TD n° 1 : Le TDA LISTE

(D'après E. DELOZANNE, P. JACOBONI)

Objectifs du TD1 :

- Assimiler les notions d'interface et de réalisation
- Réviser les différentes façons de retourner un résultat (programmation impérative et fonctionnelle)
- Réviser les manipulations de pointeurs en C, en particulier les char *
- Assimiler les notations du cours et les différentes réalisations des listes étudiées en cours
- Se confronter aux problèmes de gestion de la mémoire dynamique en C

Exercice 1 : Description d'algorithmes à l'aide des primitives du TDA LISTE

a) Écrire une opération qui copie une liste dans une autre. Discuter les différentes possibilités (en-tête en style impératif et en style fonctionnel)
b) Écrire une opération qui modifie une liste en remplaçant toutes les occurrences de x par y puis retourne le nombre d'occurrences remplacées.
c) Écrire une opération qui renverse une liste dans une autre liste.Écrire une opération qui retourne une copie d'une liste obtenue en permutant les éléments placés aux positions p et q.
d) Écrire une opération qui concatène deux listes
e) Écrire un programme testant ces opérations.

Exercice 1 bis :

a) Réécrire l'opération de 1b) comme primitive du TDA Liste dans les deux réalisations : (1) par tableau, 2) par pointeurs).

Exercice 2 : Base de données - Stockage indirect

a) Donner une réalisation du TDA ELEMENT pour un stockage indirect. Ici un élément est un pointeur sur une fiche client d'une application de gestion. Les informations sur un client sont :
   - son prénom ;
   - son nom ;
   - son adresse (rue, ville et code postal).

   Détailler la gestion des champs dynamiques et écrire un programme de test.
b) Écrire une opération du TDA LISTE qui fusionne deux bases de données.
c) Écrire fonction qui affiche la liste des fiches des personnes de la même ville à l'aide des primitives du TDA LISTE et du TDA ELEMENT.
d) Calculer la complexité de l'opération écrite dans c).

Exercice 3 : Insertion en début et en fin de liste

a) Écrire les opérations qui permettent d'insérer au début et à la fin d'une liste à partir des primitives du TDA LISTE.
b) Écrire ces deux opérations comme des opérations primitives du TDA LISTE dans les deux réalisations proposées en cours.
c) Comparer les complexités de ces opérations pour chacune de ces trois mises en œuvre

Les primitives du TDA Liste

```
/*************************************************************
* Fichier :   LSTPRIM.H
*             Format :        Source C
*             Version  :      19/9/96
*             Programmeurs :  Delozanne, Jacoboni , Futtersack
*             Contenu :       Déclaration des primitives du TDA LISTE
*                              (correspondant au cours 1).
*************************************************************/
#ifndef _LSTPRIM_H        /* pour l'inclusion conditionnelle */
#define  _LSTPRIM_H

#include "eltprim.h"      /* le TDA Liste utilise le TDA ELEMENT */

#include "lstsdd.h"       /* inclusion du fichier où est indiquée la structure
                          de donnée retenue pour réaliser le TDA LISTE/*

/***********************************/
/* Déclaration des primitives du TDA LISTE*/
/* primitive de test */
              /* teste si la liste est vide */
    bool ListeVide (LISTE);

/* primitives d'accès */

    /* accès aux éléments */
              /*retourne l'élément à la position p dans la liste,
               sans modifier la liste, retourne l'élément vide si la liste
               est vide ou si la position est mauvaise */
    ELEMENT ListeAcceder(POSITION,LISTE);

    /* accès aux positions */
              /* retourne la position qui suit la position du dernier élément
               de la liste*/
    POSITION ListeSentinelle(LISTE);

              /* retourne la première position de la liste si la liste est
              non vide ou ListeSentinelle */
    POSITION ListePremier(LISTE);

              /* retourne la position qui suit la position paramètre dans la
              liste si la liste est non vide ou ListeSentinelle */
    POSITION ListeSuivant(POSITION,LISTE);

/* primitives de modification de la liste */
              /* modifie la liste en insérant l'élément à la position ;
              retourne faux si la liste est pleine ou si la position est
              mauvaise */
    bool ListeInserer (ELEMENT,POSITION,LISTE);

              /* supprime de la liste l'élément dont la position est passée en
              paramètre ; retourne faux si la liste est vide */
    bool ListeSupprimer (POSITION, LISTE);

/* primitives de création et de destruction */
              /* crée et retourne une liste vide en lui allouant de la mémoire
              dynamique*/
    LISTE ListeCreer(void);

              /* libére la mémoire dynamique allouée pour la liste */
    void ListeDetruire(LISTE);

#endif
```