et-picsou

March 11, 2024

1 Le coffre fort d'Oncle Picsou

Avec notre mini-projet de gestion de comptes bancaires, vous apprendrez à créer et gérer des comptes bancaires, à faire des dépôts et des retraits, et même à transférer de l'argent entre différents comptes - tout cela en utilisant le langage de programmation C. Bien sûr, cela peut ne pas sembler aussi excitant que de nager dans une mer de pièces d'or, mais n'oubliez pas que même Oncle Picsou doit garder un œil sur ses finances. Et qui sait, une fois que vous aurez maîtrisé ces compétences, peut-être que votre propre coffre-fort rempli d'or ne sera pas si loin… Alors, enfilez votre haut-de-forme et sautez dans l'aventure !

1.1 Création de comptes (30 minutes) :

1. **Définition de la structure `Compte`** : Cette étape concerne la définition d'une structure pour représenter un compte bancaire. La structure doit contenir un nom (qui est un tableau de caractères), un numéro de compte (qui est un entier) et un solde (qui est un flottant). (**5 minutes**)
2. **Création d'une variable pour le prochain numéro de compte** : Vous devez garder une trace du prochain numéro de compte à attribuer. Pour cela, définissez une variable globale qui commence à 1 et que vous allez incrémenter chaque fois que vous créez un nouveau compte. (**5 minutes**)
3. **Création d'une fonction pour créer un nouveau compte** : Cette fonction sera responsable de la création de nouvelles instances de la structure de compte. Elle devrait : (**5 minutes**)
   - Prendre en entrée le nom du titulaire et le solde initial.
   - Créer une nouvelle instance de la structure `Compte`.
4. Copier le nom donné dans le champ correspondant de la structure de compte. (**5 minutes**) La fonction `strncpy` est utilisée pour copier une certaine quantité de caractères d'une chaîne de caractères source à une chaîne de caractères de destination. La syntaxe de cette fonction est la suivante :

   ```
   strncpy(destination, source, nombreDeCaracteres);
   ```

   - `destination` est la chaîne de caractères où la source sera copiée.
   - `source` est la chaîne de caractères qui sera copiée à destination.
   - `nombreDeCaracteres` est le nombre maximal de caractères à copier de la source à destination.

Dans le contexte de votre projet, vous utilisez `strncpy` pour copier le nom donné dans le champ `nom` de la structure de compte.

Voici comment cela pourrait se passer :

```
strncpy(nouveauCompte.nom, nom, sizeof(nouveauCompte.nom));
```

- `nouveauCompte.nom` est la destination où le nom sera copié. Il représente le champ `nom` dans la structure `nouveauCompte` du compte bancaire.
- `nom` est la chaîne source qui représente le nom du titulaire du compte.
- `sizeof(nouveauCompte.nom)` est le nombre maximal de caractères à copier. Il est égal à la taille du champ `nom` dans la structure `nouveauCompte`.

Il est important de noter que `strncpy` ne termine pas automatiquement la chaîne de caractères de destination avec un caractère nul si la source a `nombreDeCaracteres` caractères ou plus. Pour cette raison, il est souvent recommandé d'ajouter manuellement un caractère nul à la fin de la chaîne de caractères de destination pour s'assurer qu'elle est correctement terminée :

```
nouveauCompte.nom[sizeof(nouveauCompte.nom) - 1] = '\0';
```

Cette instruction garantit que le dernier caractère du tableau `nom` est un caractère nul, ce qui signifie la fin de la chaîne de caractères dans le langage C.

5. Compléter la fonction `creerCompte()`: (**5 minutes**)
   - Attribuer le prochain numéro de compte disponible à la structure de compte, puis incrémenter cette valeur pour le prochain compte.
   - Initialiser le solde de la structure de compte avec le solde initial donné.
   - Renvoyer la nouvelle instance de la structure `Compte`.
6. **Test de la fonction de création de compte** : Une fois la fonction de création de compte mise en place, vous devriez la tester pour vous assurer qu'elle fonctionne correctement. Dans la fonction `main`, créez quelques comptes en utilisant la fonction de création de compte. Pour chaque compte, affichez les informations du compte (nom, numéro de compte, et solde) à l'écran pour vérifier qu'elles sont correctes. (**5 minutes**)

1.2 Dépôts : (25 minutes)

1. **Créer des comptes de test** : Avant d'implémenter la fonction de dépot, vous devez avoir quelques comptes sur lesquels effectuer des dépôts. Créez plusieurs comptes avec différents numéros de compte et soldes initiaux. Vous pouvez le faire en appelant votre fonction `creerCompte` plusieurs fois avec différentes valeurs d'arguments. (**5 minutes**)

2. **Préparation du tableau de comptes** : Ensuite, ajoutez ces comptes à un tableau de structures Compte. Ce tableau sera utilisé comme argument lors de l'appel à la fonction depot. (**5 minutes**)

3. **Définition de la fonction** : (**5 minutes**) Commencez par définir une fonction `depot`. Cette fonction doit prendre quatre paramètres :

   - Un pointeur vers un tableau de structures Compte.
   - Le nombre total de comptes existants (pour savoir jusqu'où parcourir le tableau).
   - Le numéro du compte sur lequel le dépôt doit être effectué.
   - Le montant à déposer.

4. **Création de la boucle de parcours** : (**5 minutes**) À l'intérieur de la fonction `depot`, vous devez parcourir le tableau de comptes. Vous pouvez le faire avec une boucle `for` qui commence à l'indice 0 et se termine à l'indice `nbComptes - 1`.

5. **Recherche du compte** : À chaque itération de la boucle, vérifiez si le numéro du compte courant (i.e., `comptes[i].numero`) correspond au numéro de compte passé en argument à la fonction.

6. **Effectuer le dépôt** : Si vous trouvez un compte qui correspond, ajoutez le montant du dépôt au solde de ce compte. N'oubliez pas de mettre à jour le solde du compte dans la structure elle-même.

7. **Communication avec l'utilisateur** : Après avoir effectué le dépôt, affichez un message à l'utilisateur pour confirmer l'action. Par exemple : "Dépôt de XX euros effectué sur le compte numéro YY. Nouveau solde : ZZ euros."

8. **Gérer le cas d'un compte non trouvé** : Si vous parcourez tout le tableau de comptes sans trouver de correspondance pour le numéro de compte, affichez un message indiquant que le compte n'existe pas. Vous pouvez effectuer cette vérification après la boucle `for`, car si vous atteignez ce point sans avoir trouvé le compte, cela signifie qu'il n'existe pas.

9. **Test de la fonction de dépôt** : Une fois la fonction mise en place, vous devriez la tester pour vous assurer qu'elle fonctionne correctement. Dans la fonction `main`, appelez la fonction depot avec différents scénarios : Une fois que vous avez un tableau de comptes prêt, vous pouvez commencer à tester la fonction depot. Pour un test complet, vous devriez envisager différents scénarios. Par exemple: (**5 minutes**)

    - Faire un dépôt sur un compte existant avec un montant positif.
    - Essayer de faire un dépôt sur un compte qui n'existe pas.
    - Faire un dépôt avec un montant nul ou négatif.
    - Dans chaque scénario, appelez la fonction depot avec le tableau de comptes, le numéro de compte approprié et le montant du dépôt. Notez que le dépôt ne devrait réussir que dans le premier scénario.

1.3 Retraits (20 minutes) :

Passons au retrait d'argent sur un compte bancaire. Nous allons suivre des étapes similaires à celles que nous avons utilisées pour la fonction de dépôt :

1. **Définition de la fonction** : (**5 minutes**) Commencez par définir une fonction `retrait`. Cette fonction doit prendre les mêmes quatre paramètres que la fonction `depot` :

    - Un pointeur vers un tableau de structures Compte.
    - Le nombre total de comptes existants.
    - Le numéro du compte duquel le retrait doit être effectué.
    - Le montant à retirer.

2. **Création de la boucle de parcours** : (**2 minutes**) À l'intérieur de la fonction `retrait`, vous devez également parcourir le tableau de comptes.

3. **Recherche du compte** : (**2 minutes**) À chaque itération de la boucle, vérifiez si le numéro du compte courant correspond au numéro de compte passé en argument à la fonction.

4. **Effectuer le retrait** : (**2 minutes**) Si vous trouvez un compte qui correspond, vérifiez que le solde du compte est suffisant pour couvrir le montant du retrait. Si c'est le cas, soustrayez le montant du retrait du solde du compte. Si ce n'est pas le cas, affichez un message indiquant que le retrait n'est pas possible en raison d'un solde insuffisant.

5. **Communication avec l'utilisateur** : (**2 minutes**) Après avoir effectué le retrait, affichez un message à l'utilisateur pour confirmer l'action, par exemple : "Retrait de XX euros effectué sur le compte numéro YY. Nouveau solde : ZZ euros."

6. **Gérer le cas d'un compte non trouvé** : (**2 minutes**) Si vous parcourez tout le tableau de comptes sans trouver de correspondance pour le numéro de compte, affichez un message indiquant que le compte n'existe pas.

7. **Test de la fonction de retrait** : (**5 minutes**) Une fois que vous avez mis en place la fonction `retrait`, vous devriez la tester pour vous assurer qu'elle fonctionne correctement. Voici quelques scénarios de test possibles :

   - Faire un retrait d'un montant inférieur au solde du compte.
   - Essayer de faire un retrait d'un montant supérieur au solde du compte.
   - Essayer de faire un retrait sur un compte qui n'existe pas.
   - Faire un retrait d'un montant nul ou négatif.

   Dans chaque scénario, appelez la fonction `retrait` avec le tableau de comptes, le numéro de compte approprié et le montant du retrait. Vérifiez que la fonction se comporte comme prévu.

1.4 Affichage du solde (10 minutes) :

Passons à l'affichage du solde d'un compte bancaire. Voici les étapes que vous pouvez suivre pour implémenter et tester cette fonctionnalité :

1. **Définition de la fonction** (**5 minutes**) :
   - Commencez par définir une fonction `afficherSolde`. Cette fonction doit prendre deux paramètres :
     - Un pointeur vers un tableau de structures Compte.
     - Le nombre total de comptes existants.
     - Le numéro du compte dont le solde doit être affiché.
2. **Création de la boucle de parcours** :
   - À l'intérieur de la fonction `afficherSolde`, vous devez parcourir le tableau de comptes.
3. **Recherche du compte** :
   - À chaque itération de la boucle, vérifiez si le numéro du compte courant correspond au numéro de compte passé en argument à la fonction.
4. **Afficher le solde** :
   - Si vous trouvez un compte qui correspond, affichez le solde de ce compte.
5. **Gérer le cas d'un compte non trouvé** :
   - Si vous parcourez tout le tableau de comptes sans trouver de correspondance pour le numéro de compte, affichez un message indiquant que le compte n'existe pas.
6. **Test de la fonction d'affichage du solde** (**5 minutes**) :
   - Une fois que vous avez mis en place la fonction `afficherSolde`, vous devriez la tester pour vous assurer qu'elle fonctionne correctement. Voici quelques scénarios de test possibles :

  - Afficher le solde d'un compte existant.
  - Essayer d'afficher le solde d'un compte qui n'existe pas.
- Dans chaque scénario, appelez la fonction `afficherSolde` avec le tableau de comptes et le numéro de compte approprié. Vérifiez que la fonction se comporte comme prévu.

1.5 BONUS: Transfert entre deux comptes (15 minutes):

1. **Définition de la fonction** (**5 minutes**) :
   - Commencez par définir une fonction `transferer`. Cette fonction doit prendre quatre paramètres :
     - Un pointeur vers un tableau de structures Compte.
     - Le nombre total de comptes existants.
     - Le numéro du compte source.
     - Le numéro du compte destinataire.
     - Le montant à transférer.
2. **Création de la boucle de parcours** :
   - À l'intérieur de la fonction `transferer`, vous aurez besoin de deux boucles de parcours (ou d'une boucle unique avec deux vérifications à chaque itération) pour trouver à la fois le compte source et le compte destinataire.
3. **Recherche des comptes** :
   - À chaque itération de la boucle, vérifiez si le numéro du compte courant correspond au numéro du compte source ou du compte destinataire passé en argument à la fonction.
4. **Vérification du solde du compte source** :
   - Une fois que vous avez trouvé le compte source, vérifiez si son solde est suffisant pour couvrir le montant du transfert. Si ce n'est pas le cas, informez l'utilisateur et terminez la fonction.
5. **Transfert d'argent** :
   - Si le solde du compte source est suffisant, déduisez le montant du transfert de ce compte et ajoutez-le au compte destinataire.
6. **Communication avec l'utilisateur** :
   - Après avoir effectué le transfert, affichez un message à l'utilisateur pour confirmer l'action. Par exemple, "Transfert de XX euros effectué du compte numéro YY au compte numéro ZZ. Nouveaux soldes : YY: AA euros, ZZ: BB euros."
7. **Gérer le cas de comptes non trouvés** :
   - Si vous parcourez tout le tableau de comptes sans trouver de correspondance pour le numéro de compte source ou le numéro de compte destinataire, imprimez un message indiquant que le compte n'existe pas.
8. **Test de la fonction de transfert** (**5 minutes**) :
   - Une fois que vous avez mis en place la fonction `transferer`, vous devriez la tester pour vous assurer qu'elle fonctionne correctement. Voici quelques scénarios de test possibles :
     - Transférer de l'argent entre deux comptes existants.
     - Essayer de transférer de l'argent d'un compte qui n'existe pas.
     - Essayer de transférer de l'argent vers un compte qui n'existe pas.
     - Essayer de transférer un montant plus élevé que le solde du compte source.
   - Dans chaque scénario, appelez la fonction `transferer` avec le tableau de comptes et les numéros de compte appropriés. Vérifiez que la fonction se comporte comme prévu.