INF3135 : TP1

AZIZ SALAH

Date limite de la remise : le mardi 29-octobre-2013 avant 23h59

1. Objectifs pédagogiques du travail

- Maitriser la manipulation des pointeurs et leur arithmétique ;
- La saisie des données à partir d'un fichier ;
- Découpage fonctionnel et composition ;
- Gestion des erreurs.

2. Description du problème

Étant donné un fichier d'entiers représentant un tableau 2D, nous voulons supprimer certaines lignes et/ou colonnes du tableau et afficher le résultat à l'écran.

3. Format du fichier de données :

Nous supposons que le tableau 2D d'entiers est représenté par un fichier texte. Les lignes du fichiers contiennent un nombre variable d'entiers. Le fichier de données représente un tableau 2D $n \times m$ où n est le nombre de lignes du fichier qui contiennent au moins un entier alors que m est le nombre d'entiers de la ligne qui en contient le plus. Le fichier de données peut comporter des lignes vides d'entiers celles-ci sont ignorées et ne représentent aucune ligne dans le tableau 2D. Toute ligne du fichier qui comporte moins que m entiers est complétée par des zéros pour former une ligne du tableau 2D.

Vous pouvez supposer que les fichiers de données utilisés ne comportent que des entiers séparés par des blancs.

4. Page man de la commande filtre

```
Nom
---
    filtre -- Supprime des lignes et des colonnes dans un fichier de données
          et affiche le résultat

SYNOPSIS
--------
       filtre <fichier> [-C <liste>] [-L <liste>]
       filtre <fichier> [-L <liste>] [-C <liste>]
DESCRIPTION
-----------
```

`<fichier>` est le nom d'un fichier de données qui va représenter le tableau 2D.

Le programme `filtre` a deux options optionnelles qui sont :

`-C <liste>` : `<liste>` indique la liste des colonnes à supprimer.

`-L <liste>` : `<liste>` indique la liste des lignes à supprimer.

Date: A13.

`<liste>` peut être vide ou composée d'un ou plusieurs éléments séparés par un ou plusieurs espaces. Un élément dans `<liste>` représente un domaine. Les lignes ou les colonnes qui sont dans l'un des domaines seront supprimées sachant que :

`<num>` **:** `num` représente le domaine singleton contenant `num` ;

`<num1>-<num2>` **:** représente le domaine des entiers de `<num1>` à `<num2>` inclus ;

`<num>-` **:** représente le domaine des entiers supérieurs ou égals à `<num>`

`-<num>` **:** représente le domaine `0-<num>`.

où `num`, `num1` et `num2` sont des entiers positifs. On suppose aussi que les numérotations des colonnes et des lignes commencent de zéro.

5. Exemple d'exécution

Voici le fichier `test01.txt` comme exemple de fichiers de données à fournir au programme `filtre`

```
16 10 4 -2 3
1

6
2 5 1 10
```

Un ligne du fichier est supposée vide si elle ne comporte aucun entier. Ansi, ce fichier comporte 4 lignes non vides. La ligne «16 10 4 -2 3» comporte 5 entiers et représente la ligne avec le plus d'entiers. Ainsi le tableau 2D que ce fichier représente est le tableau `4 x 5` suivant :

16	10	4	-2	3
1	0	0	0	0
6	0	0	0	0
2	5	1	10	0

Voici une session d'essaie exécutée sur le serveur `chicoree` :

```
chicoree > cat  test01.txt
16 10 4 -2 3
1

2 5 1 10
chicoree > gcc -Wall -std=c99 filtre.c fonctions.c -o filtre
chicoree > ./filtre test01.txt
   16    10     4    -2     3
    1     0     0     0     0
    6     0     0     0     0
    2     5     1    10     0
chicoree > ./filtre test01.txt -C 0-2 1 -L
    -2     3
     0     0
     0     0
    10     0
chicoree > ./filtre test01.txt -C -1-3 -L
```

```
Erreur de syntaxe dans un domaine.
chicoree > ./filtre test01.txt -C 0-20
Tableau vide!
chicoree >
```

6. Exigences non fonctionnelles

6.1. **Gestion de l'allocation dynamique.** Le programme doit s'adapter dynamiquement avec la taille des données en utilisant l'allocation dynamique. Tout espace alloué dynamiquement doit être libéré dès qu'il n'est plus requis.

6.2. **Gestion des erreurs.** Tous les messages d'erreurs doivent être écrits sur le canal d'erreurs `stderr` seulement en utilisant la fonction `signaler_erreur` fournie. Seules les erreurs considérées par la fonction `signaler_erreur` sont à traiter dans votre programme. Le déclenchement de n'importe laquelle de ces erreurs entraine l'arrêt du programme. Le programme suppose que les données du fichier sont valides par contre un fichier peut être vide d'entier : par exemple un fichier de taille 0 ou encore un fichier qui ne comporte que des caractères blancs.

6.3. **Composition du programme.**

- `«fonctions.h»` : est un fichier fourni et définissant les prototypes de fonctions à définir dans le fichier `«fonctions.c»`. Le fichier `«fonctions.h»` ne doit pas être modifié.
- `«fonctions.c»` : une version à compléter de ce fichier est fournie. Les fonctions fournies dans ce fichier ne doivent pas être modifiées. Le fichier `«fonctions.c»` doit contenir minimalement toutes les fonctions de `«fonctions.h»`. Vous pouvez ajouter d'autres fonctions dans le fichiers `«fonctions.c»` mais celle-ci doivent être appelées seulement localement c'est à dire seulement par des fonctions de `«fonctions.c»`.
- `«filtre.c»` : ne comporte que la fonction `main` et possiblement des fonctions qui sont appelées localement seulement.
- Les règles précédentes de cette section assurent que la compilation de votre programme `«filtre.c»` par la commande :
  `gcc -Wall -std=c99 filtre.c fonctions.c -o filtre`
  devrait réussir quelque soit le fichier `«fonctions.c»` définissant seulement les fonctions de `fonctions.h`
- La compilation de votre programme ne devrait donner aucun warnning.
- Tous les fichiers doivent contenir les nom des auteurs et leurs codes permanents en commentaire.
- L'affiche des résultats doit se faire avec la fonction `affiche_Tab2D` fournie.

6.4. **Portabilité du programme.** Afin de s'assurer de la probabilité de votre programme, celui-çi devrait être compilé et testé sur les serveurs chicoree et rayon1 (ou rayon2). Le premier serveur roule sur Linux alors que les deux derniers sont des Sun.

7. Ce que vous devez remettre

Un membre de l'équipe seulement remettra vos fichiers `«fonctions.c»` et `«filtre.c»` électroniquement dans Moodle en suivant le lien approprié. Pas de remise en double svp.

8. Pondération

- Tests de fonctionnement : 70%
- Exigences non fonctionnelles : 20%
- Structure du programme, commentaires, indentation ... : 10%

Votre travail sera corrigé en le soumettant à une batterie de tests. Si jamais aucun test ne marche la note maximale sera 20%

Remarques importantes :

- Un programme ne compilant pas se verra attribuer la note 0.
- Des pénalités sont à prévoir si le programme ne respecte pas l'affichage exigé.
- Aucun programme reçu par courriel ne sera accepté.
- Les règlements sur le plagiat seront strictement appliqués.
- 10% comme pénalité de retard par journée entamé. Après cinq jours, le travail ne sera pas accepté.
- La remise d'un fichier "zippé" donne lieu à 10% de pénalité.