Exercices: Série 7 - Structures - ICC-C 2025-2026

Rappel : Pour faire lire des valeurs d'un fichier texte à votre programme, vous pouvez utiliser la redirection de l'entrée :

```
./exo < fichier.txt
```

Votre code ne saura pas qu'il lit depuis un fichier. Utilisez `scanf` et/ou `fgets` sans intercaler des messages avec `printf`. Affichez seulement le résultat à la fin du programme avec `printf`.

1. Échauffement

(La solution à cette section n'est pas fournie.)

Définissez une structure `vec3_t` qui représente un vecteur à 3 dimensions. Ses composantes sont des `double`.

Écrivez une fonction qui accepte deux `vec3_t` et renvoie leur produit vectoriel.

Rappel : le produit vectoriel peut être calculé avec

$$\boldsymbol{u} \times \boldsymbol{v} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$$

2. Introduction

Vous avez été engagé chez un nouvel opérateur téléphonique, les PTT. Vous devez écrire un programme qui produit des statistiques sur l'utilisation du téléphone par les clients. Pour ce faire, on vous donne un fichier texte `ptt.txt` (à télécharger sur Moodle) qui contient toutes les données requises.

D'abord, il contient trois entiers C, M et N. Ce sont respectivement le nombre de clients, d'appels téléphoniques et de tarifs spécifiques.

Les M lignes suivantes contiennent des informations sur des appels téléphoniques. Sur chacune de ces lignes, il y a :

- D'abord un numéro de client (un entier).
- La date de l'appel représentée comme un grand entier à 8 chiffres sans espaces, avec les quatre chiffres les plus significatifs qui correspondent à l'année, les deux suivants au mois, et les deux derniers au jour. Par exemple, le 27 mars 2026 s'écrit `20260327`.
- Un grand entier (aussi sans espaces) qui représente le numéro de téléphone appelé par le client. Heureusement, les numéros de téléphone en Suisse ont 9 chiffres (par exemple, `791112233`) et donc peuvent être stockés dans des `int` (qui vont jusqu'à 2 147 483 647).
- Enfin, un entier qui représente la durée de l'appel en minutes.

Dans ce même fichier, il y a ensuite des informations sur la facturation. Après les M lignes décrivant l'utilisation, suivent N lignes décrivant le coût des appels. Chaque ligne contient

- Un numéro de téléphone (même format qu'au dessus) et
- Un réel représentant le coût par minute en CHF pour appeler ce numéro.

Les numéros qui ne sont pas indiqués dans cette deuxième partie du fichier ont un tarif standard de 20 centimes par minute (soit 0.2 CHF).

3. Lecture

Définissez une structure `call_record_t` pour stocker des informations concernant un seul appel. Cette structure doit contenir 4 champs :

- `size_t no_client` : le numéro de client ;
- `int no_tel_appel` : le numéro appelé ;
- `int date` : la date (au format décrit ci-dessus) ;
- `int minutes`: la durée de l'appel, en minutes.

Vous aurez aussi besoin d'une structure `cost_t` pour les informations de facturation. Elle contient deux champs :

- `int tel` : le numéro de téléphone auquel s'applique ce tarif ;
- `int centimes_par_minute`: le tarif, en centimes par minutes.

Utilisez la redirection de l'entrée (avec `< ptt.txt`) pour lire le fichier texte donné, et lisez les données (avec `scanf`) dans deux tableaux.

Pour lire un coût, qui est exprimé en notation décimale dans le fichier, vous pouvez le lire comme un `double`, le multiplier par 100 puis l'arrondir avec `round` pour le transformer en centimes.

Questionnement : pourquoi n'utilise-t-on pas plutôt directement un `double chf_par_minute` ?

Challenge : pourriez-vous lire un coût sans passer par des `double` ?

4. Durée

Écrivez une fonction

```
void temps_total(
    const call_record_t records[], size_t M, int total_minutes[], size_t C)
```

qui calcule, pour chaque client, le nombre total de minutes d'appel et les stocke dans le tableau total_minutes. Le client i aura le nombre de minutes stocké dans `total_minutes[i]`. N'oubliez pas d'initialiser ses éléments à 0 avant tout.

Dans `main()`, avant d'appeler cette fonction, définissez un tableau de C entiers `total_minutes`, et passez-le en troisième argument pour y recevoir les valeurs :

```
temps_total(records, M, total_minutes, C);
```

Affichez le contenu du tableau. Vous devriez obtenir :

```
Client 0: 2287 minutes
Client 1: 623 minutes
Client 2: 608 minutes
Client 3: 1174 minutes
Client 4: 879 minutes
Client 5: 0 minutes
Client 6: 3337 minutes
Client 7: 292 minutes
Client 8: 244 minutes
Client 9: 4995 minutes
```

5. Filtre

Écrivez une fonction

```
void temps_total_mois(
    const call_record_t records[], size_t M, int mois,
    int total_minutes[], size_t C)
```

qui calcule, pour chaque client, le nombre total de minutes d'appel *pour le mois donné* et les stocke dans le tableau `total_minutes`.

Le mois sera donné comme un entier à 6 chiffres avec les quatre chiffres les plus significatifs qui correspondent à l'année et les deux suivants au mois. Par exemple, janvier 2024 est représenté comme `202401`.

Dans `main()`, avant d'appeler cette fonction, définissez un tableau de C entiers `minutes_janvier` et passez-le en quatrième argument pour recevoir les valeurs.

```
  temps_total_mois(records, M, 202401, total_minutes, C);
```

Affichez le contenu du tableau.

Pour le mois de janvier 2024, vous devriez obtenir :

```
Client 0: 0 minutes
Client 1: 0 minutes
Client 2: 0 minutes
Client 3: 172 minutes
Client 4: 105 minutes
Client 5: 0 minutes
Client 6: 582 minutes
Client 7: 0 minutes
Client 8: 71 minutes
Client 9: 492 minutes
```

Et pour le mois de novembre 2023 :

```
Client 0: 265 minutes
Client 1: 72 minutes
Client 2: 0 minutes
Client 3: 97 minutes
Client 4: 125 minutes
Client 5: 0 minutes
Client 6: 465 minutes
Client 7: 71 minutes
Client 8: 15 minutes
Client 9: 608 minutes
```

6. La facture

Écrivez une fonction

```
int centimes_minute(const cost_t costs[], size_t C, int tel)
```

qui prend un numéro de téléphone et retourne le coût par minute d'appeler ce numéro. S'il est dans le tableau `costs`, alors la fonction retourne la valeur qui y est stockée. Sinon, elle retourne le tarif standard de 20 centimes par minute.

Ensuite, écrivez une fonction

```
void centimes_total_mois(
    const call_record_t records[], const cost_t costs[], size_t M, size_t N,
    int mois, int total_cout[], size_t C)
```

qui calcule les montants des factures des clients pour un mois donné, au centime près, et les stocke au bon emplacement dans le tableau `total_cout`. Le montant de la facture du client `i` doit se trouver dans `total_cout[i]`.

Affichez le contenu du tableau.

Par exemple, pour novembre 2023 :

```
Client 0: CHF 39.00
Client 1: CHF 14.40
Client 2: CHF 0.00
Client 3: CHF 19.40
Client 4: CHF 91.80
Client 5: CHF 0.00
Client 6: CHF 73.00
Client 7: CHF 14.20
Client 8: CHF 3.00
Client 9: CHF 180.00
```

et pour août 2023 :

```
Client 0: CHF 342.60
Client 1: CHF 30.00
Client 2: CHF 0.00
Client 3: CHF 27.40
Client 4: CHF 75.00
Client 5: CHF 0.00
Client 6: CHF 94.80
Client 7: CHF 8.80
Client 8: CHF 1.00
Client 9: CHF 160.40
```

7. Si vous en voulez encore

(La solution à cette section n'est pas fournie.)

Vous avez maintenant deux tableaux distincts qui ont les clients pour indices. Remplacez-les par un seul tableau d'une nouvelle structure `client_t` que vous définirez.

Challenge : pouvez-vous éliminer le lien entre `no_client` et les indices de ce tableau ? Imaginez que les numéros de clients ne soient plus consécutifs à partir de 0, mais aussi de grands nombres comme les numéros de téléphone.