Compte rendu CPP n°3

Templates

STEPHANT André-Louis, COUSSEAU Yanis

Ecole Centrale de Nantes - Systèmes embarqués Communicants (SEC)

Table des matières

1 Introduction

Ce troisième TP à pour objectif de nous faire travailler une nouvelle notion en C++ : les templates. Pour rappel, un template est une manière de simplifier un code, permettant à une fonction d'accepter différents types de paramètres en entrée.

Dans ce TP, nous allons utiliser cette notion de templates avec des classes, ainsi que des surcharges d'opérateurs. L'objectif est de mêler plusieurs concepts différents afin de mener à bien un projet.

Dans un premier temps, nous allons créer une classe Point template. Puis, nous coderons une classe forme, dérivée de la classe précédente. Ensuite, nous utiliserons une classe rectangle, héritée de la classe forme. Enfin, nous spécialisrons nos classe, puis créerons une liste de forme.

Nous concluerons ce rendu en mettant en lumière les différentes compétences sollicitées pour ce TP.

2 Création d'une classe point template

Dans cette première partie, nous allons initialiser le premier template de notre projet : la classe point. Pour rappel, voici le cahier des charges associé à cette classe :

Écrire une classe pour un point du plan. Outre les attributs d'abscisse et d'ordonnée, elle devra comporter :

- une méthode `translater()` qui prend en argument respectivement une paire déléments du même type que les coordonnées
- un constructeur qui prend deux paramètres
- un constructeur de recopie qui devra prendre en paramètre une référence constante à un point, c'est-à-dire un `point const &`. La référence est en réalité un pointeur mais s'utilise comme si on avait affaire à un objet passé par valeur
- des accesseurs et des mutateurs donnant accès aux valeurs des attributs et permettant de changer ces valeurs
- Surcharger l'opérateur « sous forme de fonction amie de façon à permettre l'utilisation de `cout` avec des points.

Vous expliciterez la classe `point` sous forme de diagramme de classes dans votre rapport. Vous écrirez l'ensemble du code dans `point.hpp` ainsi quun programme principal permettant de tester vos différentes méthodes. Vous présenterez vos tests sous forme de jeux d'essais dans votre compte rendu.

Figure 1: Cahier des charges partie 1

Par conséquent, on crée un fichier .hpp afin de répondre au cahier des charges :

```
1 Ctrl C + Ctrl V le code ICI
```

Listing 1: Fichier PointT.hpp

On ajoute ensuite des jeux de test afin de valider le bon fonctionnement du programme :

Donnée d'entrée : NA

Résultat attendu : NA

Résultat obtenu : NA

On peut donc valider que la classe fonctionne comme attendue.

3 Formes géométriques abstraites

Dans cette seconde partie, nous allons créer une classe afin de manupiler une forme, constituée de points. L'objectif est de réaliser une forme centrée sur un point particulier (initialisé dans la partie précédente). Pour rappel, voici les instructions données dans le sujet de TP associées à cette classe :

1. Écrire une classe `forme` qui aura ce point comme attribut (centre de la forme) avec le constructeur approprié.
2. Surcharger l'opérateur « de façon à permettre l'utilisation de `cout` avec des formes, toujours sous forme d'une fonction amie
3. Déclarer deux méthodes abstraites `perimetre()` et `surface()`. Attention les types de retour de ces deux fonctions devront être génériques mais pas forcément le même que celui de l'abscisse et de l'ordonnée des points (pour éviter les pertes de précision quand on utilise des entiers) .

Figure 2: Cahier des charges partie 1

Par conséquent, on crée un fichier .hpp afin de répondre au cahier des charges :

```
1 Ctrl C + Ctrl V le code ICI
```

Listing 2: Fichier Forme.hpp

On ajoute ensuite des jeux de test afin de valider le bon fonctionnement du programme :

Donnée d'entrée : NA

Résultat attendu : NA

Résultat obtenu : NA

On peut donc valider que la classe fonctionne comme attendue.

4 Formes géométriques concrètes

On veut ici réaliser des formes géométriques concrètes. C'est à dire une forme consituée de points, avec une longueur et une hauteur donnée. On va créer ici deux classe (rectangle et carré, avec notion d'héritage pour cette dernière).

On crée donc deux fichiers .hpp :

```
1 Ctrl C + Ctrl V le code ICI
```

Listing 3: Fichier Rectangle.hpp

```
1 Ctrl C + Ctrl V le code ICI
```

Listing 4: Fichier Carre.hpp

On ajoute ensuite des jeux de test afin de valider le bon fonctionnement du programme :

Donnée d'entrée : NA

Résultat attendu : NA

Résultat obtenu : NA

On peut donc valider que la classe fonctionne comme attendue.

5 Conclusion

Ce TP nous a permis de découvrir et d'approfondir la notion de template en C++. Nous avons pu mettre en pratique cette notion à travers la création de plusieurs classes, ainsi que l'utilisation de l'héritage et des surcharges d'opérateurs.

Nous avons également renforcé notre compréhension des concepts liés au classes, ainsi que du langage C++ d'une manière générale. Ces notions nous serons importantes pour l'avenir, notamment dans le cadre du projet de fin de module à venir.