TP n°4 : Surdéfinition des opérateurs et patrons de classes

(Date de remise: Le 15 Décembre 2017 à 23 h 55)

Exercice 1– (6 points)

Créer une classe **`rectangle`** qui possède les données membres **`longueur`** et **`largeur`** de type `double` et les fonctions membres suivantes (en plus des constructeurs nécessaires) :

Fonction membre	Explication
`void setDimension(double long, double larg)`	Changer la longueur et la largeur du rectangle.
`double getLong() const`	Retourner la longueur du rectangle.
`double getLarg() const`	Retourner la largeur du rectangle.
`double surface() const`	Retourner la surface du rectangle.
`double perimeter() const`	Retourner le périmètre du rectangle.

Par ailleurs, la classe rectangle doit surdéfinir les opérateurs suivants :

Opérateur	Explication
`+`	Addition de deux rectangles (en additionnant leurs longueurs et largeurs respectives).
`*`	Multiplication de deux rectangles (en multipliant leurs longueurs et largeurs respectives).
`== et !=`	Comparaison de deux rectangles (en comparant leurs longueurs et largeurs respectives).
`<< et >>`	**Flots** de sortie et d'entrée pour afficher et lire la longueur et la largeur d'un rectangle.
`++ (pré et post incrémentation)`	Incrémenter la longueur et la largeur d'un rectangle d'une unité.

La fonction `main()` de votre programme doit créer des objets de type rectangle pour tester toutes les fonctions membres et les opérateurs surdéfinis.

Nommer les fichiers du code source `rectangles.hh` et `rectangles.cpp`.

Exercice 2– (6 points)

Créer une classe **`monTableau`** qui résoudra le problème de dépassement de capacité lors de la manipulation de tableaux en C++. Cette nouvelle classe permettra à un utilisateur de designer n'importe quel entier (positif ou négatif) comme le premier indice d'un tableau. Chaque objet de type `monTableau` contient un tableau d'un type quelconque noté **Type**. Lors de l'exécution, en accédant à

un élément d'un tableau de type monTableau et s'il y a dépassement de capacité (c.-à-d. si l'indice utilisé dépasse la taille du tableau en question), le programme doit se terminer avec un message d'erreur approprié.

Soit les instructions suivantes :

```
monTableau<int> tabInt(5);            // (1)

monTableau<double> tabDouble(2, 13); // (2)

monTableau<char> tabChar(-5, 9);     // (3)
```

- L'instruction **(1)** déclare `tabInt` comme étant un tableau de 5 éléments de type `int` : `tabInt[0]`, `tabInt[1]`,…, `tabInt[4]`.
- L'instruction **(2)** déclare `tabDouble` comme étant un tableau de 11 éléments de type `double` : `tabDouble[2]`, `tabDouble[3]`,…, `tabDouble[12]`.
- L'instruction **(3)** déclare `tabChar` comme étant un tableau de 14 éléments de type `char` : `tabChar[-5]`, `tabChar[-4]`,…, `tabChar[8]`.

Noter l'utilisation des patrons de classes pour que le tableau représenté par la classe monTableau puisse être de type quelconque.

Par ailleurs, il vous faudra surdéfinir les opérateurs suivants dans la classe `monTableau` : **`[]`**, **`==`** et **`!=`**.

La fonction `main()` de votre programme doit créer des objets de type `monTableau` pour tester toutes les fonctions membres et les opérateurs surdéfinis de cette classe.

Nommer les fichiers du code source `monTableau.h` et `monTableau.cpp`.

Remise du TP

- Ce travail compte pour 12 % de la note finale du cours.
- Date limite de remise : Le 15 Décembre 2017 à 23 h 55
- Il est recommandé de faire ce travail en équipe de 2 personnes (**maximum**). Dans ce cas, un seul des coéquipiers doit remettre le travail alors que l'autre doit compléter et remettre le fiche **RemiseCoequipier.docx** via **Studium**.
- La remise doit se faire de façon électronique via **Studium**. Aucune remise par courriel ne sera acceptée.
- Vous devez remettre un seul dossier compressé contenant les fichiers du code source (`rectangles.h`, `rectangles.cpp`, `monTableau.h` et `monTableau.cpp`). Ces fichiers doivent contenir les noms des auteurs du code source en en-tête (sous forme de commentaire) et les résultats d'exécution (tests) à la fin (sous forme de commentaire aussi).
- La non-remise électronique (volontaire ou par erreur) est sanctionnée par la note de 0.

- Un programme qui ne compile pas est sanctionné par la note de 0.
- Un programme qui compile, mais ne réalise pas la logique prévue dans la spécification est sanctionné par la note de 0.
- Les avertissements (*warnings*) non corrigés : cela dépend de la quantité : à partir de – 0.25 et plus.
- Pour toutes questions concernant ce travail pratique, envoyer un courriel à **khlifaym@iro.umontreal.ca** et **belbekka@iro.umontreal.ca**.

Bon travail !