Go to homepage

SCALE FOR PROJECT CPP MODULE 02

You should evaluate 1 student in this team

Introduction

Merci de respecter les règles suivantes:

- Restez polis, courtois, respectueux et constructifs pendant le processus d'évaluation. Le bien-être de la communauté repose là-dessus.

- Identifiez avec la personne évaluée ou le groupe évalué les éventuels dysfonctionnements de son travail. Prenez le temps d'en discuter et débattez des problèmes identifiés.

- Vous devez prendre en compte qu'il peut y avoir de légères différences d'interprétation entre les instructions du projet, son scope et ses fonctionnalités. Gardez un esprit ouvert et notez de la manière la plus honnête possible. La pédagogie n'est valide que si la peer-évaluation est faite sérieusement.

Guidelines

- Ne notez que ce qui est contenu dans le dépôt Git cloné de l'étudiant(e) ou du groupe.

- Vérifiez que le dépôt Git appartient bien à l'étudiant(e) ou au groupe, que le projet est bien celui attendu, et que "git clone" est utilisé dans un dossier vide.

- Vérifiez scrupuleusement qu'aucun alias n'a été utilisé pour vous tromper et assurez-vous que vous évaluez bien le rendu officiel.

- Afin d'éviter toute surprise, vérifiez avec l'étudiant(e) ou le groupe les potentiels scripts utilisés pour faciliter l'évaluation (par exemple, des scripts de tests ou d'automatisation).

- Si vous n'avez pas fait le projet que vous allez évaluer, vous devez lire le sujet en entier avant de commencer l'évaluation.

- Utilisez les flags disponibles pour signaler un rendu vide, un programme ne fonctionnant pas, une erreur de Norme, de la triche... Dans ces situations, l'évaluation est terminée et la note est 0, ou -42 en cas de triche. Cependant, à l'exception des cas de triche, vous êtes encouragé(e)s à continuer la discussion sur le travail rendu, même si ce dernier est incomplet. Ceci afin d'identifier les erreurs à ne pas reproduire dans le futur.

- Si le sujet requiert un fichier de configuration, vous ne devriez jamais avoir à le modifier. Si vous souhaitez éditer un fichier, prenez le temps d'expliquer pourquoi à la personne évaluée et de vous assurer que vous avez son accord.

- Vous devez aussi vérifier l'absence de fuites mémoire. Toute mémoire allouée sur le tas doit être libérée proprement avant la fin de l'exécution du programme.
Vous avez le droit d'utiliser tout outil disponible sur la machine tel que leaks, valgrind ou e_fence. En cas de fuites mémoire, cochez le flag approprié.

Attachments

subject.pdf

Tests préliminaires

Si un cas de triche est suspecté, la notation et l'évaluation prennent fin immédiatement. Pour le signaler, sélectionnez le flag "Cheat". Faites attention à l'utiliser avec calme, précaution et discernement.

Prérequis

Le code doit compiler avec c++ et les flags -Wall -Wextra -Werror
Pour rappel, ce projet doit suivre le standard C++98. Par conséquent, des fonctions C++11 (ou autre standard) et les containers ne sont PAS attendus.

Ne notez pas l'exercice si vous trouvez :

- Une fonction implémentée dans un fichier d'en-tête (sauf pour les fonctions templates).
- Un Makefile compilant sans les flags demandés et/ou avec autre chose que c++.

Sélectionnez le flag "Fonction interdite" (Forbidden function) si vous rencontrez :

- L'utilisation d'une fonction "C" (*alloc, *printf, free).
- L'utilisation d'une fonction interdite dans le projet.
- L'utilisation de "using namespace <ns_name>" ou du mot-clé "friend".
- L'utilisation d'une bibliothèque externe, ou de fonctionnalités propres aux versions postérieures à C++98.

Yes | No

Exercice 00 : Mon premier canon

Cet exercice introduit la notion de classe canonique avec un exercice arithmétique simple : les nombres à point fixe.

Makefile

Il y a un Makefile qui compile en utilisant les flags appropriés.

Yes | No

Accesseurs

La classe Fixed (ou autre) doit avoir des accesseurs pour la valeur brute :

- int getRawBits(void) const;
- void setRawBits(int const raw); Ces membres sont-ils présents et fonctionnels ?

Yes | No

Classe canonique

Une classe canonique doit avoir au moins :

- Un constructeur par défaut
- Un destructeur
- Un constructeur par recopie
- Un opérateur d'affectation Ces élements sont-ils présents et fonctionnels ?

Yes | No

Exercice 01 : Premiers pas vers une classe utile

L'exercice précédent était un bon premier pas. Cependant, la classe était peu utile puisqu'elle ne permettait de représenter que la valeur 0.0

Makefile

Il y a un Makefile qui compile en utilisant les flags appropriés.

Yes | No

Constructeur via flottant

Peut-on construire une instance à partir d'un nombre flottant ?

Yes | No

operateur <<

Y a-t-il un opérateur << et est-il fonctionnel ?

Yes | No

Point fixe vers entier

La classe doit inclure une fonction membre "int toInt(void) const,"
qui convertit un nombre à point fixe en int.
Est-elle présente et fonctionnelle ?

Yes | No

Point fixe vers float

La classe doit inclure une fonction membre "float toFloat(void) const,"
qui convertit un nombre à point fixe vers un float.
Est-elle présente et fonctionnelle ?

Yes | No

Construction avec un int

Peut-on instancier la classe avec le constructeur prenant un int ?

Yes | No

Exercice 02 : Maintenant, on peut parler

Cet exercice ajoute les opérateurs de comparaison et arithmétiques à la classe.

Makefile

Il y a un Makefile qui compile en utilisant les flags appropriés.

Yes | No

Opérateurs de comparaison

Les 6 opérateurs de comparaison (>, <, >=, <=, == et !=) sont-ils présents et fonctionnels ?

Yes | No

Opérateurs arithmétiques

Les 4 opérateurs arithmétiques (+, -, * et /) sont-ils présents et fonctionnels ?
(Si vous effectuez une division par 0, il est acceptable que le programme crash.)

Yes | No

Autres opérateurs

Les quatre opérateurs d'incrémentation et de décrémentation
(pré-incrémentation et post-incrémentation, pré-décrémentation et
post-décrémentation) sont-ils présents et fonctionnels ?

Yes | No

Surcharge de fonctions membres statiques publiques

Pour finir, vérifiez que les fonctions membres statiques min() et max()
sont implémentées et fonctionnelles.

Yes | No

Exercice 03 : BSP

Cet exercice devrait vous faire réaliser à quel point il est facile de mettre en œuvre des des algorithmes complexes une fois que les bases fonctionnent comme prévu.

Makefile

Il y a un Makefile qui compile en utilisant les flags appropriés.

Yes | No

Classe Point

Il existe une classe Point qui possède deux attributs (x et y) de type Fixed const.
Elle possède également un constructeur qui prend deux flottants et initialise x et y avec ces valeurs.

Yes | No

Fonction bsp

Il existe une fonction bsp() dont le prototype est "bool bsp(Point const a, Point const b, Point const c, Point const point)".
La fonction renvoie True si le point est à l'intérieur du triangle décrit par les sommets a, b, et c. Sinon, elle renvoie False.

Yes | No

Main et tests

Il y a un main pour tester que la fonction bsp() fonctionne comme décrit ci-dessus. Exécutez plusieurs tests pour vous assurer que la valeur de retour est correcte.

Yes | No

Ratings

Don't forget to check the flag corresponding to the defense

Ok | Outstanding project

Empty work | Incomplete work | Invalid compilation | Cheat | Crash

Concerning situation | Leaks | Forbidden function

Can't support / explain code

Conclusion

Give this repository a star. ⭐