Semaine 9
Listes - Parcours et mises à jour (suppression et insertion)

Munissez-vous aussi du document support de cours pour éventuellement compléter les rappels si besoin.

1 Algorithmes de parcours

Rappels. On illustre l'algorithme de parcours à travers une action qui affiche le contenu de la liste L :

```
Action afficheContenu(E L : Tliste)
var Adr : TAdresse;
début
Adr <- adressePremier(L);
Tant Que Adr <> NULL
Faire début
      écrire (valeurElément(L, Adr))
      Adr <- adresseSuivant(L, Adr)
      fin
fin
```

Exercice 23 : Ecrire une action qui calcule la moyenne d'une liste de `TEtudiant`. On suppose qu'un `TEtudiant` possède un champ Note...

```
Action moyenneListe(E L : Tliste, S Vide : booléen, S Moyenne : réel)
var Adr : TAdresse
    Somme : réel
    NbElem : entier
début
Somme <- 0
NbElem <- 0
Adr <- adressePremier(L)
Tant Que Adr <> NULL
Faire début
      Somme <- Somme + (valeurElément(L, Adr)).Note
      NbElem <- NbElem+1
      Adr <- adresseSuivant(L, Adr)
      fin
Si NbElem <> 0
Alors début
      Moyenne <- Somme/NbElem
```

```
      Vide <- Faux
Sinon Vide <- Vrai
fin
```

2 Mise à jour : suppression.

La suppression d'un élément s'effectue en 2 temps :

- la recherche de élément,

- puis sa suppression.

Exercice 24 : Prototype d'une fonction de recherche.

1. La fonction de recherche vue en cours est-elle adaptée ici ?
   Réponse : non, car il nous faut l'adresse sur précédent pour pouvoir utiliser supprimerAprès...
2. Prototype et sémantique de l'action de recherche.
   Réponse :
   ```
   Action rechercheElem( E L : TListe, Elem : TInfo ;
                         S Trouvé : booléen, AdrPrec : TAdresse )
   ```
   Si `Trouvé` est faux, l'élément n'est pas dans la liste, et `AdrPrec` n'a pas de sens. Si `Trouvé` est vrai, `AdrPrec` contient l'adresse de l'élément précédant l'élément cherché (NULL s'il est entête).

L'écriture de cette action de recherche sera différente selon que la liste sera triée ou non. Par contre, l'action qui supprime un élément restera la même.

Admettons que l'action `rechercheElem` fonctionne correctement.

Exercice 25 : Ecrire une action qui supprime un élément dans une liste.

```
Action SupprimerElément (ES L : Tliste ; E Elem : TInfo ; S Trouvé : booléen)
var Adr : TAdresse

début
rechercheElem(L, Elem, Trouvé, Adr)
Si Trouvé
Alors Si Adr = NULL
      Alors SupprimerEnTête(L)
      Sinon SupprimerAprès(L, Adr)
fin
```

Cas des listes non triées

Exercice 26 : Ecrire l'action `rechercheElem` dans le cas d'une liste non triée.

```
Action rechercheElem( E L : TListe, Elem : TInfo ;
```

```
                      S Trouvé : booléen, AdrPrec : TAdresse )
var Adr : TAdresse
    ElémLu : TInfo

début
Adr <- AdressePremier(L)
AdrPrec <- NULL
Trouvé <- Faux
Tant que non Trouvé et Adr<>NULL
Faire début
      ElémLu <- ValeurElément(L, Adr)
      Si ElémLu = Elem
      Alors Trouvé <- Vrai
      Sinon début
            AdrPréc <- Adr
            Adr <- AdresseSuivant(L, Adr)
            fin
      fin
fin
```

Cas des listes triées

Exercice 27 : Ecrire l'action `rechercheElem` dans le cas d'une liste triée.

```
Action rechercheElem( E L : TListe, Elem : TInfo ;
                      S Trouvé : booléen, AdrPrec : TAdresse )
var Adr : TAdresse
    ElémLu : TInfo
    Fini : booléen

début
Adr <- AdressePremier(L)
AdrPréc <- NULL
Trouvé <- Faux
Fini <- Faux
Tant que non Trouvé et non Fini et Adr <> NULL
Faire début
      ElémLu <- ValeurElément(L, Adr)
      Si ElémLu = Elem
      Alors Trouvé <- Vrai
      Sinon Si ElémLu > Elém
            Alors Fini <- Vrai
            Sinon début
                  AdrPréc <- Adr
                  Adr <- AdresseSuivant(L, Adr)
                  fin
      fin
fin
```

3 Mise à jour : insertion.

Cas des listes non triées
Il suffit d'insérer en tête, et les primitives le permettent.

Cas des listes triées

Comme dans le cas de la suppression, l'insertion d'un élément s'effectue en 2 temps :

- la recherche de la position d'insertion,

- puis l'insertion du nouvel élément.

Exercice 28 : Prototype d'une action de recherche de position.

```
Action rechercheElemPos( E L : TListe, Elem : TInfo ;
                         S AdrPrec : TAdresse )
```

`AdrPrec` contient l'adresse de l'élément après lequel doit avoir lieu l'insertion du nouvel élément, NULL si l'insertion doit se faire entête.

Admettons que l'action `rechercheElemPos` fonctionne correctement.

Exercice 29 : Ecrire une action qui ajoute un nouvel élément au bon endroit dans une liste triée.

```
Action InsèreElément (ES L : TListe ; E Elém : TInfo)
var AdrPréc : TAdresse

début
rechercheElemPos(L, Elém, AdrPréc)
Si AdrPréc = NULL Alors InsérerEnTête(L, Elém)
                  Sinon InsérerAprès(L, AdrPréc, Elém)
fin
```

Exercice 30 : Ecrire l'action `rechercheElemPos`.

```
Action rechercheElemPos( E L : TListe, Elem : TInfo ;
                         S AdrPrec : TAdresse )
var Adr : TAdresse
    Trouvé : booléen

début
Adr <- AdressePremier(L)
AdrPréc <- NULL
Trouvé <- Faux
Tant que non Trouvé et Adr <> NULL
Faire Si ValeurElément(L, Adr) >  Elém
      Alors Trouvé <- Vrai
      Sinon début
```

```
            AdrPréc <- Adr
            Adr <- AdresseSuivant(L, Adr)
            fin
fin
```