INF3135 : TP2

AZIZ SALAH

Date limite de la remise : le lundi 09-décembre-2013 avant 23h59

1. Objectifs pédagogiques du travail

- Maintenance
- Remodelage (refactoring)

2. Description du mandat

Ce travail vise la maintenance du programme `filtre` réalisé au TP1. Nous voulons maintenant que le programme `filtre` construise son tableau 2D à partir de plusieurs fichiers. Nous voulons aussi que les fichiers traités soit des fichiers textes. Dans le cas d'un seul fichier, une ligne du tableau 2D est construite à partir des mots d'une ligne du fichier. Les mots sont repérés à l'aide des caractères blancs qui les séparent.

3. Page man de la commande filtre

```
Nom
---
    filtre -- Supprime des lignes et des colonnes d'un tableau 2D construit
              à partir de plusieurs fichiers textes et affiche le résultat

SYNOPSIS
--------
       filtre -V <liste_fichiers> [-C <liste>] [-L <liste>]
       filtre -V <liste_fichiers> [-L <liste>] [-C <liste>]
       filtre -H <liste_fichiers> [-C <liste>] [-L <liste>]
       filtre -H <liste_fichiers> [-L <liste>] [-C <liste>]

DESCRIPTION
-----------
```

`<liste_fichiers>` est une liste de fichiers de données où chacun représente un tableau 2D. On suppose que la liste ne peut pas être vide.

`-H` indique que les tableaux 2D issus des fichiers sont fusionnés horizontalement de la gauche vers la droite. Si un tableaux comporte moins de lignes il sera complété avec des lignes composées de chaines de caractères vides avant la fusion.

`-V` indique que les tableaux 2D issus des fichiers sont fusionnés verticalement de la gauche vers la droite. Si un tableaux comporte moins de colonnes il sera complété avec des colonnes de composées de chaines de caractères vides avant la fusion.

Date: A13.

4. Exemple d'exécution

Voici une session d'essaie exécutée sur le serveur `malt` pour illustrer la fusion horizontale :

```
malt > cat  test01.txt
6 101 -2
13 14
malt > cat  test02.txt
abc 1F0 4 -2.TG 3
1 2 3 4 5
6

2 5 1 10

chicoree > ./filtre -H test01.txt test02.txt
[6          101        -2        abc        1F0        4          -2.TG      3          ]
[13         14                   1          2          3          4          5          ]
[                                6                                                      ]
[                                2          5          1          10                    ]
```

La représentation en donnée du résultat en utilisant des chaine de caractère est la suivante :

$$\begin{pmatrix} "6" & "101" & "-2" & "abc" & "1F0" & "4" & "-2.GT" & "3" \\ "13" & "14" & "" & "1" & "2" & "3" & "4" & "5" \\ "" & "" & "" & "6" & "" & "" & "" & "" \\ "" & "" & "" & "2" & "5" & "1" & "10" & "" \end{pmatrix}$$

Ce résultat est la fusion horizontale de

$$\begin{pmatrix} "6" & "101" & "-2" \\ "13" & "14" & "" \end{pmatrix}$$

obtenu du fichier `test01.txt` et

$$\begin{pmatrix} "abc" & "1F0" & "4" & "-2.GT" & "3" \\ "1" & "2" & "3" & "4" & "5" \\ "6" & "" & "" & "" & "" \\ "2" & "5" & "1" & "10" & "" \end{pmatrix}$$

obtenu du fichier `test02.txt`.

5. Exigences non fonctionnelles

5.1. **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.

5.2. **Gestion des erreurs.** Identique au TP1. Si l'un des fichiers de la liste n'existe pas ou ne contient pas de données, le programme quitte en affichant le message d'erreur approprié.

5.3. **Composition du programme.**

- Vous pouvez réutiliser votre code du TP1 en tenant compte des critères de qualité en ce qui concerne la cohésion des fonctions.
- Les fichiers fournis :
  - + `«affichage.c»` contient l'implémentation des seules fonctions d'affichage permises.
  - + `«affichage.h»` contient en plus des prototypes des fonctions d'affichage, le type `struct tab2D` utilisé par l'affichage.
  - + `«exemple.c»` contient un exemple de manipulation des structures de données.

- En plus des fichiers «`affichage.c`» et «`affichage.h`» qui ne doivent pas être modifiés, votre programme doit être décomposé en modules. Au minimum vous devez avoir trois fichier «`.c`»
  + un fichier ou plus pour les fonctions qui ne sont pas spécifiques au problème ;
  + un fichier ou plus pour les fonctions spécifiques au problème ;
  + un fichier qui ne contient que la fonction main : il s'agit de «`filtre.c`».
- Chaque fonction doit être commentée à la javadoc.
- Ne pas oublier de fournir les fichier .h aussi
- La compilation de votre programme doit se faire par la commande make :
  `make filtre`
  L'exécutable doit s'appeler `filtre` et tous les fichiers du programme sont supposés être dans le même répertoire.
- La compilation de votre programme ne devrait donner aucun warnning.
- Tous les fichiers doivent contenir les noms des auteurs et leurs codes permanents en commentaire.

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

6. Ce que vous devez remettre

Un membre de l'équipe seulement remettra les fichiers électroniquement un par un (pas de zip SVP) dans Moodle en suivant le lien approprié. Pas de remise en double SVP.

NB : Moodle permet aussi la suppression de fichiers remis.

7. Pondération

- Tests de fonctionnement : 50%
- Exigences non fonctionnelles : 50%
  - makefile
  - cohesion
  - commentaire
  - décomposition
  - warnnings

Remarques importantes :

- Un programme ne compilant pas se verra attribuer la note 0 au TP2.
- Aucun programme reçu par courriel ne sera accepté. En cas de panne des serveurs, un délai supplémentaire vous sera accordé sans pénalité de retard.
- Les règlements sur le plagiat seront strictement appliqués.
- 10% comme pénalité de retard par journée entamée. Après cinq jours, le travail ne sera pas accepté.
- La remise d'un fichier "zippé" donne lieu à 10% de pénalité.