Chapitre 11 – Listes, piles, files

Structures séquentielles : Les listes

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 (sans avoir besoin de décaler les éléments)

TAD Place

- Nécessité de 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(Place, Elément) → Place	{constructeur}
Succ(Place) → Place	{constructeur}
Contenu(Place) → Elément	{observateur}

TAD Liste itérative

Sorte : Liste-it

Utilise : Elément, Place, Entier, Booléen

Opérations :

Créer-liste-vide() → Liste-it	{constructeur}
Insérer(Liste-it, Place, Elément) → Liste-it	{constructeur}
Supprimer(Liste-it, Place) → Liste-it	{constructeur}
Concaténer(Liste-it, Liste-it) → Liste-it	{constructeur}
Tête(Liste-it) → Place	{observateur}
Queue(Liste-it) → Place	{observateur}
Fini(Liste-it) → Place	{observateur}
Est-vide(Liste-it) → Booléen	{observateur}

Localisation élément

Retourne une place vide p si l'élément e n'est pas dans la liste

```
Fonction chercher-elt (l : liste-it, e : élément) : place
Var p : place
Début
p ← tête(l)
TantQue p≠fini(l) et contenu(p)≠e Faire
    p ← succ(p)
FinTantQue
Retourner(p)
Fin
```

Localisation ième

Retourne une place vide p, si i est plus grand que le nombre d'élément de la liste l ou si l est vide

```
Fonction chercher-ieme (l : liste-it, i : entier) : place
Var p : place
    j : entier
Début
p ← tête(l)
j ← 1
TantQue p≠fini(l) et j<i Faire
    p←succ(p)
    j ← j+1
FinTantQue
Retourner(p)
Fin
```

Insertion et suppression à un indice i

- Si i est supérieur au nombre d'éléments de la liste l, l'insertion se fera en queue de l

```
inserer(l,chercher-ieme(l,i),e)
```

- Si i est supérieur au nombre d'éléments de la liste l, la suppression ne sera pas faite

```
supprimer(l,chercher-ieme(l,i))
```

TAD Liste récursive

Sorte : Liste-recur

Utilise : Elément, Place, Entier, Booléen

Opérations :

Créer-liste-vide() → Liste-recur	{constructeur}
Cons(Liste-recur, Elément) → Liste-recur	{constructeur}
Concaténer(Liste-recur, Liste-recur) → Liste-recur	{constructeur}
Tête(Liste-recur) → Place	{observateur}
Reste(Liste-recur) → Liste-recur	{constructeur}
Est-vide(Liste-recur) → Booléen	{observateur}

Localisation élément

Retourne une place vide p, si l'élément e n'est pas dans la liste.
(Si l est vide, tête(l) retourne une place vide)

```
Fonction chercher-elt (l : liste-recur, e : élément) : place
Var p : place
Début
p ← tête(l)
Si ¬est-vide(l) et contenu(p)≠ e
        Alors p ← chercher-elt(reste(l),e)
FinSi
Retourner(p)
Fin
```

Localisation ième

Retourne une place vide p, si i est plus grand que le nombre d'élément de la liste l ou si l est vide

```
Fonction chercher-ieme (l : liste-recur, i : entier) : place
Var p : place
Début
p ← tête(l)
Si ¬est-vide(l) et i≠1
        Alors p ← chercher-ieme(reste(l),i-1)
FinSi
Retourner(p)
Fin
```

Insertion à un indice i

```
Si i est supérieur au nombre d'éléments de la liste l,
l'insertion se fera en queue de l

Fonction inserer-ieme (l : liste-recur, i : entier,
                       e : élément) : liste-recur
Var res : liste-recur
Début
Si est-vide(l) ou i=1
       Alors res ← cons(l,e)
       Sinon res ← cons(insérer-ieme(reste(l),i-1,e),tête(l))
FinSi
Retourner(res)
Fin
```

Suppression à un indice i

Si i est supérieur au nombre d'éléments de la liste l, la suppression ne sera pas faite

```
Fonction supprimer-ieme (l : liste-recur, i : entier) : liste-recur
Var res : liste-recur
Début
Si est-vide(l)
   Alors res ← l
   Sinon Si i=1
            Alors res ← reste(l)
            Sinon res ← cons(supprimer-ieme(reste(l),i-1),tête(l))
FinSi
Retourner(res)
Fin
```

Les piles

Structures séquentielles.

Une pile est un cas particulier de liste : les insertions et les suppressions se font à une seule extrémité, appelé *sommet de la pile*. LIFO : Last In First Out.

TAD Pile

Sorte : Pile

Utilise : Elément, Booléen

Opérations :

Créer-pile-vide() → Pile	{constructeur}
Empiler(Pile, Elément) → Pile	{constructeur}
Dépiler(Pile) → Pile	{constructeur}
Sommet(Pile) → Elément	{observateur}
Est-vide(Pile) → Booléen	{observateur}

Les files

Structures séquentielles.

Une file est un cas particulier de liste : on fait les ajouts à une extrémité et les suppressions se font à l'autre extrémité. FIFO : First In First Out.

TAD File

Sorte : File

Utilise : Elément, Booléen

Opérations :

créer-file-vide() → File	{constructeur}
Ajouter(File, Elément) → File	{constructeur}
Retirer(File) → File	{constructeur}
premier(File) → Elément	{observateur}
Est-vide(File) → Booléen	{observateur}