suite finie (e_1, ..., e_n) d'éléments de E. La liste est vide si n=0. On différencie la place d'un élément dans la liste de l'élément lui-même. Les insertions et les suppressions peuvent se faire à toute place de la liste.

TAD Place

- Nécessité de commencer par définir le TAD Place qui est une case mémoire contenant un élément et un accès à la place suivante.

Sorte : Place

Utilise : Elément, TypeDebase

Opérations :

affecter-elt(Place, Elément) → Place {constructeur}

affecter-succ(Place, Place) → Place {constructeur}

Succ(Place) → Place {constructeur}

Contenu(Place) → Elément {observateur}

TAD Liste

Sorte : Liste

Utilise : Elément, Place, Entier, Booléen

Opérations :

Créer-liste-vide() → Liste {constructeur}

Insérer(Liste, Elément, Entier) → Liste {constructeur}

Supprimer(Liste, Entier) → Liste {constructeur}

Concaténer(Liste, Liste) → Liste {constructeur}

Accès(Liste, Entier) → Place {observateur}

Longueur(Liste) →Entier {observateur}

Liste-vide(Liste) → Booléen {observateur}

Ième(Liste, Entier) → Elément {observateur}

ième(l,i) = contenu(accès(l,i))

Représentation contiguë (par tableau) 1/4

- Les éléments de la liste sont mémorisés dans les cellules contiguës d'un tableau. Les notions de rang et de place sont confondues. Les insertions et suppressions dans la liste impliquent des décalages dans le tableau (i≠n).
- Le ième élément de la liste est placé dans la ième cellule du tableau.

```
Type liste = Enregistrement
            t : tableau [1..n] d'élément
            dernier : entier
              FinEnregistrement
Var l : liste
```

Représentation contiguë (par tableau) 2/4

```
Procédure insérer (E/S l : liste, E e : élément, i : entier)
Var j : entier
Début
Si (i>l.dernier+1) ou (i<1)
Alors erreur
Sinon   Pour j←l.dernier à i inc -1 Faire
            l.t[j+1]←l.t[j]
        FinPour
        l.dernier←l.dernier+1
        l.t[i]←e
FinSi
Fin
```

Représentation contiguë (par tableau) 3/4

```
Procédure supprimer (E/S l : liste, E i : entier)
Var j : entier
Début
Si (i>l.dernier) ou (i<1)
    Alors erreur
    Sinon   Pour j←i à l.dernier inc +1 Faire
            l.t[j]←l.t[j+1]
            FinPour
            l.dernier←l.dernier-1
FinSi
Fin
```

Représentation contiguë (par tableau) 4/4

```
Fonction localiser (E l : liste, e : élément) : entier
Var i : entier
Début
i←0
Répète
    i←i+1
Jusqu'à (i>l.dernier) ou (l.t[i]=e)
Si i>l.dernier
    Alors i←0
FinSi
Retourner(i)
Fin
```

Représentation chaînée 1/5

On utilise des pointeurs pour lier entre eux les éléments successifs, et la liste est alors déterminée par l'adresse de son premier élément.

```
Type    liste = ^cellule
        cellule = Enregistrement
                    val : élément
                    succ : liste
                  FinEnregistrement

Var l : liste
```

Représentation chaînée 2/5

DESSIN

Représentation chaînée 3/5

```
Procédure insérer (E/S l : liste, E e : elem, i : entier)
Var j : entier
    p, q : liste
Début
Si i=1 Alors p←allouer(cellule)
         p^.val←e
         p^.succ←l
         l←p
       Sinon  j←2
              q←l
              TantQue (j<i) et (q≠nil) Faire
                  j←j+1
                  q←q^.succ
              FinTantQue
              Si q≠nil Alors p←allouer(cellule)
                            p^.val←e
                            p^.succ←q^.succ
                            q^.succ←p
              FinSi
FinSi
Fin
```

Représentation chaînée 4/5

```
Procédure supprimer (E/S l : liste, E i : entier,)
Var j : entier
    p, q : liste
Début
Si i=1
    Alors p←l
          l←l^.succ
          récupérer(p)
    Sinon j←2
          q←l
          TantQue (j<i) et (q^.succ≠nil) Faire
            j←j+1
            q←q^.succ
          FinTantQue
          Si q^.succ≠nil
            Alors p←q^.succ
                  q^.succ←q^.succ^.succ
                  récupérer(p)
          FinSi
FinSi
Fin
```

Représentation chaînée 5/5

```
Fonction localiser (l : liste, e : élément) : entier
Var  i, j : entier
     q : liste
Début
j←0
i←0
q←l
TantQue (q≠nil) et (j=0) Faire
     i←i+1
     Si q^.val=e
          Alors j←i
          Sinon q←q^.succ
     FinSi
FinTantQue
Retourner(j)
Fin
```

Représentation par faux pointeurs

Certains langages de programmation ne comportent pas de pointeur.

On peut alors utiliser un tableau et des entiers pour représenter l'indice des éléments dans le tableau.

Il existe un nombre assez grand de cases pour y avoir plusieurs listes à la fois.

Pour pouvoir facilement insérer et supprimer des éléments dans les listes, il est important de chaîner entre elles les cases libres du tableau.

Déclaration

```
Type listab = tableau[1..n] de cellule
     cellule = Enregistrement
               val : élément
               succ : entier
               FinEnregistrement

Var t : listab
```

Exemple

l1 = 4 (a,~~b~~,c) {Suppression de b}

l2 = 6 (d,e)

libre = ~~8~~ 7 {q}

1		9
2	c	0
3	e	0
4	a	~~7~~ 2 {p}
5		10
6	d	3
7	~~b~~	~~2~~ 8
8		1
9		5
10		11
11		12
12		0

Principe

Pour insérer e dans une liste l, on utilise la première cellule libre et on la place à la bonne position dans l.
e est ensuite placé dans le champs val de cette cellule.
Pour supprimer une cellule de l, on la retire de l et on la met en tête de libre.
Nécessité d'une procédure transfert(p,q) :

- prendre la cellule c sur laquelle pointe p
- faire pointer q sur c en sauvegardant q
- faire pointer p sur l'élément pointé par c
- faire pointer c sur l'élément pointé par q auparavant.

Transfert

```
Procédure transfert(E/S t : listab, p,q :entier)
                    {p pointe sur c}
Var temp : entier
Début
   temp←q
   q←p
   p←t[q].succ
   t[q].succ←temp
Fin
```

Suppression

```
Procédure supprimer(E/S t:listab, l,libre:entier; E i:entier)
Var j,q : entier
Début
  j←1
  q←l
  TantQue (j<i) et (t[q].succ≠0) Faire
     q←t[q].succ
     j←j+1
  FintantQue {q pointe sur le p voulu de transfert}
  Si t[q].succ≠0
     Alors transfert(t,q,libre)
  FinSi
Fin
```

Insertion

```
Procédure insérer(E/S t:listab, l,libre:entier ;
                  E i:entier, e:élément)
Var  j,q : entier
Début
   j←1
   q←l
   TantQue (j<i) et (q≠0) Faire
      q←t[q].succ
      j←j+1
   FintantQue {q pointe sur le p voulu de transfert}
   Si q≠0
       Alors transfert(t,libre,q)
             t[q].val←e
   FinSi
Fin
```

Chapitre 13 – Les listes

Structures séquentielles.

Une liste sur un ensemble E est une suite finie (e_1, ..., e_n) d'éléments de E. La liste est vide si n=0. On différencie la place d'un élément dans la liste de l'élément lui-même. Les insertions et les suppressions peuvent se faire à toute place de la liste.

TAD Place

- Nécessité de commencer par définir le TAD Place qui est une case mémoire contenant un élément et un accès à la place suivante.

Sorte : Place
Utilise : Elément, TypeDebase
Opérations :
affecter-elt(Place, Elément) → Place {constructeur}
affecter-succ(Place, Place) → Place {constructeur}
Succ(Place) → Place {constructeur}
Contenu(Place) → Elément {observateur}

TAD Liste

Sorte : Liste
Utilise : Elément, Place, Entier, Booléen
Opérations :
Créer-liste-vide() → Liste {constructeur}
Insérer(Liste, Elément, Entier) → Liste {constructeur}
Supprimer(Liste, Entier) → Liste {constructeur}
Concaténer(Liste, Liste) → Liste {constructeur}
Accès(Liste, Entier) → Place {observateur}
Longueur(Liste) →Entier {observateur}
Liste-vide(Liste) → Booléen {observateur}
Ième(Liste, Entier) → Elément {observateur}
ième(l,i) = contenu(accès(l,i))

Représentation contiguë (par tableau) 1/4

- Les éléments de la liste sont mémorisés dans les cellules contiguës d'un tableau. Les notions de rang et de place sont confondues. Les insertions et suppressions dans la liste impliquent des décalages dans le tableau (i≠n).
- Le ième élément de la liste est placé dans la ième cellule du tableau.

```
Type liste = Enregistrement
            t : tableau [1..n] d'élément
            dernier : entier
              FinEnregistrement
Var l : liste
```

Représentation contiguë (par tableau) 2/4

```
Procédure insérer (E/S l : liste, E e : élément, i : entier)
Var j : entier
Début
Si (i>l.dernier+1) ou (i<1)
Alors erreur
Sinon   Pour j←l.dernier à i inc -1 Faire
            l.t[j+1]←l.t[j]
        FinPour
        l.dernier←l.dernier+1
        l.t[i]←e
FinSi
Fin
```

Représentation contiguë (par tableau) 3/4

```
Procédure supprimer (E/S l : liste, E i : entier)
Var j : entier
Début
Si (i>l.dernier) ou (i<1)
    Alors erreur
    Sinon   Pour j←i à l.dernier inc +1 Faire
            l.t[j]←l.t[j+1]
            FinPour
            l.dernier←l.dernier-1
FinSi
Fin
```

Représentation contiguë (par tableau) 4/4

```
Fonction localiser (E l : liste, e : élément) : entier
Var i : entier
Début
i←0
Répète
    i←i+1
Jusqu'à (i>l.dernier) ou (l.t[i]=e)
Si i>l.dernier
    Alors i←0
FinSi
Retourner(i)
Fin
```

Représentation chaînée 1/5

On utilise des pointeurs pour lier entre eux les éléments successifs, et la liste est alors déterminée par l'adresse de son premier élément.

```
Type    liste = ^cellule
        cellule = Enregistrement
                    val : élément
                    succ : liste
                  FinEnregistrement

Var l : liste
```

Représentation chaînée 2/5

DESSIN

Représentation chaînée 3/5

```
Procédure insérer (E/S l : liste, E e : elem, i : entier)
Var j : entier
    p, q : liste
Début
Si i=1 Alors p←allouer(cellule)
         p^.val←e
         p^.succ←l
         l←p
       Sinon  j←2
              q←l
              TantQue (j<i) et (q≠nil) Faire
                   j←j+1
                   q←q^.succ
              FinTantQue
              Si q≠nil Alors p←allouer(cellule)
                            p^.val←e
                            p^.succ←q^.succ
                            q^.succ←p
              FinSi
FinSi
Fin
```

Représentation chaînée 4/5

```
Procédure supprimer (E/S l : liste, E i : entier,)
Var j : entier
    p, q : liste
Début
Si i=1
    Alors p←l
          l←l^.succ
          récupérer(p)
    Sinon j←2
          q←l
          TantQue (j<i) et (q^.succ≠nil) Faire
            j←j+1
            q←q^.succ
          FinTantQue
          Si q^.succ≠nil
            Alors p←q^.succ
                  q^.succ←q^.succ^.succ
                  récupérer(p)
          FinSi
FinSi
Fin
```

Représentation chaînée 5/5

```
Fonction localiser (l : liste, e : élément) : entier
Var i, j : entier
    q : liste
Début
j←0
i←0
q←l
TantQue (q≠nil) et (j=0) Faire
    i←i+1
    Si q^.val=e
        Alors j←i
        Sinon q←q^.succ
    FinSi
FinTantQue
Retourner(j)
Fin
```

Représentation par faux pointeurs

Certains langages de programmation ne comportent pas de pointeur.
On peut alors utiliser un tableau et des entiers pour représenter l'indice des éléments dans le tableau.
Il existe un nombre assez grand de cases pour y avoir plusieurs listes à la fois.
Pour pouvoir facilement insérer et supprimer des éléments dans les listes, il est important de chaîner entre elles les cases libres du tableau.

Déclaration

```
Type listab = tableau[1..n] de cellule
     cellule = Enregistrement
               val : élément
               succ : entier
               FinEnregistrement

Var t : listab
```

Exemple

l1 = 4 (a,~~b~~,c) {Suppression de b}
l2 = 6 (d,e)

libre = ~~8~~ 7 {q}

1		9
2	c	0
3	e	0
4	a	~~7~~ 2 {p}
5		10
6	d	3
7	~~b~~	~~2~~ 8
8		1
9		5
10		11
11		12
12		0

Principe

Pour insérer e dans une liste l, on utilise la première cellule libre et on la place à la bonne position dans l.
e est ensuite placé dans le champs val de cette cellule.
Pour supprimer une cellule de l, on la retire de l et on la met en tête de libre.
Nécessité d'une procédure transfert(p,q) :

- prendre la cellule c sur laquelle pointe p
- faire pointer q sur c en sauvegardant q
- faire pointer p sur l'élément pointé par c
- faire pointer c sur l'élément pointé par q auparavant.

Transfert

```
Procédure transfert(E/S t : listab, p,q :entier)
                {p pointe sur c}
Var temp : entier
Début
  temp←q
  q←p
  p←t[q].succ
  t[q].succ←temp
Fin
```

Suppression

```
Procédure supprimer(E/S t:listab, l,libre:entier; E i:entier)
Var j,q : entier
Début
  j←1
  q←l
  TantQue (j<i) et (t[q].succ≠0) Faire
     q←t[q].succ
     j←j+1
  FintantQue {q pointe sur le p voulu de transfert}
  Si t[q].succ≠0
     Alors transfert(t,q,libre)
  FinSi
Fin
```

Insertion

```
Procédure insérer(E/S t:listab, l,libre:entier ;
                  E i:entier, e:élément)
Var  j,q : entier
Début
   j←1
   q←l
   TantQue (j<i) et (q≠0) Faire
      q←t[q].succ
      j←j+1
   FintantQue {q pointe sur le p voulu de transfert}
   Si q≠0
       Alors transfert(t,libre,q)
             t[q].val←e
   FinSi
Fin
```