Automne 2015

GIF-1003

Thierry EUDE, Ph.D

Travail individuel
à rendre avant le
jeudi 1er octobre 2015 à 12h00
(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.
4. Solution (étape itérative liée à une itération de l'étape 3) : apportez la correction en illustrant son efficacité (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 :

```
07:36:36 **** Build of configuration Debug for project Adeboguer ****
make all
Building file: ../fonctionsUtilitairesSolution.cpp
Invoking: GCC C++ Compiler
g++   -O0   -g3   -Wall   -c   -fmessage-length=0   -MMD   -MP   -MF"fonctionsUtilitairesSolution.d"   -
MT"fonctionsUtilitairesSolution.d"                  -o                  "fonctionsUtilitairesSolution.o"
"../fonctionsUtilitairesSolution.cpp"
Finished building: ../fonctionsUtilitairesSolution.cpp

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

Building target: Adeboguer
Invoking: GCC C++ Linker
g++  -o "Adeboguer"  ./fonctionsUtilitairesSolution.o ./programmePrincipalSolution.o
Finished building target: Adeboguer


07:36:38 Build Finished (took 1s.909ms)
```

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 futur 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 valideLigneGEOJSON (std::string& p_ligne, std::ostringstream& p_parametres)`
Cette fonction permet d'extraire les informations relatives à des bornes de stationnement contenues dans une chaîne de caractères reçue en paramètre. Si cette chaine de caractères ne correspond pas à un format GEOJSON (voir le format ci-dessous), la fonction retourne false, true sinon. Les informations extraites sont alors stockées dans un ostringtream qui est également passé en paramètre.

Exemple d'un enregistrement provenant du fichier GEOJSON :
{"type":"Feature","geometry":{"type":"Point","coordinates":[-71.2217178685479,46.803835920695]},"properties":{"ID":"300070","COTE_RUE":"Ouest","LECT_MET":"208","SEGMENT_RU":"105","DIRECTION":null,"NOM_TOPOG":"Avenue Louis-St-Laurent","NO_BORNE":"2371","NO_CIVIQ":null,"ID_VOIE_PUB":105663,"GEOM":"POINT (249714.049 5185174.046)"}}

Le but de la fonction est d'extraire de la chaîne de caractères les propriétés suivantes:
ID, COTE_RUE, LECT_MET, SEGMENT_RU, DIRECTION, NOM_TOPOG, NO_BORNE, GEOM

`bool valideLigneCVS (std::string& p_ligne, std::ostringstream& p_parametres)`
Cette fonction permet d'extraire des informations relatives à des bornes de paiement contenues dans une chaîne de caractères reçue en paramètre. Si cette chaine de caractères ne correspond pas à un format CVS (voir le format ci-dessous), la fonction retourne false, true sinon. Les informations extraites sont alors stockées dans un ostringtream qui est également passé en paramètre.

Exemple d'un enregistrement provenant du fichier CSV :
300685|Sud|53,58|2,0||Boulevard René-Lévesque Ouest|B19|-71,225995|46,804545
Ces informations correspondent aux propriétés suivantes :
ID|COTE_RUE|LECT_MET|SEGMENT_RU|DIRECTION|NOM_TOPOG|NO_BORNE|LONGITUDE|LATITUDE.

Le but de la fonction est d'extraire de la chaîne de caractères les propriétés suivantes:
ID, COTE_RUE, LECT_MET, SEGMENT_RU, DIRECTION, NOM_TOPOG, NO_BORNE, LONGITUDE, LATITUDE.

`bool validerGeom(const std::string& p_geom)`
Cette fonction valide la longitude et la latitude de la borne de stationnement selon le standard WKT (Well-Known Text).
Le format à valider doit être le suivant :
POINT (longitude latitude) où longitude et latitude sont des éléments de type double.

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) portabilité de votre code source développé
5) Structure, organisation du code développé, lisibilité
6) 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 zip**.
Le premier travail pour le cours GIF-1003 est un **travail individuel.**

Vous devez remettre votre travail en utilisant le dépôt de l'ENA (https://www.portaildescours.ulaval.ca/lieninterne/redirection/64498/evaluation_boitedepot/158431)
Ce travail est intitulé TP #1. **Aucune remise par courriel n'est acceptée**.
Vous pouvez remettre autant de versions que vous le désirez. Seul le dernier dépôt est conservé. **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 1er octobre 2014 12h** (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