Langage C 2020-2021 : Projet

Objectif

Le projet vise à la mise en place de différentes structures et fichiers permettant de modéliser un système bancaire. Il capitalise les différentes compétences présentées en TP et sa réalisation doit démontrer une compréhension des fonctionnements basiques de langage C (compilation, types de données) et plus avancés (pointeurs, débogueur).

Modalités

- Le travail s'effectue en binôme
- Le travail est à rendre lors de la séance du 13 décembre
- Une courte présentation de vos travaux sera effectuée à la fin de la séance
- L'archive contenant le code ainsi qu'une brève présentation est à rendre au même moment

Première partie

Les transactions seront définies à l'aide d'une date, d'un montant et de deux noms. Elles seront stockées dans un fichier binaire qui correspond à chaque compte (*ex: 102.dat*). Le fichier est découpé en deux parties. Tout d'abord, un `entete` qui contient une `date` ainsi que le solde correspondant à cette date. Ensuite, la liste de toutes les `transactions`. Un client peut mettre à jour l'entête pour qu'il fasse référence à une date donnée à l'aide de la fonction dédiée.

Structure `date`:

1. Définir un type structure `date` qui contient trois membres entiers : jour, mois, année.
2. Écrire une fonction `void (date *d)` qui attribue la date actuelle au pointeur (*note: les fonctions de la librairie `<time.h>` vous seront utiles*).

Structure `compte` :

3. Définir un type structure `compte` qui contient les membres suivants : le nom ainsi que le numéro de compte correspondant (considérés uniques).

Gestion des fichiers :

4. Écrire une fonction `void ouvrir(FILE **f, char nom[])` qui ouvre le fichier donné ou le crée sinon. Écrire une fonction `void fermer(FILE*f)` qui ferme le fichier.

Structure **`transaction`** :

5. Définir un type structure `transaction` qui contient les membres suivants :
    - `date` : la date de la transaction
    - `montant` : le montant de la transaction
    - `label` : un texte de 30 lettres décrivant la transaction
    - `nom` : émetteur si la transaction est positive (réception d'argent) ou receveur si la transaction est négative (envoi d'argent)

    Écrire la fonction **`transaction creation_transaction(date, float, char*, char*)`** qui crée une nouvelle transaction à l'aide des valeurs de chacun des membres.
6. Écrire une fonction **`int ajout_transaction(FILE*, *transaction)`** qui ajoute une transaction au fichier et renvoie le résultat de l'écriture. *Note: vous pouvez écrire la structure directement dans le fichier via* `fwrite()` .

Structure **`entete`** :

7. Définir un type structure `entete` qui contient les une date et un solde. Définir sa fonction de création **`creation_entete(date, float)`** associée.
8. Définir la fonction **`FILE* creation_fichier(entete, char*)`** qui crée un fichier avec un **`entete`** donné.
9. Définir la fonction **`mise_a_jour_solde(FILE*, date)`** qui met à jour le solde de l'entête à la date donnée à l'aide de l'entête précédent et de la liste des transactions. *Note: la fonction* `fseek` *peut permettre de passer l'entête.*

Deuxième partie

Dans un second temps, la banque connaît les comptes et les utilisateurs. Le lien entre eux donné par la structure **`compte`** est stocké dans un fichier binaire (*ex: banque.dat*). La banque offre plusieurs opérations à l'utilisateur via un menu. Une des options cloture la session. Un exemple de fonction menu vous est donné en fin de sujet. Une fonction **`main`** faisant appel aux fonctions `ouvrir`, `fermer` et `menu` pourra permettre de tester votre programme.

1. Définir une fonction **`int creer_utilisateur(char*)`** qui crée le fichier de compte correspondant de numéro aléatoire et met à jour le répertoire de la banque.
2. Définir une fonction **`int compte_de(char*)`** qui renvoie le numéro de compte associé à un nom.
3. Définir une fonction **`float solde_de(char*, date)`** qui renvoie le solde de compte associé à un nom à une date donnée.
4. Définir une fonction **`int mise_a_jour_solde(char*, date)`** qui met à jour l'entête d'un utilisateur pour une date donnée.
5. Définir une fonction **`int virement_de_compte_a_compte(int, int, date, float)`** qui crée deux transactions dans chacun des fichiers, une en émission, l'autre en réception. Définir une fonction **`int virement_de_a(char*, char*, date, float)`** qui fait la même chose via les noms.
6. Définir une fonction **`int imprimer_releve (char*, int)`** qui imprime le relevé de compte d'un usager pour un mois donné.

```c
void menu(FILE *fic)
{
    char choix;
    do {
        printf("\n\nAjouter un nouveau client.............: A\n");
        printf("Lister tous les comptes de clients.....: L\n");
        printf("Relevé d'un compte client..............: R\n");
        printf("Virement depuis un compte client.......: V\n");
        printf("Mise a jour du solde d'un client.......: M\n")
        printf("Quitter................................: Q\n");
        printf(" votre choix: ");
        rewind(stdin);
        scanf("%c",&choix);
        switch(choix)
        {
            case 'a':
            case 'A': ajout(...);
                ...
        }
    } while (choix != 'q' && choix != 'Q');
}
```