

## ISIMA 3<sup>ème</sup> année - MODL/C++

### TP 3 : STL

On souhaite réaliser une application permettant d'analyser des données statistiques : on considère un échantillon de valeurs et l'on veut obtenir un histogramme qui répartit les valeurs dans des « classes » (au sens statistique). Essayez d'utiliser au maximum les fonctionnalités de la STL.

- 1) Définir une classe **Echantillon** regroupant des valeurs (classe **Valeur**). Pour la classe **Echantillon**, un simple vecteur suffira *a priori*. Pour la classe **Valeur**, on stockera dans un premier temps un simple réel. Tests 1-4

Dans la classe **Echantillon**, proposer des méthodes pour ajouter des valeurs, obtenir les valeurs minimum et maximum, et accéder à n'importe quelle valeur à partir de son indice. Gérer les erreurs éventuelles à l'aide d'exceptions. Tests 5-9

- 2) Définir la classe **Histogramme**. Elle contient un ensemble de "classes" (au sens statistique), un vecteur suffira initialement. La classe **Classe** représentera une "classe", c'est-à-dire une borne inférieure, une borne supérieure et une quantité. Tests 10-11

L'intervalle de valeurs (bornes inférieure et supérieure) de l'histogramme, ainsi que le nombre de classes seront fournis à la construction, afin de générer automatiquement des classes couvrant l'intervalle de valeurs avec toutes la même amplitude. Test 12

- 3) Compléter la classe **Histogramme** pour construire un histogramme à partir d'un échantillon. Test 13

On souhaite maintenant pouvoir obtenir une vision alternative de l'histogramme dans laquelle les "classes" sont présentées par ordre décroissant de quantité (ainsi la première classe sera celle contenant la plus grande quantité de valeurs de l'échantillon, la dernière étant celle contenant la plus petite quantité).

- 4) On propose l'approche suivante : la classe **Histogramme** utilise un ensemble (`std::set`) plutôt qu'un vecteur pour stocker les classes. Par défaut, l'ensemble est trié en fonction du résultat de l'opérateur `<` sur les éléments stockés. Définir l'opérateur `<` pour les classes de manière à ce qu'elles soient rangées par ordre croissant de leur borne inférieure. Test 12
- 5) Modifier la classe **Histogramme** pour que l'on puisse choisir l'ordre dans lequel ranger les classes. Par défaut, l'ensemble utilise le foncteur `std::less` de la STL pour trier les éléments, mais il est possible de définir son propre foncteur (opérateur binaire qui compare deux éléments) pour trier les classes par quantité décroissante notamment. Test 14

La classe devient donc générique avec comme paramètre le type du foncteur qui va servir à trier les classes. Le foncteur sera fourni à la construction de la classe **Histogramme**. Essayer d'abord cette nouvelle classe avec le foncteur `std::greater` (entête `<functional>`, nécessite l'opérateur `>`). Test 15

Et ensuite avec un foncteur (**ComparateurQuantite** à écrire) permettant un tri des classes par quantité décroissante (si deux classes ont la même quantité, l'ordre des bornes inférieures est appliqué). Test 16

- 6) On ne peut pas changer dynamiquement de relation d'ordre sur les classes. Proposer un moyen de convertir un histogramme avec une certaine relation d'ordre en un histogramme avec une autre relation d'ordre. Test 17

Dans notre modélisation, nous avons oublié que les notes étaient associées à des étudiants. Nous allons à présent réparer cet oubli.

- 7) Modifier la classe **Valeur** pour prendre en compte à la fois un étudiant (son nom) et sa note. Tests 18-21
- 8) Ajouter une structure dans l'histogramme permettant de conserver les valeurs se trouvant dans chacune des classes. Les classes **Valeur** et **Classe** n'ont pas à être modifiées, on propose d'utiliser une structure associative à clé multiple (`std::multimap`) de manière à pouvoir associer plusieurs valeurs à une classe. Tests 22
- 9) Ecrire une méthode dans la classe **Histogramme** pour obtenir la liste de toutes les valeurs d'une "classe". Il s'agit d'identifier les valeurs associées à la classe dans la *multimap* définie précédemment. Test 23
- 10) A partir de la question précédente, surcharger l'opérateur << pour la classe **Histogramme** afin d'afficher les classes en fonction de l'ordre défini dans l'histogramme, et pour chaque classe, la liste des étudiants avec leur note.