Hiver 2018

GIF-1003

Thierry EUDE, Ph.D

Travail individuel
à rendre avant
jeudi 15 février 2018 14h00
(Voir modalités de remise à la fin de l'énoncé)

Tout travail remis constitue une contribution originale et distincte de travaux remis par d'autres. Le plagiat est strictement défendu. Tout travail plagié obtiendra la note 0. De plus, les étudiants ou étudiantes ayant collaboré au plagiat seront soumis aux sanctions normalement prévues à cet effet par les règlements de l'Université.

Par ailleurs, vu l'importance des communications écrites dans le domaine de la programmation, il sera tenu compte autant de la présentation que de la qualité du français et ce, dans une limite de 10% des points accordés.

Travail Pratique #1
Investigations, fonctions

UNIVERSITÉ LAVAL
Faculté des sciences et de génie
Département d'informatique
et de génie logiciel

Ce travail a pour but de vous familiariser avec votre environnement de programmation en langage C++. Plus précisément, les objectifs sont:

- l'investigation pour la correction d'un programme comportant des erreurs
- l'approfondissement de la programmation procédurale;
- l'approfondissement de la programmation modulaire;
- la manipulation de chaînes de caractères;
- la manipulation de fichier texte;

Première partie : investigation

Travail à réaliser

Il s'agit d'investiguer afin de corriger un programme. Vous devez alors décrire et illustrer dans un rapport toute la démarche que vous aurez adoptée pour atteindre cet objectif. Il s'agit d'être complet mais concis. Pour cela vous devez utiliser le gabarit fourni.
Votre rapport doit comporter les étapes suivantes :

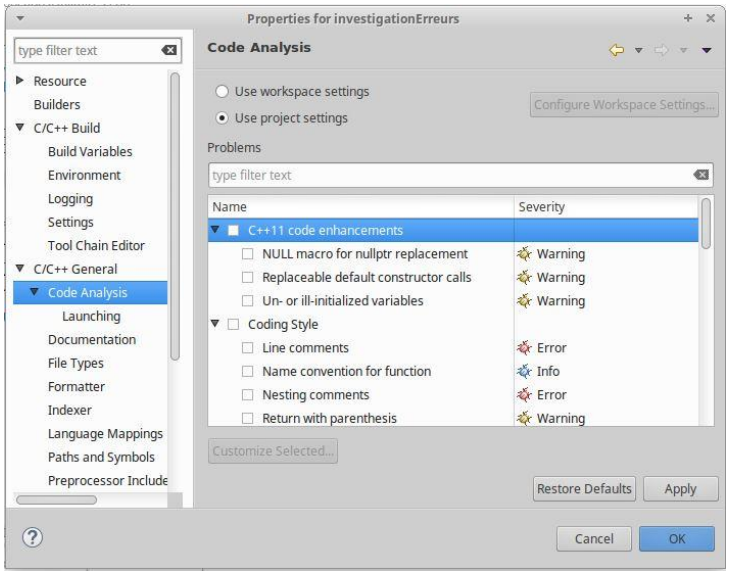
1. Cahier des charges : Quelle est, à la lueur d'une première observation du code, le problème qu'est sensé résoudre le programme.
2. Stratégie : comment comptez-vous vous y prendre pour corriger le programme? (étapes)
3. Localisation des erreurs (étape itérative) : Pour chaque erreur identifiée, **justifier comment vous l'avez localisée** (copies d'écran, commentaires d'explication des messages d'erreur,…). Donc chaque erreur identifiée devrait être accompagnée d'un moyen qui permettrait de la localiser, une localisation « intuitive » n'est pas suffisante. Vous devez démontrer que vous maitrisez l'utilisation d'outils et techniques pour localiser, et corriger des erreurs dans un programme. Précisez également s'il s'agit d'une erreur de syntaxe, d'une mise en garde à corriger, d'une mauvaise pratique, d'une erreur d'édition de lien ou d'une erreur de logique (erreur à l'exécution).
4. Solution (étape itérative liée à une itération de l'étape 3) : apportez la correction en illustrant son efficacité (exemple: copie d'écran démontrant que l'erreur est corrigée).

Si toutes les erreurs de compilation sont corrigées vous devriez avoir comme résultat de compilation le résultat suivant :

```
14:43:13 **** Build of configuration Debug for project solution ****
make all
Building file: ../fonctionsUtilitaires.cpp
Invoking: GCC C++ Compiler
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"fonctionsUtilitaires.d" -MT"fonctionsUtilitaires.d"
-o "fonctionsUtilitaires.o" "../fonctionsUtilitaires.cpp"
Finished building: ../fonctionsUtilitaires.cpp

Building file: ../programmePrincipal.cpp
Invoking: GCC C++ Compiler
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"programmePrincipal.d" -MT"programmePrincipal.d" -o
"programmePrincipal.o" "../programmePrincipal.cpp"
Finished building: ../programmePrincipal.cpp

Building target: solution.exe
Invoking: GCC C++ Linker
g++  -o "solution.exe"  ./fonctionsUtilitaires.o ./programmePrincipal.o
Finished building target: solution.exe


14:43:14 Build Finished (took 872ms)
```

Il vous restera ensuite de vous assurer qu'il n'y a pas d'erreur d'exécution.
Attention! Vous devez configurer Eclipse pour ne pas avoir de mise en gardes relatives à C++11 et plus.
Dans les propriétés du projet, modifier la configuration par défaut dans C/C++ General ->Code Analysis

Désactiver C++11 code enhancements



Deuxième partie : fonctions

Travail à réaliser

Il s'agit de constituer une librairie de fonctions utilitaires qui pourront être réutilisées dans le futur projet de session (TP2, 3 et 4).
Dans les fichiers `validationFormat.h` et `validationFormat.cpp`, vous devrez implanter les fonctions de validation dont les spécifications sont les suivantes.

`bool validerFormatNom (const std::string& p_nom)`
Cette fonction valide le format d'un nom. Un nom ne doit être composé que de lettres mais les espaces sont permis.

`bool validerTelephone(const std::string& p_telephone)`
Cette fonction valide le format des numéros de téléphone canadien au format national. Pour l'analyse de la validité voir : http://fr.wikipedia.org/wiki/Numéro_de_téléphone

`bool validerFormatFichier(std::istream& p_is)`
Cette fonction valide le format du fichier de données texte lu par l'intermédiaire d'un flux d'entrée passé en paramètre.
Le fichier contient le nom (non vide) d'un client sur la première ligne, suivi de son prénom (non vide), sa date de naissance sous la forme de nombres entiers et sous le format jj mm aaaa (chaque nombre est différent de 0), son numéro de téléphone (pensez à utiliser la fonction développée préalablement) et son numéro de folio bancaire (rien à valider ici).

Suivent ensuite des informations sur ses différents comptes bancaires qu'il possède à ce folio qui peuvent être des comptes chèque ou d'épargne (sans apriori d'ordre). C'est la première information qui détermine s'il s'agit d'un compte chèque ou d'un compte d'épargne. Si c'est un compte chèque les données commencent par « cheque », si c'est un compte d'épargne les données commencent par « epargne ».

Exemple « générique » :

```
nom
prenom
jj mm aaaa (date de naissance)
téléphone
numero de folio
type de compte
(si compte cheque)
numéro de compte
taux d'intérêt
solde
description
nombre de transactions
taux d'intérêt minimum
(si compte epargne)
numéro de compte
taux d'intérêt
solde
description
jj mm aaaa (date d'échéance)
…
```

Il est possible d'avoir plusieurs comptes de chaque type.

Donc on aura pour la validation du format de fichier:
nom valide,
prenom valide
date : on supposera que la ligne contient 3 entiers; on doit donc seulement vérifier que ces entiers ne sont pas nuls
numéro de téléphone valide
numéro de folio (rien à valider)
type de compte (deux valeurs possibles seulement: epargne ou cheque)
si cheque
on considère que les données lues correspondent aux types de valeurs attendues (il n'y a pas de validation à faire)
après les 6 lignes on doit trouver soit un autre type de compte, soit c'est la fin du fichier.
si epargne
on considère que les données lues correspondent aux types de valeurs attendues (il n'y a pas de validation à faire)
après les 5 lignes on doit trouver soit un autre type de compte, soit c'est la fin du fichier.

Un exemple de fichier de données vous est fourni. Il est recommandé de constituer vos propres fichiers de données pour faire vos tests.

Important.

Pensez à développer un programme de test vous permettant de vérifier la conformité au cahier des charges de vos fonctions. Ce programme n'est pas à remettre.

Erreurs de compilation

Si un travail comporte ne serait-ce qu'une erreur de compilation, il sera fortement pénalisé, et peut même se voir attribuer la note zéro systématiquement.

Erreurs à l'exécution

Testez intensivement vos programmes. Rappelez-vous qu'un programme qui ne plante pas n'est pas nécessairement sans bogue, mais qu'un programme qui plante même une seule fois comporte nécessairement un bogue. D'ailleurs, si un programme ne plante nulle part sauf sur l'ordinateur de votre ami, c'est qu'**il comporte un bogue**. Il risque donc de planter un jour ou l'autre sur votre ordinateur et, encore pire, sur celui des correcteurs.
Si vous ne testez pas suffisamment votre travail, il risque de provoquer des erreurs à l'exécution lors de la correction. La moindre erreur d'exécution rend la correction extrêmement difficile et vous en serez donc fortement pénalisés.

Critères d'évaluation

1) Respect des biens livrables
2) lisibilité et pertinence du rapport d'investigation
3) modifications du code à corriger
4) explications et commentaires
5) moyens utilisés pour localiser les erreurs puis pour démontrer que les corrections apportées sont bien effectives
6) portabilité de votre code source développé
7) Structure, organisation du code développé, lisibilité
8) Exactitude du code développé (fonctionnalité)

Particularités du barème

- *Si des pénalités sont appliquées, elles le seront sur l'ensemble des points.*
- *Si un travail comporte ne serait-ce qu'une erreur de compilation, il sera fortement pénalisé, et peut même se voir attribuer la note zéro systématiquement.*
- *Il est très important que votre travail respecte strictement les consignes indiquées dans l'énoncé, en particulier les prototypes des fonctions, sous peine de fortes pénalités*

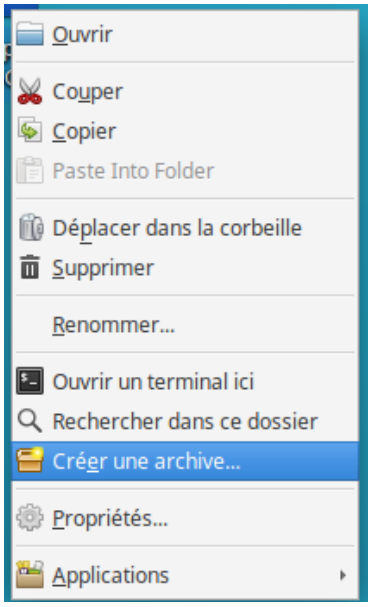
Le rapport d'investigation sera utilisé pour évaluer qualité 3, investigation du BCAPG, tel que précisé dans les objectifs spécifiques du plan de cours.

Bien livrable :

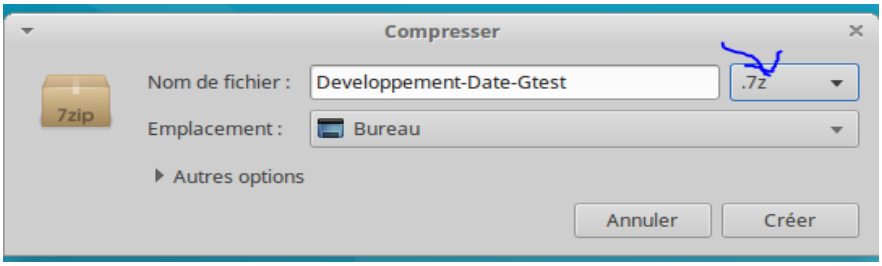
Vous devez rendre six fichiers `rapportInvestigation` (format doc ou pdf, au choix), `fonctionsUtilitaires.h`, `fonctionsUtilitaires.cpp` et `programmePrincipal.cpp` corrigés, `validationFormat.h` et `validationFormat.cpp` mis dans une archive **.7z**.

⚠ Très important ⚠

Utilisez l'outil natif de la machine virtuelle Linux du cours:
Clic droit sur le répertoire



choisir créer une archive, sélectionner **.7z** (premier de la liste)



Le premier travail pour le cours GIF-1003 est un **travail individuel.**
Vous devez remettre votre travail en utilisant le dépôt de l'ENA.

Ce travail est intitulé TP1. **Aucune remise par courriel n'est acceptée**.
Vous pouvez remettre autant de versions que vous le désirez. Veuillez numéroter vos versions pour que l'on puisse savoir quelle sera la dernière. **Il est de votre responsabilité de vous assurer de ce que vous avez déposé sur le serveur**.
N'attendez donc pas le dernier moment pour essayer… vérifier le bon fonctionnement dès que possible.

Date de remise

Le **jeudi 15 février 2018 14h** (par intranet uniquement, voir ci-dessus). Pour tout retard non motivé (voir plan de cours; motifs acceptables pour s'absenter à un examen), la note 0 sera attribuée.

Bon travail