8INF259: STRUCTURES DE DONNÉES

TP1 : Dossier de clients

Travail seul ou en équipe de 2

À remettre au plus tard le 18 février 2019 à 23h55

1.Objectif

Le principal but de ce travail est de se familiariser avec les pointeurs et les classes dans le langage C++, **sans avoir recours à la STL de ce langage**. Il est donc demandé, dans ce TP, de concevoir un programme, écrit en C++, qui crée une liste chaînée en mémoire centrale à partir des données contenues dans un fichier texte.

2. Mise en contexte

Une compagnie de livraison, *Messaginc*, offre un service de livraison de messages entre les habitants des petites municipalités. Le fonctionnement est simple. Chaque inscrit peut utiliser le service afin d'envoyer un court message de 80 caractères à une autre adresse du même village. Chaque message rapporte 1 dollar à l'entreprise.

Messaginc conserve la liste de ses clients dans un fichier texte "CLIENT.txt". Un nom de client est associé à une adresse.

Messaginc conserve également dans un fichier "HISTORIQUE.txt" l'historique des messages, contenant le nom de la source du message, le nom du destinataire du message ainsi que le message lui-même.

Messaginc vous demande de faciliter la consultation de l'historique des messages en concevant un programme qui utilise une liste chaînée pour rassembler les messages, les clients et les adresses.

3. Travail à effectuer

Il est demandé de programmer une classe DossierClient, qui crée, à partir des données contenues dans les deux fichiers textes, une liste chaînée (décrite plus loin). Par la suite, le programme doit pouvoir faire certaines opérations sur les données chargées en mémoire. Les opérations à effectuer par cette liste chaînée sont les suivantes:

1. Charger ou sauvegarder la liste des clients et leur historique.
2. Ajouter ou supprimer un client de la liste (une adresse).
3. Afficher un message et l'ajouter à l'historique.
4. Afficher le nombre de messages échangés entre deux clients
5. Afficher le client qui envoie le plus de messages.
6. Afficher la rue la plus payante.

4. Format des fichiers textes

Les informations contenues dans le fichier texte "CLIENT.txt" ont le format suivant :

```
Village
Rue 1
Numero 1
Nom du client 1
Rue 2
Numero 2
Nom du client 2
.
.
Rue N
Numero N
Nom du client N
```

- "Village" est une chaîne de caractères identifiant le nom du village.
- "Rue" est une chaîne de caractères identifiant une rue.
- L'entier "Numero" indique l'adresse du client;
- "Nom du client" est une chaîne de caractères identifiant le nom du client.

Les informations contenues dans le fichier texte "HISTORIQE.txt" ont le format suivant :

```
Client source 1
Client destinataire 1
Message 1
Client destinataire 2
Message 2
.
.
.
Client destinataire N
Message N
&
Client source 2
Client destinataire 1
Message 1
Client destinataire 2
Message 2
.
.
.
Client destinataire N
Message N
```

- "Client source" est une chaîne de caractères identifiant le nom du client source.
- "Message" est une chaîne de 80 caractères.
- "Client destinataire" est le nom du destinateur du message;
- Le caractère "&" sert à séparer les clients.

Votre programme lira à partir d'un fichier texte "TRANSACTION.txt", les transactions à effectuer par le gestionnaire dont le format est: op [param1, param1, ...]. Les opérations possibles sont comme ci-dessous:

- X : supprimer un client X de la liste chaînée.
+ X : ajouter un client X à la liste chaînée.
= X Y M : ajouter un message M envoyé du client X au client Y.
& X Y : afficher le nombre de messages échanger entre le client X et Y.
! : Afficher le client qui envoie le plus de messages.
$: Afficher le nom de la rue la plus payante.
O CLIENT HITORIQUE: ouvre le dossier client "CLIENT" et l'historique "HISTORIQUE".
S CLIENT HITORIQUE: enregistre le dossier client "CLIENT" et l'historique "HISTORIQUE".

5. Structures des données

Les structures de données à utiliser sont les suivantes:

```
struct Message
{
        char destinataire[50];
        char message[80];
        message *suivant;
};

struct Client
{
        char rue[50];
        int numero;
        char nom[50];
        Message *listeMessage;
        Client *suivant;
};

class DossierClient{
   private:
      Client * tete; // début de la liste chaînée.
   public:
      DossierClient ();
      ~ DossierClient ();

      void Ouvrir(char* fichierClient, char* fichierHistorique);
      void Sauvegarder (char* fichierClient, char* fichierHistorique);

      void AjouterClient (char* NOM, char* RUE, int numero);
      void SupprimerClient (char* NOM);
      void AjouterMessage(char* nomClient, char* nomDestination, char* message);

      int NombreEchange(char* X, char* Y);
      char* MeilleurClient ( ) const;
      char* RuePayante ( ) const ;
};
```

6. Exigences et Livrables

Afin de simplifier la correction du devoir, vous devez respecter ces quelques consignes:

- Votre devoir doit obligatoirement fonctionner (compilation et exécution) avec l'environnement de programmation Visual Studio 2015 ou 2017.
- Vous devez remettre tous vos fichiers d'entêtes et vos fichiers sources (.h et .cpp);
- Vous pouvez remettre la structure de projet contenant les fichiers .sln et .vxproj. Cependant, vous devrez vous assurer d'avoir nettoyé votre solution avec la remise (aucun exécutable et aucun fichier compiler).
- Vous devez déposer votre devoir via le site Moodle du cours dans la section prévue à cet effet.

7. Guide de correction

Présenté à titre indicatif seulement

Guide de correction		
Fichier de déclaration (.h)	**Déclaration des structures** **Déclaration de la classe DossierClient**	/1
Gestion des pointeurs	**Constructeur de Liste chaînée** **Destructeur de Liste chaînée** **Fonctions pour ajouter et supprimer d'une liste**	/3
Lecture et écriture	**Création de la liste chaînée à partir des fichiers** **Exécution des fonctionnalités à partir d'un fichier** **Écriture d'un nouveau fichier à partir de la liste chaînée**	/3
Fonction de recherche	**Fonction "Echange"** **Fonction "MeilleurClient"** **Fonction "RuePayante"**	/3
Lisibilité et documentation		/-0.5
Respect des consignes		/-0.5
	Total	/10