Objectif et description

Afin de démontrer l'acquisition des compétences visées par les cours C11 et C21, vous devez mettre en pratique tout ce que vous connaissez en réalisant un programme qui permet de faire la gestion et le partage des dépenses d'un groupe solidaire de participants. Ce programme doit être en mesure de proposer un scénario de règlement finale de manière à répartir en part égale (la quote-part), la charge totale des dépenses du groupe, en fonction des dépenses de chacun.

Contexte de réalisation et de remise

Ce travail individuel doit être fait en laboratoire et devra être complété à l'extérieur des cours. Remettre un dossier avec vos sources ainsi que l'exécutable et les autres fichiers nécessaires sur LÉA sous la remise ***TP5***.

Spécifications des fonctions du programme

1. Ajouter un participant (fonction c++ : CMD_*AjouterUnParticipant*)
   - Lire toutes les informations sur le participant.
   - Initialiser le nombre de dépenses à zéro.
   - Enregistrer le nouveau participant.

2. Afficher le dossier d'un participant (fonction c++ : CMD_*AfficherUnParticipant*)
   - Afficher toutes les informations du participant identifié par son numéro.
   - Afficher la liste des dépenses ainsi que les dates et le total des dépenses.

3. Ajouter des dépenses (fonction c++ : CMD_*AjouterDesDepenses*)
   - Afficher les informations du participant identifié par son numéro.
   - Offrir à l'usager (à répétition) d'ajouter une dépense jusqu'à un maximum de 10.
   - Lire les informations d'une dépense.
   - La date d'une dépense doit être déterminée automatiquement par le système.
   - Rafraichir la liste des dépenses et son total en fonction des ajouts.
   - Enregistrer (mettre à jour) les dépenses du participant.

4. Supprimer des dépenses (fonction c++ : CMD_*SupprimerDesDepenses*)
   - Afficher les informations du participant identifié par son numéro.
   - Offrir à l'usager (à répétition) de supprimer une dépense par son numéro dans la liste.
   - Rafraichir la liste des dépenses et son total en fonction des retraits.
   - Enregistrer (mettre à jour) les dépenses du participant.

5. Afficher l'état des comptes (fonction c++ : CMD_*AfficherEtatDesComptes*)
   - Afficher la date, le nombre de parts et le montant actuel de la quote-part.
   - Afficher pour chaque participant le total de ses dépenses, s'il est en avance (CYAN) ou en retard (ROUGE) en fonction de la quote-part, et la différence entre ses dépenses et cette quote-part (en valeur absolue).
   - Afficher le grand total des dépenses de tous.

6. Afficher un scénario de règlement (fonction c++ : *CMD_AfficherUnScenarioDeReglements*):
   - Afficher la date, le nombre de parts et le montant actuel de la quote-part.
   - Afficher pour chaque participant en avance (PeA) par rapport à la quote-part à payer, son nom, le total de ses dépenses, le total à recevoir.
   - Afficher aussi pour chaque PeA une liste de participants en retard (PeR) qui devraient le rembourser. Afficher pour chaque PeR de cette liste son nom, et le montant à rembourser.
   - Le montant total qu'un PeR doit rembourser à un ou des PeA ne peut dépasser la différence entre la quote-part à payer et le total de ses dépenses.

7. Supprimer tous les participants (fonction c++ : *CMD_SupprimerTousLesParticipants*)
   - Offrir à l'usager la possibilité de supprimer toutes les dépenses de tous les participants sans détruire les informations personnelles de chacun. Dans l'affirmative, enregistrer (mettre à jour) les dépenses de chaque participant.
   - Autrement, offrir la possibilité de supprimer tous les participants. Dans l'affirmative, détruire et recréer la base de données.

8. Quitter (fonction c++ : CMD_QuitterLeProgramme)
   - Afficher un message de votre cru ou autre chose …

Spécifications des informations de la base de données

1. Définir la base de données avec les éléments suivants :

   Adresse
   - Numéro civique (8 caractères max.)
   - Rue (30 caractères max.)
   - Ville (30 caractères max.)
   - Code Postal (7 caractères max.)
   - Téléphone (16 caractères max.)

   Dépense
   - Montant (0.00 … 99999.99 $ max)
   - Description (64 caractères max.)
   - Date et heure (time_t)

   Participant
   - Prénom (20 caractères max.)
   - Nom (20 caractères max.)
   - **Adresse**
   - Nombre actuel de dépense (0 à 10)
   - Liste de **Dépense** (10 max.)

2. Pour les tableaux de char, prévoir un caractère de plus pour la valeur butoir de fin de chaine.

Spécifications techniques

1. Le numéro du premier participant est 1.
2. Le montant maximum pour une dépense est : 99999.99 $.
3. Toutes les valeurs numériques monétaires, entrées ou calculées, sont conservées dans un « double » arrondi à une précision à deux décimales en tout temps.
4. La quote-part se calcule en additionnant toutes les dépenses d'abord que l'on divise ensuite par le nombre de participants.

5. Le système doit stocker les participants dans un fichier binaire contenant des structures ***Participant*** qui doivent être lues et écrites par accès direct. Voir les exemples dans les notes de cours ou le code Démo.
6. Le numéro du participant ne doit pas être stocké mais déduit par la position dans le fichier (BD).
7. Utiliser uniquement des tableaux de caractères (char prenom[…]) plutôt que des *strings* pour enregistrer une information de type texte dans le fichier.
8. Définir obligatoirement des constantes pour les dimensions (ex: const int LONGUEUR_PRENOM=21).
9. Trois fonctions doivent exister : db_size() pour obtenir le nombre de participants dans la BD, db_read() pour lire un participant dans la BD et db_write() pour enregistrer un participant dans la BD. Tous les accès au fichier (repartiteur.db) doivent se faire obligatoirement par ces fonctions. Voici les prototypes :

    - `int db_size();`
    - `void db_read(size_t noParticipant, Participant* p);`
    - `void db_write(size_t noParticipant, Participant* p);`

10. Le numéro d'un participant doit toujours être strictement positif (>0) dans le code, sauf au moment de calculer la position du participant dans la BD, dans les fonctions ci-haut mentionnées.
11. Le programme ne doit jamais conserver de participant(s) en mémoire sauf durant le traitement d'une commande.
12. Le fichier de base de données (repartiteur.db) doit être créé par le programme, dans le même dossier que l'exécutable, si celui-ci n'existe pas. Sinon, il est simplement ouvert. Le fichier est fermé seulement à la fin de l'exécution du programme.

Spécifications techniques de qualité du code

1. Le code du programme doit être distribué dans plusieurs modules (cpp et son header) et chaque module doit avoir un entête (un commentaire) qui explique la raison d'être de ce module. Vous devez avoir le fichier db.cpp.
2. Le programme doit pouvoir être compilé sans erreur et sans avertissement.

Spécifications techniques de qualité de l'exécution

1. Le programme doit valider le numéro du participant lorsque demandé à l'usager.
2. Aucune validation n'est exigée pour l'entrée de texte. Il faut conserver seulement le texte qui peut entrer dans la structure de données sans dépasser les limites.
3. Le programme doit être blindé, et le contraire pourrait entrainer de lourdes pénalités.
4. Les interactions avec l'usager doivent être parfaitement identique au programme du prof.