CS-119(a) – ICC-C Série 10

2024-04-30

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

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

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 en premier deux entiers `M` et `N`. 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 entre 0 et 9 (pas beaucoup de clients chez cet opérateur, mais on espère en attirer plus).
- 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 30 avril 2024 s'écrit `20240430`.
- 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 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 tarif par minute en CHF pour appeler ce numéro.

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

Exo1 Lecture

Définissez une `struct call_record` avec le synonyme `call_record_t` pour stocker des informations d'un appel, et une autre `struct tarif` avec le synonyme `tarif_t` pour les informations de facturation.

Définissez une troisième struct pour tout stocker :

```
typedef struct _data
{
    int M, N;
    call_record_t *records;
    tarif_t *tarifs;
} data_t;
```

Définissez une variable `data_t data`. Lisez les données (avec `scanf`) directement dans cette variable. Utilisez la redirection de l'entrée pour lire le fichier texte donné. Utilisez l'allocation dynamique de la mémoire (`malloc`) pour allouer le tableau de `M` éléments `data.records` et le tableau de `N` éléments `data.tarifs`. N'oubliez pas d'appeler `free` à la fin du programme !

Exo2 Durée

Ecrivez une fonction

```
void temps_total(const data_t *pdata, int *total_minutes);
```

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 10 entiers `int total_minutes[10]` et passez le en deuxième argument pour y recevoir les valeurs : `temps_total(&data, total_minutes);`

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
```

Exo3 Filtre

Ecrivez une fonction

```
void temps_total_mois(const data_t *pdata,
                      int mois,
                      int *total_minutes);
```

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 client `i` aura le nombre de minutes stocké dans `total_minutes[i]`. N'oubliez pas d'initialiser ses éléments à 0 avant tout.

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 pour janvier 2024 : `202401`.

Dans `main` avant d'appeler cette fonction définissez un tableau de 10 entiers `int minutes_janvier[10]` et passez le en troisième argument pour recevoir les valeurs : `temps_total(&data, 202401, minutes_janvier);`

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
```

Exo4 La facture

Ecrivez une fonction

```
float cout_minute(const data_t *pdata, int tel);
```

qui prend un numéro de téléphone et retourne le coût par minute d'appeler ce numéro. S'il est dans le tableau `pdata->tarifs`, alors la fonction retourne la valeur qui y est stockée, sinon elle retourne le tarif standard de `0.2` CHF par minute.

Ensuite écrivez une fonction

```
void cout_total_mois(const data_t *pdata,
                     int mois,
                     float *total_cout);
```

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
```