Algorithmique 1 : TD 3

Feuille 3 : Listes doublement chaînées

Dans les exercices suivants on considère le type `listeDC`, liste doublement chaînée, défini en cours [1].

Exercice 3.1

Soit l'implémentation de `listeDC` par `listeDC_car` avec l'allocation dynamique. Pour rappel voir annexe A.

1. Écrire les primitives du type `listeDC_car`.

```
fonction insererApres(ref L: listeDC_car, val X: car): vide;
fonction insererEnTete(ref L: listeDC_car, val X: car): vide;
fonction supprimerApres(ref L: listeDC_car): vide;
fonction supprimerEnTete(ref L: listeDC_car): vide;
```

2. Écrire les fonctions :
   (a) `fonction dernier(val L: listeDC_car): car;`
   qui renvoie la valeur du dernier élément de la liste `L` ;
   (b) `fonction ajout_en_queue_sans_doublons(ref L: listeDC_car, val X: car) : vide;`
   qui ajoute l'élément `X` en queue de la liste `L` s'il n'y est pas déjà ;
   (c) `fonction supprimer_derniere_occurrence(ref L: listeDC_car, val X: car) : vide;`
   qui supprime dans la liste `L` la dernière occurrence de l'élément `X` donné (s'il existe, sinon la fonction ne fait rien).

Exercice 3.2

En mathématiques, quand on considère un ensemble d'objets $\{x_1, x_2, \ldots, x_N\}$, on fait abstraction de l'ordre des objets et l'on ne repète pas plusieurs fois un même objet. Ainsi, l'ensemble des éléments qui composent la liste (a, b, c, a, d, e, c, d) est $\{a, b, c, d, e\}$.

1. Définir le type `ensemble` d'objets.
2. Ecrire une fonction qui convertit une liste `L` de type `listeDC_car` en un ensemble d'objets.
3. Ecrire une fonction qui calcule la réunion de deux ensembles constituée des éléments qui sont au moins dans un des ensembles.
4. Ecrire une fonction qui calcule l'intersection de deux ensembles constituée des éléments qui sont dans les deux ensembles.

Problème récurrent (notre fil d'ariane)

Exercice 3.3 Gestion d'une piste d'atterrissage des avions

Un avion est un enregistrement contenant :
- l'indicatif (6 caractères)
- la destination (30 caractères)
- l'autonomie résiduelle de carburant comptée en heures de vol (entier)
- deux booléens indiquant s'il y a un pirate à bord et s'il y a le feu.

1. Définir les structures de données nécessaires.
2. Ecrire la fonction `Priorité` ainsi que la gestion complète de la piste.
3. Envisager le cas de suppression d'un élément quelconque de la file lorsque le pirate a mis sa menace de détournement à exécution.

Quelles sont les notions que vous venez de voir qui peuvent permettre d'amorcer le fil d'ariane ?

1. http ://www.labri.fr/perso/maylis/ASDF/supports/notes/listesModif.html

Annexe A Implémentation du type listeDC par allocation dynamique

```
curseur= ^cellule;
car= type_predefini;
cellule= structure
           valeurElement: car;
           pointeurPrecedent: curseur;
           pointeurSuivant: curseur;
         finstructure;
listeDC_car= structure
               premier: curseur;
               dernier: curseur;
               cle:curseur;
             finstructure;
```

La liste vide est représentée par `NIL`.

- **Gestion de la mémoire**
  `fonction new(ref p: ^cellule) : vide;`
  `fonction delete(ref p: ^cellule) : vide;`
- **Accès**
  `fonction valeur(val L:listeDC_car) : car;`
  `fonction debutListe(ref L:listeDC_car) : vide;`
  `fonction finListe(ref L:listeDC_car): vide;`
  `fonction suivant(ref L:listeDC_car) : vide;`
  `fonction precedent(ref L:listeDC_car) : vide;`
  `fonction listeVide(val L:listeDC_car) : booleen;`
  `fonction getCleListe(val L:listeDC_car) : curseur;`
- **Modification**
  `fonction creerListe(ref L:listeDC_car) : vide;`
  `fonction insererApres(ref L:listeDC_car, val L:objet;) : vide;`
  `fonction insererEnTete(ref L:listeDC_car, val X:objet) : vide;`
  `fonction supprimerApres(ref L:listeDC_car) : vide;`
  `fonction supprimerEnTete(ref L:listeDC_car) : vide;`
  `fonction detruireListe(ref L:listeDC_car) : vide;`
  `fonction setCleListe(ref L:listeDC_car, curseur c) : vide;`