Automne 2021

GIF-1003

Thierry EUDE, Ph.D

Travail individuel
à rendre avant le
jeudi 30 septembre 2021 à 14h00
(Voir modalités de remise à la fin de l'énoncé)

Toute personne prise à plagier, à tricher, activement ou passivement, ou à contrevenir aux directives données dans le cadre d'un examen ou d'un travail noté et contributoire à la note finale du cours, peu importe la pondération attribuée à l'examen ou au travail en question, fera face aux conséquences de ses gestes, qui peuvent aller jusqu'à l'exclusion de son programme de formation. Une politique stricte de tolérance zéro est appliquée en tout temps et sous toutes circonstances. Tous les cas seront référés à la direction du Département. 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;

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.
Attention, il s'agit ici de valider une démarche d'investigation. **Les justifications par la preuve des outils utilisés pour localiser puis corriger les erreurs sont plus importantes que les corrections du code en tant que tel.** Votre rapport doit comporter les étapes suivantes :

1. Cahier des charges : Quel est, à la lueur d'une première observation du code, le problème qu'est censé 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 par des preuves 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. C'est principalement ce moyen qui sera évalué. 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 par des preuves (copie d'écran **démontrant que l'erreur est corrigée**).

Si toutes les erreurs de compilation sont corrigées vous devriez avoir comme résultat de compilation un résultat similaire au suivant :

```
…

g++        -c  -g  -MMD  -MP  -MF  "build/Debug/GNU-Linux/fonctionsUtilitaires.o.d"  -o  build/Debug/GNU-
Linux/fonctionsUtilitaires.o fonctionsUtilitaires.cpp
mkdir -p build/Debug/GNU-Linux
rm -f "build/Debug/GNU-Linux/programmePrincipal.o.d"
g++        -c  -g  -MMD  -MP  -MF  "build/Debug/GNU-Linux/programmePrincipal.o.d"  -o  build/Debug/GNU-
Linux/programmePrincipal.o programmePrincipal.cpp
mkdir -p dist/Debug/GNU-Linux
g++               -o   dist/Debug/GNU-Linux/investigation   build/Debug/GNU-Linux/fonctionsUtilitaires.o
build/Debug/GNU-Linux/programmePrincipal.o
make[2] : on quitte le répertoire « /mnt/hgfs/tp_gif-1003_netbeans »
make[1] : on quitte le répertoire « /mnt/hgfs/tp_gif-1003_netbeans »

BUILD SUCCESSFUL (total time: 1s)
```

Il vous restera ensuite de vous assurer qu'il n'y a pas d'erreur d'exécution.

Deuxième partie : fonctions

Travail à réaliser

Il s'agit de constituer une librairie de fonctions utilitaires qui pourront être réutilisées dans le projet de session (TP2, 3 et 4).
Dans les fichiers `validationFormat.h` et `validationFormat.cpp`, vous devrez implanter les fonctions 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 et les tirets '-' sont permis s'ils ne sont pas doublés; deux (ou plus) espaces ou '-' consécutifs; un tiret ne pouvant pas être suivi d'un espace et inversement.

```
bool validerCodeIssn (const std::string& p_issn)
```
Cette fonction valide le code ISSN d'une publication en série entre autres par la vérification à l'aide du calcul de la clé (dernier chiffre). Voir https://fr.wikipedia.org/wiki/ISSN pour le format d'un code ISSN ainsi que pour le calcul de la clé.
Exemple de code valide :
`ISSN 1467-8640`
Le format doit être strictement respecté, soient, le type du code, les séparateurs et le numéro qui est vérifié par la clé.

```
bool validerCodeIsbn (const std::string& p_isbn)
```
Cette fonction valide le code ISBN d'un **livre** par la vérification à l'aide du calcul de la clé (dernier chiffre). On ne s'intéresse ici qu'au code ISBN-13. Voir fr.wikipedia.org/wiki/ISBN ou en.wikipedia.org/wiki/ISBN pour le calcul de la clé.
Exemple de codes valides :
`ISBN 978-2-7605-5379-8`
Le format doit être strictement respecté, soient, le type du code, les séparateurs, le préfixe, le domaine (voir liste exhaustive https://fr.wikipedia.org/wiki/Domaine_ISBN), le numéro éditeur (voir liste exhaustive fr.wikipedia.org/wiki/ISBN), le numéro de publication et la clé.

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 du bien livrable (très important)
2) lisibilité et pertinence du rapport d'investigation
3) **justification par des preuves de la localisation**
4) **justification par des preuves de l'efficacité des corrections**
5) explications et commentaires
6) modifications du code à corriger
7) portabilité de votre code source développé
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*

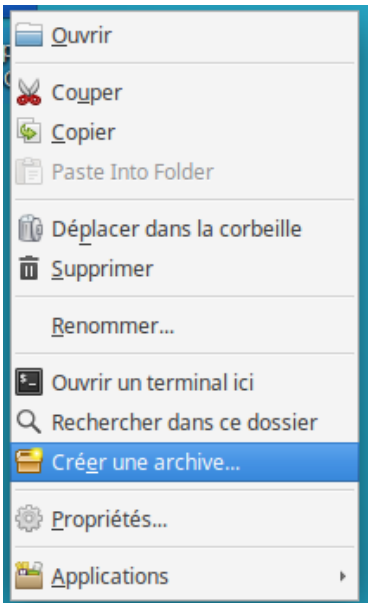
Le rapport d'investigation sera utilisé pour évaluer qualité 3, investigation du BCAPG, tel que précisé dans les objectifs spécifiques du plan de cours.

Bien livrable :

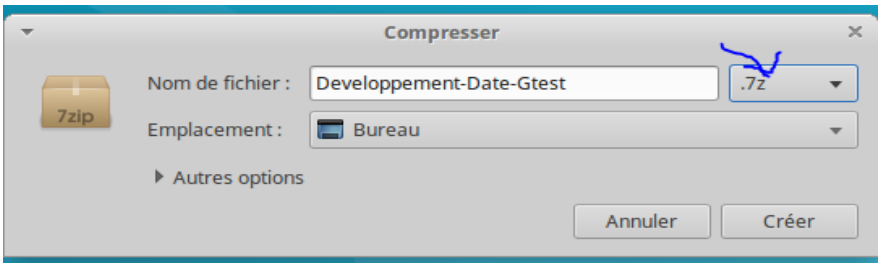
Vous devez rendre six fichiers `rapportInvestigation` (format doc ou pdf, au choix), `fonctionsUtilitaires.h`, `fonctionsUtilitaires.cpp` et `programmePrincipal.cpp` corrigés, `validationFormat.h` et `validationFormat.cpp` mis dans une archive **.7z**.

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 (portail).

Ce travail est intitulé TP1. **Aucune remise par courriel n'est acceptée**.
Vous pouvez remettre autant de versions que vous le désirez, pensez à effacer les anciennes pour ne laisser que celle qui sera corrigée. **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 30 septembre 2021 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