UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 339

Laboratoire#1 : Écriture de classes en C++

Hiver 2026

Le but de ce laboratoire est de pratiquer les classes, l'héritage, l'utilisation des pointeurs, et aussi d'apprendre à lire et compléter le code écrit par quelqu'un d'autre. Nous avons vu en cours l'exemple de conception d'une application permettant de créer un ensemble de figures géométriques dans un plan et de les déplacer.

Le code de base de l'application vous est fourni. Les classes `point`, `figure` et `triangle` vous sont entièrement fournies. Vous devez compléter les classes `cercle`, `polygone`, et `plan`, et écrire entièrement la classe `rectangle` afin de compléter l'application.

Ce devoir est à faire en équipe de deux. Il devra être complété avant le vendredi 30 janvier 2026 à 23h59. Vous devez remettre, sur `turnin.dinf.usherbrooke.ca`, les fichiers générés au cours de ce laboratoire.

Description de la tâche à réaliser

L'application comporte sept (7) types : `Point`, `Figure`, `Cercle`, `Polygone`, `Triangle`, `Rectangle`, et `Plan`. Tous ces types comportent au moins les opérateurs suivants permettant de déplacer les objets et les afficher :

- `translater : Point` $\rightarrow \emptyset$: permet de déplacer l'objet par translation suivant un vecteur représenté par un Point donné en paramètre d'entrée.
- `tournerOrigine : réel` $\rightarrow \emptyset$: permet de déplacer l'objet par rotation autour de l'origine du plan suivant un angle donné en paramètre d'entrée.
- `tourner : Point` $\times$ `réel` $\rightarrow \emptyset$: permet de déplacer l'objet par rotation autour d'un centre suivant un angle. Le centre et l'angle de la rotation sont donnée en paramètres d'entrée.
- `afficher` : $\emptyset \rightarrow \emptyset$: permet d'afficher la représentation de l'objet.

Les types correspondant à des figures géométriques comportent également les opérateurs suivants permettant de calculer la circonférence et l'aire des objets :

- `calculerCirconference` : $\emptyset \rightarrow$ `réel` : permet de calculer la circonférence de l'objet. Dans le cas du Plan, il s'agit de calculer la somme des circonférences des figures du plan.
- `calculerAire` : $\emptyset \rightarrow$ `réel` : permet de calculer l'aire de l'objet. Dans le cas du Plan, il s'agit de calculer la somme des aires des figures du plan.

Le code comporte les fichiers suivants :

— `point.h` et `point.cpp` : spécifications et implémentations des opérations du type Point (fournis complètement)
— `figure.h` : spécifications et implémentations des opérations du type Figure (fourni complètement)
— `cercle.h` : spécifications du type Cercle (fourni complètement)
— `cercle.cpp` : implémentations des opérations du type Cercle (**à compléter**)
— `polygone.h` : spécifications du type Polygone (fourni complètement)
— `polygone.cpp` : implémentations des opérations du type Polygone (**à compléter**)
— `triangle.h` et `triangle.cpp` : spécifications et implémentations des opérations du type Triangle (fournis complètement)
— `rectangle.h` et `rectangle.cpp` : spécifications et implémentations des opérations du type Rectangle (**à écrire entièrement**)
— `plan.h` : spécifications du type Plan (fourni complètement)
— `plan.cpp` : implémentations des opérations du type Plan (**à compléter**)

Les opérateurs des classes `Cercle`, `Polygone` et `Plan` à coder sont indiqués par le commentaire `/*... a completer ...*/` dans les fichiers `cercle.cpp`, `polygone.cpp` et `plan.cpp`. Voir les spécifications des opérateurs dans les fichiers "headers" `.h` correspondant. Les fichiers `rectangle.h` et `rectangle.cpp` sont à écrire entièrement. La classe `Rectangle` doît comporter au moins le constructeur spécifié ci-après : `Rectangle : Point` $\times$ `Point` $\times$ `Point` $\times$ `Point` $\rightarrow \emptyset$, permettant de construire un nouvel objet Rectangle à partir de ses quatre sommets donnés en paramètres d'entrée.

Inspirez-vous des opérateurs et classes déjà codés pour faire du code cohérent avec ce qui existe déjà. Vous devriez adopter un plan systématique de codage et de test. Vous devriez d'abord tout concevoir sur papier. Puis, codez les fonctions une par une dans un ordre qui vous permet de les tester de façon aussi indépendante que possible. Un exemple de fichier de test `main.cpp` est fourni avec le code de départ. Une fois votre programme complété, vous devez vous assurer qu'il compile et fonctionne bien sous Linux en le testant sur n'importe quelle machine Linux du Département d'informatique. Les commandes pour compiler et exécuter le programme dans un terminal sous linux sont fournies dans le script `compilation_et_execution.sh`. N'oubliez de rajouter dans la ligne de commande le fichier `rectangle.cpp` lorsque vous l'aurez écrit.

Remise du travail

Pour soumettre votre travail, connectez-vous, dans un fureteur, au serveur `http://turnin.dinf.usherbrooke.ca` en utilisant votre CIP, puis choisissez le cours `IFT339` et le projet `TP1`. Chargez vos fichiers `cercle.cpp`, `polygone.cpp`, `plan.cpp`, `rectangle.h` et `rectangle.cpp` et soumettez-les. Indiquez bien les noms des deux membres de l'équipe, le cas échéant, en commentaire dans chaque fichier. Ne faites qu'une seule soumission par équipe. Ne remettez pas d'autre fichier, ni d'exécutable. Vos fichiers de code seront intégrés à un programme de test contenant déjà les autres fichiers du programme. Vous n'avez donc pas à re-soumettre ces derniers.

Barême

- 25 points pour soumission réussie d'un programme qui compile sans erreur
- 10 points pour respect des normes de programmation (se référer au document sur les normes de programmation sur le site web du cours)
- 40 points pour le respect de la conception et des instructions fournies (spécifications des opérateurs et instructions de remise)
- 25 points la complétion correcte du code (5 points pour chacun de `cercle.cpp`, `polygone.cpp`, `plan.cpp`, `rectangle.h` et `rectangle.cpp`)