IFT1169 – TRAVAIL PRATIQUE #1 – 16 septembre 2025

« Générateur de mots de passe »
Mohamed Lokbani

Mise à jour
16-09-2025 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 12 octobre 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 les **interfaces graphiques** et la **programmation orientée objet**.

Énoncé : de nos jours, de nombreuses applications sont protégées par un mot de passe. Cette protection est une mesure de sécurité essentielle pour restreindre l'accès à vos données et éviter qu'elles soient utilisées à des fins malveillantes. Le choix d'un mot de passe efficace repose sur plusieurs critères, tels que sa **longueur, sa complexité et son unicité**.

Cependant, la plupart des gens créent des mots de passe faciles à mémoriser en utilisant leur date de naissance, des noms de proches, des nombres simples comme « 12345 » ou des caractères consécutifs comme « qwerty ». Ces habitudes prévisibles sont facilement exploitables par les pirates informatiques à l'aide de programmes automatisés. Une façon de pallier ce problème est de **générer un mot de passe aléatoire**, sans logique ni lien personnel. Il devient alors pratiquement impossible à deviner.

Votre travail consiste à réaliser une interface graphique qui aura pour tâche de générer un tel mot de passe aléatoire.

Description : pour ce travail pratique, nous fournissons le fichier « **genere.cpp** ». Ce fichier contient les bases pour l'implémentation de la génération de nombres aléatoires dans l'intervalle (1,6). Nous avons enrichi le code avec des explications détaillées pour vous aider à en comprendre le fonctionnement. Vous devez **modifier et étendre ce code** pour qu'il puisse générer des mots de passe complexes et s'intégrer à votre interface graphique. Le fichier fourni n'est qu'un point de départ; l'objectif est de le transformer pour qu'il réponde à toutes les spécifications de cet énoncé. Il est donc normal que la plus grande partie de votre travail consiste à adapter, modifier et compléter ce code de base.

Catégories disponibles

Un mot de passe est composé de majuscules, de minuscules, de chiffres et de caractères spéciaux. L'utilisateur a le choix de combiner ces catégories pour définir la composition du mot de passe. En revanche, vous devez vous assurer qu' **au moins 3 des 4 catégories** sont présentes lors de la génération.

- **Caractères spéciaux :** `!@#$%^&*()-_=+[]{}|;:,.<>?`
- **Chiffres :** `0` à `9`
- **Majuscules :** `A` à `Z`
- **Minuscules :** `a` à `z`

À noter que la longueur du mot de passe est fixée par l'utilisateur à travers l'interface graphique.

Interface graphique

Quand un mot de passe est généré, l'interface doit proposer à l'utilisateur de valider ce choix. Si le choix est validé, elle doit lui proposer un test de saisie : il devra **ressaisir le mot de passe qui vient d'être généré** pour prouver qu'il l'a bien lu. Cela permet d'éviter les erreurs de lecture entre des caractères similaires comme le 'O' majuscule, le 'o' minuscule et le chiffre

‘0’. Si après 3 tentatives infructueuses de sa part, un nouveau mot de passe lui est proposé, le cycle de vérification recommence. Si le test est réussi, on le félicite et on lui propose de le garder dans un endroit sécurisé.

À noter : l'utilisateur doit avoir la possibilité de quitter le programme à tout moment, sans être bloqué dans le processus de validation. S'il quitte son application à ce stade, vous devez l'informer qu'il n'a pas finalisé le processus d'obtention du mot de passe.

L'interface doit également contenir des éléments classiques, tels qu'une **aide** décrivant son utilisation, les noms des **auteurs**, la **version** et l'**année** de réalisation.

Nous vous invitons à laisser place à votre imagination et votre créativité!

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

- 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 « **tp1A25.zip** » via le système de remise de « **Studium** ». Incluez également un rapport décrivant le travail effectué.

Barème

Ce TP est noté sur **12 points**, répartis comme suit :

- **Organisation du code :** 2 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

Exigences supplémentaires

Pour le rapport, nous ne vous demandons pas d'expliquer l'utilité de chaque variable « i » ou de chaque boucle « for » ou « while ». Vos commentaires dans le code doivent être plus pertinents et expliquer leur utilité.

Votre rapport doit être clair. Mettez-vous à la place d'une personne qui connaît l'objectif de votre programme, mais pas 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, ainsi que l'utilité des différentes fonctionnalités et leur implémentation, les défis rencontrés et la manière avec laquelle ils ont été résolus, la description des différents fichiers remis et leurs organisations, etc.

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

- Le fichier « tp01A25.cpp »
- Feuille de route « A25Tp1_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 tp01.

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