IFT1169 – TRAVAIL PRATIQUE #2 – 14 octobre 2025

« Générateur de mots de passe, 2e partie »

Mohamed Lokbani

Mise à jour
14-10-2025 diffusion de l'énoncé.

Équipes : le travail est à faire en **monôme ou en binôme**. Vous ne devez pas être plus de deux par équipe et ne remettre qu'un seul travail.

Remise : une seule remise est à effectuer par voie électronique au plus tard le **dimanche 9 novembre 2025 à 23h59**. Aucune prorogation ne sera accordée.

Conseils : n'attendez pas le dernier jour pour commencer votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer la **gestion des fichiers (lecture/écriture)**, la **surcharge des opérateurs**, la notion de **fonctions/classes amies (friend)** et l'utilisation des **fonctions génériques (templates)** en programmation orientée objet avec C++. Il s'appuie et étend les fonctionnalités développées dans le TP1.

Énoncé : dans le prolongement du travail réalisé pour le générateur de mots de passe du TP1, vous devez maintenant développer une application capable de **gérer la création et la sauvegarde de multiples comptes d'utilisateurs**. Un compte est représenté par un login et différente version de mots de passe.

1. Modification de l'Interface graphique (Extension du TP1) : vous allez enrichir l'interface graphique existante pour permettre ces nouvelles fonctionnalités :

Génération multiple de comptes : l'utilisateur doit pouvoir spécifier le nombre ***N*** de comptes à générer. La valeur ***N*** doit être un entier positif.

Préfixe de Login : l'utilisateur doit saisir un **préfixe** (une chaîne de caractères) qui sera utilisé pour tous les logins générés.

Nom du fichier de sauvegarde : l'utilisateur doit pouvoir spécifier le **chemin complet et le nom du fichier** où les informations de compte (logins et mots de passe) seront sauvegardées.

Test de Login spécifique : une section de l'interface doit permettre à l'utilisateur de **saisir un login** et de **lancer une recherche** dans le fichier de sauvegarde pour afficher le mot de passe associé (si le compte existe).

2. Génération de Logins : le système doit générer ***N*** logins uniques en combinant le préfixe fourni par l'utilisateur avec un numéro séquentiel.

Format du Login : le format sera « [préfixe]-[séquence] ».

Séquence numérique : la partie séquentielle doit être formatée pour avoir un **nombre fixe de chiffres** égal à la taille maximale du nombre ***N***.

Exemples
- Si ***N = 10*** (taille max 2 chiffres), les séquences doivent aller de « 01 » à « 10 ». Les logins dans ce cas seront « [préfixe]-01 », « [préfixe]-02 » ..., « [préfixe]-10 ».
- Si ***N = 100*** (taille max 3 chiffres), les séquences doivent aller de « 001 » à « 100 ». Les logins seront « [préfixe]-001 », « [préfixe]-002 » ..., « [préfixe]-100 ».

3. Gestion de la Sauvegarde (Lecture/Écriture Fichier) : après la génération des ***N*** pairs (Login/Mot de passe), l'application doit les **sauvegarder** dans le fichier spécifié par l'utilisateur.

Format de fichier : chaque paire Login/Mot de passe doit être enregistrée sur une ligne distincte où chacun des éléments est séparé par le délimiteur tabulation.

Exemple de ligne dans le fichier :
iro-01 mot_de_passe_aleatoire_1

L'application doit gérer les **erreurs de fichier** (chemin invalide, permissions, etc.) et en informer l'utilisateur via l'interface.

4. Implémentation POO Avancée : vous devez utiliser les concepts suivants pour structurer votre code.

- Une classe « Compte » pour encapsuler le login et le mot de passe, ainsi que les opérateurs nécessaires pour l'écriture facile d'un objet « Compte » dans un flux de sortie, la lecture simplifiée d'une ligne du fichier et la construction d'un objet « Compte » à partir d'un flux d'entrée et pour vérifier si le « login » de deux objets « Compte » est identique.

- Une fonction générique pour chercher si un login existe ou pas. Cette fonction va ouvrir le fichier, pour le parcourir ligne par ligne afin d'extraire le login et le comparer avec celui recherché. Elle doit retourner un booléen indiquant si le compte a été trouvé ou non.

Fichier projet : si vous réalisez le travail avec **Codelite**, vous devrez remettre l'espace de travail, après l'avoir préalablement nettoyé. Si vous travaillez « à l'ancienne », vous devrez inclure le fichier « **makefile** » qui permet de recompiler votre projet.

Hypothèses et contraintes

- Pas de STL (Conteneurs) : l'utilisation des conteneurs de la STL est **interdite**. Vous devez gérer le stockage des informations dans le fichier.
- Vous devez utiliser le standard « **c++20** » comme référence.
- Faites attention au format d'affichage des lettres accentuées.
- Nous validerons votre travail sur une machine de la **DESI**. Assurez-vous d'avoir effectué toutes les validations nécessaires sur une telle machine avant de le remettre.

Fichiers à remettre : vous devez remettre le fichier compressé « **tp2A25.zip** » via le système de remise de « **Studium** ». Incluez également un rapport décrivant le travail effectué.

Barème

Ce TP est noté sur 13 **points**, répartis comme suit :

- **Organisation du code :** 3 points
- **Fonctionnalités :** 3 points
- **Interface graphique :** 3 points
- **Rapport :** 2 points
- **Avis global :** 2 points (programmation, clarté du code, etc.)

Le plagiat est strictement interdit et sanctionné par le Règlement disciplinaire sur la fraude et le plagiat concernant les étudiants. Pour plus d'informations, consultez : www.integrite.umontreal.ca

Pénalités

En plus des points indiqués ci-dessus, vous risquez d'en perdre dans les cas suivants :

- **Non-remise électronique :** 0 (volontaire ou par erreur)
- **Programme qui ne compile pas :** 0
- **Programme qui compile, mais ne respecte pas les spécifications :** 0
- **Avertissements (warnings) non corrigés :** la pénalité dépend de la quantité, à partir de -0.25 point.
- **Aberrations de codage :** même si « tous les chemins mènent à Rome », faites l'effort nécessaire pour éviter de prendre le plus long.

Documents fournis

- Feuille de route « A25Tp2_feuille_de_route.odt »

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 tp02.

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