IFT1169 – TRAVAIL PRATIQUE #2 – 06 novembre 2018

« Un ensemble de conteneurs »

Mohamed Lokbani

Équipes : le travail est à faire en binôme, mais vous ne remettez qu'un travail par équipe.

Remise : une seule remise est à effectuer par voie électronique **le lundi 3 décembre 2018, 23h59 au plus tard**, **sans possibilités de prorogation.**

Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer les STL.

Énoncé : ce travail consiste à lire une série de données éparpillées dans 4 fichiers. Ces données seront sauvegardées dans une collection la plus appropriée. Par la suite, il vous sera demandé de fournir le résultat obtenu pour certaines opérations sur ces collections.

On vous présente le travail à réaliser sous la forme d'étapes pour simplifier l'énoncé. En réalité, certaines étapes peuvent s'entrelacer entre elles. On peut par exemple, en même temps, lire le fichier, mettre ses caractères en minuscule avant de sauvegarder le résultat obtenu dans un conteneur.

Étape 1 : consiste à lire les fichiers et à valider que la lecture s'est faite correctement. Vous allez utiliser la même approche que pour le TP#1.

Étape 2 : vous allez mettre en minuscule tous les caractères du texte. Ceci va permettre d'éviter de distinguer deux mots par la casse. Exemple pour « Avis » et « avis », nous allons considérer que nous avons affaire à un seul mot « avis ».

Étape 3 : insérer tous les éléments dans un conteneur, en prenant soin de calculer au passage la fréquence d'apparition du mot.

Étape 4 : mettre en place les différentes opérations demandées sur les conteneurs, par exemple trouver l'élément le plus fréquent, le moins fréquent, les éléments communs, etc. Ces opérations sont détaillées plus loin dans l'énoncé.

Présentation des données : les données sont réparties dans 4 fichiers. Ces fichiers sont au format UTF-8. Chacun des fichiers représente un texte d'un auteur donné.

Nom du fichier	Texte
arlequin.txt	"Arlequin poli par l'amour" de Marivaux https://fr.wikipedia.org/wiki/Arlequin_poli_par_l%27amour
camelias.txt	"La Dame aux camélias" de Dumas https://fr.wikipedia.org/wiki/La_Dame_aux_cam%C3%A9lias
germinal.txt	"Germinal" de Zola https://fr.wikipedia.org/wiki/Germinal_(roman)
ptprince.txt	"Le petit prince" de Saint-Exupéry https://fr.wikipedia.org/wiki/Le_Petit_Prince

Chacun des textes a été découpé en mots. Chaque ligne contient un seul mot (ou un mot composé).

Nous allons traiter un 5e fichier fictif. Il s'agit de rassembler les informations des 4 fichiers dans un fichier qui n'existe pas. L'opération est faite de manière transparente à la volé.

Argument : le programme prend un entier comme argument. Sa valeur doit-être strictement supérieure à zéro . Cet argument est optionnel. S'il est absent, la valeur par défaut est « 5 ».

Opérations à réaliser :

1- Trouver le mot le plus cité dans un fichier donné. Si nous avons plusieurs mots, on affiche uniquement le premier par ordre alphabétique.

2- Trouver le mot le moins cité dans un fichier donné. Si nous avons plusieurs mots, on affiche uniquement le premier par ordre alphabétique.

3- Trouver les « N » clés associées à la plus petite valeur. La valeur de « N » est celle fournie en argument, sinon elle prend la valeur par défaut, 5. Si cette recherche fournit un nombre de clés, inférieur au nombre demandé, on se contente de fournir le nombre disponible. Exemple, si N vaut 100 et la recherche n'a pu trouver que 10 clés, on se contente d'afficher que les 10 clés disponibles.

4- Trouver les « N » valeurs plus élevées et afficher leurs clés respectives. On cherche donc les M mots les plus cités, or il se peut que plusieurs mots soient cités autant de fois. Dans ce cas, on doit les afficher tous. Le nombre M peut-être différent du nombre N, car la recherche se fait sur les N meilleurs valeurs. Une valeur peut avoir plusieurs mots. On vous encourage à examiner attentivement les fichiers fournis pour bien comprendre ce cas de figure.

5- Trouver les éléments différents entre deux fichiers. La recherche est donc croisée. La recherche entre les fichiers 1 et 2, consiste à chercher si un élément du fichier 2 se trouve dans 1. La recherche entre les fichiers 2 et 1, consiste à chercher si un élément du fichier 1 se trouve dans 2. Pour cette raison que cette recherche est croisée. Chaque recherche est affichée à part. Là aussi, vous pouvez examiner les fichiers fournis.

6- Trouver les éléments communs entre deux fichiers. À cause de l'affichage imposé en sortie, la recherche est croisée. La recherche entre les fichiers 1 et 2, consiste à chercher si un élément du fichier 2 se trouve dans 1. La recherche entre les fichiers 2 et 1, consiste à chercher si un élément du fichier 1 se trouve dans 2. Pour cette raison que cette recherche est croisée. Chaque recherche est affichée à part. Là aussi, vous pouvez examiner les fichiers fournis. Pourquoi la recherche est-elle croisée? On vous demande d'afficher en sortie le mot et le nombre de fois où il a été cité. Le mot « adieu » a été cité « 6 » fois dans le fichier « arlequin.txt » alors qu'il est cité « 16 » dans le fichier « camelias.txt ». Le fichier « arlcaminter.txt » va contenir donc « adieu 6 » alors que le fichier « camarlinter.txt » va contenir « adieu 16 ».

Chronomètre :

Nous avons ajouté le code nécessaire pour chronométrer le temps nécessaire pour réaliser tout le travail demandé dans ce TP#2. Pour activer le chronomètre, il suffit d'ajouter cette option à la ligne de commande au moment de la compilation :

-D TOP_CHRONO

Organisation des fichiers : nous avons fourni le fichier « tp2.cpp ». Ce fichier ne doit pas être modifié en aucun cas. Nous avons ajouté au début du programme le fichier d'en-tête « tp2.h » pour permettre d'inclure vos différents fichiers. Vos programmes doivent entre écrits dans d'autres fichiers C++.

Fichiers disponibles :

- Les 4 fichiers de départs « arlequin.txt », « camelias.txt », « germinal.txt » et « ptprince.txt ».

- Les 3 fichiers résultats obtenus pour: N = 5 (valeur par défaut), N=20 et N=100, associés aux opérations 1 à 4.

- Les fichiers des résultats croisés ayant comme suffixe « diff » pour l'opération 5 et « inter » pour l'opération 6.

Notions permises : toutes les notions du C++ sont autorisées.

Contraintes et vérification du résultat en sortie : nous allons comparer de manière automatique (en utilisant des scriptes) la sortie obtenue par votre programme et la nôtre. De ce fait: vous devez suivre les spécifications à la lettre. Vous devez obtenir exactement les mêmes sorties que nous. Pour ce travail, le format d'affichage est très basic. Nous n'avons pas imposé un format fixe. Vous devez juste faire les opérations dans l'ordre et afficher les résultats simplement avec les séparateurs demandés.

Voir le TP#1 pour les commandes utilisées afin de comparer deux fichiers.

Fichiers à remettre : vous devez remettre le fichier compressé « tp2A18.zip » par l'entremise du système de remise de Studium. Par ailleurs, vous devez inclure dans votre remise un rapport décrivant le travail effectué.

Remise : vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI.

Barème : ce TP2 est noté sur **16 points**. Les points sont répartis comme suit :

Conception du programme : 2 points
Organisation du programme sous la forme de classes. Vous évitez ainsi de coder toutes les fonctionnalités dans la fonction « main »!!!! Utilisation de la STL et ses algorithmes.

Opérations : 10 points

1- 1.5 point
2. 1.5 point
3. 1.5 point
4. 1.5 point
5. 2 points
6. 2 points

Avis global : 2 points
Programmation, clarté du code, etc.

Rapport : 2 points
On ne vous demande pas de nous dire à quoi sert la variable « i », la boucle « for » ou la boucle « while ». Vos commentaires dans le code devraient nous expliquer leurs utilités. On vous demande de vous mettre à la place d'une personne qui connaît l'objectif de votre programme sans connaître votre code. En lisant votre rapport, elle devra pouvoir naviguer avec aisance dans votre programme. Elle saura par exemple comment la programmation

orientée objet a été introduite et pour quelle raison; l'utilité des différentes fonctionnalités, etc.

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ....
- La non-remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Un programme qui ne compile pas : 0.
- Un programme qui compile, mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Documents fournis
- Les résultats obtenus
- Feuille de route.

Communications
- Par courriel : une seule adresse « dift1169@iro.umontreal.ca ».

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre courriel, au moins la chaîne [IFT1169] et une référence au tp01.

- À travers Studium : forum TP#2.

- En personne : à la démo ou sur rendez-vous.

Mise à jour

06-11-2018 diffusion de l'énoncé.