MARTIN-PARISOT Céline
ANKI Rayan

C/C++
L3 APP - LSI2

RAPPORT : Projet n°2 C++

SOMMAIRE :

Ce TP a pour but la gestion de dossiers bancaires. Pour cela nous devons implémenter une application console simple sans utilisation de composant graphique.

I°) Menu Principal

A - Affichage

Nous avons choisi de créer une classe menu qui représente l'affichage de notre application.

Cette classe possède deux attributs privés : le choix de l'utilisateur et le nombre de records enregistrés.
Elle a également plusieurs fonctions publiques : constructeur par défaut et d'initialisation, destructeur, getter, setter. Nous avons implémenté une fonction permettant d'initialiser le menu avec le nombre de comptes enregistrés présents et une fonction qui permet d'incrémenter le nombre de records enregistrés.

Nous avons ensuite une fonction "display_menu" qui affiche le menu à l'utilisateur et applique son choix d'action à l'aide d'un switch case.

B - Search()

Dans cette classe nous avons également implémenté la fonction Search(), qui permet soit (la première fois) d'afficher le nombre de records, soit (ensuite) de rechercher un compte grâce à un id (dans le nom du fichier) de compte et de l'afficher.

Si c'est le premier appel de cette fonction, on affiche le nombre de records.

Sinon, on commence par demander l'id du compte à rechercher.
On déclare ensuite une variable "fichier" initialisée à 0, qui permet de garder l'id du fichier dans lequel on cherche le compte correspondant au numéro entré par l'utilisateur. Puis on essaie d'ouvrir le fichier et, si on y arrive, on affiche son contenu.

C - Show() et ShowAll()

Enfin, nous avons implémenté deux fonctions d'affichage des records : Show() et ShowAll().
Show() demande à l'utilisateur quel numéro de compte il souhaite consulter. L'utilisateur a aussi à entrer le nom et le prénom du détenteur du compte. Ensuite, s'il existe, le solde de ce compte est affiché. Pour ce faire, on boucle sur tous les fichiers de compte enregistrés, on stocke les informations et on compare le numéro de compte, le nom et le prénom avec ceux entrés par l'utilisateur pour trouver le compte à afficher.
ShowAll() affiche tous les comptes enregistrés. Cette fonction n'était pas requise mais il nous a semblé utile de l'ajouter pour avoir une vision globale de tous les enregistrements (comme on les a divisés par fichiers distincts).

Nous avons en effet choisi de créer un fichier par compte enregistré. Pour simplifier les noms de fichiers, ils sont comme suit : “record_***$id***.txt”. id est une variable qui commence à 0 puis est incrémentée à chaque ajout de compte. Cela simplifie l’affichage car les numéros de comptes peuvent être très grands (compte n°1234567654567654456). Ces fichiers de compte sont stockés dans un dossier “bank_records”.

II°) Classe Bank

La classe Bank représente la gestion des données de notre application.
Elle possède plusieurs attributs privés représentant les informations des comptes :

```
private:
    std::string firstname;
    std::string lastname;
    std::string account_number;
    std::string telephone;
    std::string balance;
    int id_record;
```

“id_record” représente le numéro du fichier d’enregistrement du compte.

Cette classe possède également un constructeur par défaut, un constructeur d’initialisation, un destructeur. Elle possède ensuite les fonctions de gestion de données : Add(), Edit() et Delete().

A - Add(int number_of_records)

La fonction Add permet d’ajouter un compte à notre base de fichiers. Elle prend en paramètre le nombre d’enregistrements actuels afin d’adapter l’id, et donc le nom du fichier, en conséquence.
On commence par créer le nom du fichier avec la variable “number_of_records”. Grâce au nom du fichier, celui-ci sera créé dans le dossier “bank_records”.

On crée ensuite le fichier et, s’il n’a pas pu être créé, on renvoie une instance de Bank vide.
On déclare un objet Bank auquel on ajoute les attributs de compte entrés par l’utilisateur.

En vérifiant que le fichier est bien ouvert, on écrit les attributs du compte dans ce dernier.

Enfin, on ferme le fichier et on renvoie l’instance de Bank créée.

B - Edit()

Edit permet de modifier les informations du compte demandé par l’utilisateur via l’id de son fichier.

On commence par demander l’id du fichier à modifier à l’utilisateur. On affiche ensuite ce fichier afin de visualiser les informations déjà présentes. Puis, on stocke dans des variables les nouvelles informations entrées par l’utilisateur : numéro de téléphone et solde.

On ouvre le fichier pour écrire dedans : on réécrit les premières informations (numéro de compte, nom, prénom) puis on écrit les nouvelles informations.

Enfin, on ferme le fichier.

C - Delete()

Delete permet de supprimer le fichier du compte demandé par l'utilisateur via son id.

On commence par demander l'id du fichier à supprimer à l'utilisateur. On affiche ensuite ce fichier afin de demander confirmation pour sa suppression. Si l'utilisateur souhaite effectivement supprimer ce fichier, on le supprime grâce à la fonction remove de la bibliothèque standard. Cette fonction renvoie un int pour décrémenter le nombre de records total s'il y a suppression.

III°) Protections

Nous avons également décidé de protéger les entrées utilisateurs avec des fonctions (demanderStringEtVerifier et demanderNombreEtVerifier) dans la classe Menu. Ainsi, les numéros sont bien des nombres et les chaînes de caractères ne contiennent que des lettres. (nous n'avons pas eu le temps de les insérer partout dans le code mais elles sont implémentées pour le choix dans le menu par exemple). Le ctrl^d est également protégé en vérifiant std::cin.eof().

IV°)Execution

Il suffit de run le makefile en écrivant "make" dans le terminal puis "./bank" pour exécuter le programme. Ensuite, il n'y a plus qu'à suivre le menu.