IFT1169 – TRAVAIL PRATIQUE #3 – 13 novembre 2020

« 2 pour 1 »
Mohamed Lokbani

Équipes : le travail est à faire en monôme (une seule personne).

Remise : une seule remise est à effectuer par voie électronique **le lundi 14 décembre 2020, 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 l'utilisation et la gestion d'une collection non ordonnée du type « map » ainsi que les interfaces graphiques.

Énoncé : ce travail est subdivisé sur plusieurs étapes vous permettant de construire graduellement le programme demandé. Vous allez définir une collection qui va emmagasiner des informations relatives aux utilisateurs d'un système informatique quelconque. Par la suite, vous allez manipuler les caractéristiques de cette collection pour mesurer ses performances. Finalement, vous allez interagir avec cette collection soit à l'aide d'une ligne de commandes, soit à l'aide d'une interface graphique.

Question -1- : écrire un programme qui va lire le contenu du fichier « mdp.dat » et l'afficher en sortie.

- On suppose que le nom du fichier est fixé. Vous devez vérifier qu'il est accessible.
- Le format d'affichage en sortie doit respecter ce qui suit :
  - L'alignement est à gauche.
  - La taille en caractères des différents champs est comme suit :

Nom d'utilisateur (login)	Mot de passe	Nom	Prénom
10	15	10	10

- Vous devez obtenir exactement le même résultat que le fichier « sortie.dat » fourni avec cet énoncé.

Question -2- : vous reprenez le précédent programme pour lui ajouter une collection responsable de contenir les données lues à partir du fichier « mdp.dat ». Cette collection doit-être du type « unordered_map ».

- La taille de la collection ne sera pas fixée pour le moment.
- La clé est du type « string ». Elle représente le « login » de l'utilisateur.
- La valeur est du type un ensemble (classe ou structure) regroupant les éléments (mot de passe, nom, prénom).
- Vous devez modifier l'affichage en sortie comme suit :
  - Affichez d'abord le nombre d'éléments dans la collection.
  - Affichez le nombre de chaînages (buckets) utilisé par votre collection.
  - Affichez le contenu de la collection en respectant le nombre de caractères imposé pour chaque champ :

Login	Numéro du chaînage (bucket)	Mot de passe	Nom	Prénom
10	4	15	10	10

Question -3- : jusqu'à présent, nous avons utilisé la fonction de hachage par défaut. Vous reprenez le précédent programme pour lui ajouter une nouvelle fonction de hachage. Cette fonction va utiliser la technique suivante :

- Commencez par convertir chaque caractère de la clé en un nombre entier en utilisant le code « ASCII ».
- Par la suite, calculez la somme totale obtenue pour une clé donnée.
- Finalement, calculez-le modulo 16 de cette somme.
- La valeur obtenue représente votre clé de hachage.
- Exemple : login = garfield
  - somme = g (103) + a (97) + r (114) + f (102) + i (105) + e (101) + l (108) + d (100) = 830
  - somme_module = 830%16 = 14
- Affichez le contenu du conteneur de la même manière que dans la précédente question.

<u>Question -4- :</u> vos réponses détaillées doivent figurer dans le rapport associé à ce travail.

- Dans la question -3-, fallait-il définir aussi une fonction de comparaison en cas d'égalité de clés? Pourquoi?
- Dans la question -3-, le fait de fixer la taille du conteneur à 16, par défaut, elle va avoir un effet sur le résultat obtenu?
- Dans la question -3-, le fait de fixer la taille du conteneur à 64, par défaut, elle va avoir un effet sur le résultat obtenu?
- Éditez le fichier « mdp.dat » pour y ajouter le login « espace » avec des données fictives. Est-ce que le login « espace » est dans la même « bucket » que le login « jedi »? Pourquoi?
- Prenez le même fichier « mdp.dat » avec l'entrée « espace » et testez-le sur la question -2-. Est-ce que le login « espace » est dans la même « bucket » que le login « jedi »? Pourquoi?
- Est-ce que les deux précédents résultats demeurent vrais, peu importe la taille du conteneur?

<u>Question -5- :</u> vous reprenez le précédent programme pour lui ajouter une interaction avec l'utilisateur.

- Les 4 cas de figure qui peuvent se présenter à vous :

  - Nom d'utilisateur est incorrect

    Login : garfild
    Password : Lasagne
    L'authentification a échoué.
  - Mot de passe est incorrect

    Login : garfield
    Password : Laagne
    L'authentification a échoué.
  - Nom d'utilisateur et mot de passe sont incorrects

    Login : garfild
    Password : Laagne
    L'authentification a échoué.
  - Nom d'utilisateur et mot de passe sont corrects

    Login : garfield
    Password : Lasagne
    L'authentification a réussi. Bienvenue à Jim Davis

Pour des raisons de sécurité, quand il y a une erreur dans l'un des deux champs, on ne précise pas lequel des deux pose un problème.

<u>Question -6- :</u> l'interaction à travers la ligne de commandes est remplacée par une interaction graphique.
<u>Fenêtre d'authentification</u> : elle contient les deux champs, code d'accès et mot de passe. Le mot de passe ne doit pas être lisible à l'écran. Il doit être masqué.
<u>En cas d'échec :</u> reprendre le scénario d'un cas d'échec de la question 5. Après 5 tentatives, vous bloquez les accès pour 5 minutes. C'est probablement un hacker! Afficher « Trop de tentatives, essayer plus tard ».
<u>En cas de réussite :</u> reprendre le scénario d'un cas de réussite de la question 5. Afficher par la suite, une fenêtre

avec la date et l’heure de la connexion, en plus d’un bouton « OK ». En cliquant sur ce bouton, l’utilisateur ferme en quelque sorte la fenêtre et le programme prend fin.

<u>Fichiers à remettre :</u>

« **TP3_A20.workspace** » : tout le travail doit-être réalisé dans codelite dans l’espace de travail (« workspace ») « TP3_A20 ». Chaque étape de ce travail pratique est matérialisée dans un projet. Ainsi l’espace de travail va contenir au final 5 projets. Pour rappel, les 5 projets sont en rapport avec les questions 1, 2, 3, 5 et 6.
« **TP3_A20 _Rapport.pdf** » : le rapport doit décrire le travail effectué ainsi que les réponses aux questions posées à « Question 4 ».
« **TP3_A20 _feuille2route.pdf** » : une auto-évaluation du travail réalisé.

Vous devez regrouper l’ensemble de ces fichiers dans « TP3_A20.zip » et le remettre par l’entremise du système Studium, dans le dépôt « tp03 ».

La correction va se faire sur un poste de la DESI en utilisant le compilateur « g++ » (9.2). Vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI.

<u>Barème :</u> ce TP3 est noté sur **15 points**. Les points sont répartis comme suit :

Question -1- : lecture entrée/sortie (**1 point**), attention au format de l'affichage.
Question -2- : collection et structure de données (**2 points**).
Question -3- : fonction de hachage (**2 points**).
Question -4- : diverses questions (**3 points**).
Question -5- : interaction avec l'utilisateur sur la ligne de commandes (**2 points**).
Question -6- : interaction avec l'utilisateur à travers une interface graphique (**4 points**).
Rapport : **1 point (autre que les réponses à la question -4-)**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ....
- un makefile manquant dans l'une des étapes (-0.5).
- 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!

Des questions à propos de ce TP? 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 tp03.

<u>Mise à jour</u>
13-11-2020 diffusion de l’énoncé.

Annexe

- Description des méthodes de la classe « unordered_map » :

http://fr.cppreference.com/w/cpp/container/unordered_map

- Description des fichiers disponibles sur le site web du TP#3.

Nature	Nom
Affichage obtenu à la question -1-	sortie_q1.txt
Affichage obtenu à la question -2-	sortie_q2.txt
Affichage obtenu à la question -3-	sortie_q3.txt
Entrée fournie pour la question -5-	entree_q5.txt
Affichage obtenu à la question -5-	sortie_q5.txt
Fichier de données	mdp.dat

Redirections

On peut rediriger les sorties vers un fichier quelconque, par exemple pour la 1re question:

tp3_q1.exe > sortie_q1.txt

On peut éviter aussi de fournir manuellement les données pour la 5e question comme suit:

tp3_5.exe < entree_q5.txt > sortie_q5.txt

Comparaison des sorties

Vous pouvez utiliser la commande « diff » pour comparer les deux sorties, celle que vous obtenez avec celle fournie dans l'énoncé, comme suit:

Linux :

diff –b –w –i –B votre_sortie_q1.txt sortie_q1.txt

Windows :

Utilisez la commande « diff » dans « MinGw », sinon la commande « fc ».

Si la différence est provoquée par des caractères non imprimables (« bizarroïdes »), il faudra vous assurer que le fichier texte est dans le bon format d'encodage. Pour cela, utilisez par exemple :

« Conversion dos à unix » : dos2unix < entree > sortie
« Conversion unix à dos » : unix2dos < entree > sortie