TP6 : Liste doublement chaînée

Commencez par créer un répertoire TP6 à l'intérieur de votre répertoire LIFAPSD. L'objectif de ce TP est d'écrire une nouvelle implémentation de liste chaînée, différente de celle vue en cours et en TD, mais ayant la même interface (mêmes services proposés aux utilisateurs du module). L'implémentation que vous allez écrire est celle d'une **liste doublement chaînée**, dans laquelle :

- chaque cellule contient un pointeur sur la cellule suivante et un pointeur sur la cellule précédente,
- la classe Liste contient un pointeur sur la première cellule et un pointeur sur la dernière cellule.

Ainsi, il est possible de parcourir la liste dans les deux sens, et d'ajouter un élément en tête et en queue de liste en temps constant.

Exercice 1 : Initialisation et Makefile

a. Récupérez sur le site de l'UE les fichiers ElementL.h, ElementL.cpp et Liste.h. Enregistrez-les dans votre répertoire TP6. Toujours dans ce répertoire, créez avec gedit un nouveau fichier que vous appellerez Liste.cpp. Ecrivez-y les #include requis, puis le code du constructeur. Enregistrez le fichier mais ne le fermez pas, vous y reviendrez plus tard.

b. Créez un nouveau fichier main.cpp, et écrivez-y un programme principal minimal, qui teste la création d'une instance de la classe Liste.

c. Créez un fichier Makefile et écrivez-y les lignes nécessaires pour compiler votre programme. Inspirez-vous de ce que vous avez fait lors du TP précédent. Dans le terminal, compilez votre code (make), exécutez-le et corrigez-le si nécessaire.

Exercice 2 : Implémentation des fonctionnalités de base du module

Définissez dans Liste.cpp les fonctionnalités ci-dessous, en respectant les prototypes déclarés dans le .h. Remarque importante : Testez *au fur et à mesure* que vous ajoutez une fonctionnalité : appelez-la dans le main, enregistrez tous vos fichiers, recompilez en utilisant la commande `make` et exécutez le programme.

a. affichage de la liste (de droite à gauche, et de gauche à droite),

b. test si la liste est vide,

c. ajout d'un élément en tête de liste,

d. ajout d'un élément en queue de liste,

e. suppression de l'élément de tête,

f. vider la liste,

g. le destructeur,

h. renvoie du nombre d'éléments,

i. accès au i-ème élément,

j. modification du i-ème élément.

Exercice 3 : Fonctionnalités plus avancées

a. Testez dans votre main ce qu'il se passe lorsqu'on fait a = b quand a et b sont deux listes chaînées, et qu'une liste est libérée de la mémoire ensuite. Expliquez. Implémentez alors l'opérateur d'affectation qui recopie le contenu d'une liste dans l'instance. Le contenu précédent de la liste doit être libéré.

b. Implémentez la procédure de recherche d'un élément dans une liste quelconque.

c. Implémentez la procédure d'insertion d'un élément en i-ème position.

d. Implémentez la procédure de tri d'une liste doublement chaînée, en utilisant l'algorithme de tri par insertion.