ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

TP ILO C++
MeanValue

Prérequis

- Ce TP se présente sous la forme d'une classe C++ à compléter dont vous pourrez trouver un squelette d'implémentation dans /pub/ILO/MeanValue.zip
- Référence C++ :
  - http://www.cplusplus.com/reference/

MeanValue<T, R>

MeanValue<T, R> est un patron de classe destiné à accumuler des valeurs de type T afin de pouvoir par la suite lui demander la valeur moyenne de ces valeurs et l'écart type exprimés dans le type R ainsi que les valeurs min et max de toutes les valeurs de type T accumulées.

Exemple :

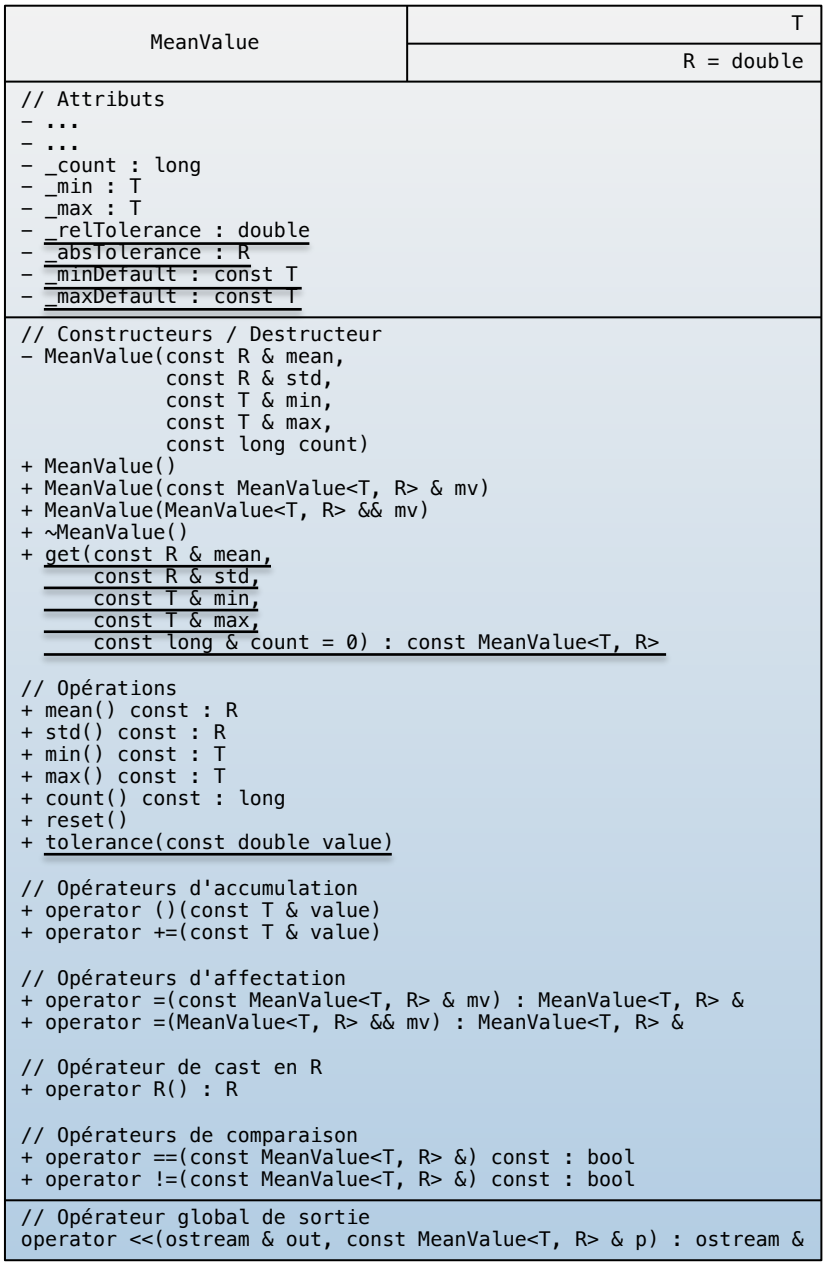
```
MeanValue<int, double> m;

for (size_t i = 0; i < nbtests; ++i)
{
        m(<nouvelle valeur>);
        // ou bien
        m += <nouvelle valeur>;
}

double moyenne = m.mean();
double ecarttype = m.std();
int minimum = m.min();
int maximum = m.max();
long compte = m.count();
```

Les opérations sont les suivantes :

- mean : permet d'obtenir la valeur moyenne des valeurs accumulées.
- std : permet d'obtenir l'écart type des valeurs accumulées.
- min : permet d'obtenir la plus petite des valeurs accumulées.
- max : permet d'obtenir la plus grande des valeurs accumulées.
- count : permet d'obtenir le nombre de valeurs accumulées jusqu'à présent.
- reset : permet de remettre à zéro toutes les valeurs fournies par les méthodes précédentes.



Les constructeurs sont les suivants :

- Un constructeur valué (privé, à utiliser en conjonction avec la méthode de classe « `get` ».
- Un constructeur par défaut
- Un constructeur de copie.
- Un constructeur de déplacement.
- Une méthode de classe « `get` » permettant d'obtenir une instance constante de MeanValue avec ses valeurs de moyenne / écart type / min & max préremplies. Une telle instance ne pourra donc pas accumuler des valeurs mais nous servira lors de la comparaison de deux MeanValues avec l'opérateur ==.

Les opérateurs sont les suivants :

- `void operator ()(const T & v)` aussi appelé opérateur d'appel de fonction permet d'accumuler une nouvelle valeur « v ».
- `void operator +=(const T & v)` permet aussi d'accumuler une nouvelle valeur (il est d'ailleurs conseillé d'appeler l'opérateur précédent dans cet opérateur).
- `MeanValue<T, R> & operator =(const MeanValue<T, R> & m)` est l'opérateur de copie permettant de copier les attributs de m dans `*this`.
- `MeanValue<T, R> & operator =(MeanValue<T, R> && m)` est l'opérateur de déplacement permettant de déplacer les attributs de m dans *this. Les attributs de m après cette opération sont remis à zéro.
- `operator R()` est l'opérateur de cast (transtypage) en `R` et permet d'obtenir la valeur moyenne lorsqu'une instance de `MeanValue<T, R>` est castée dans le type `R`. En reprenant l'exemple précédent, on aurai donc pu écrire : `double moyenne = (double)m;`
- `bool operator ==(const MeanValue<T, R> & m)` est l'opérateur de comparaison de l'instance courante avec l'instance m. On considèrera que deux instances sont identiques si les valeurs de leurs attributs internes sont identiques à une tolérance près (d'où les attributs `_relTolerance` & `_absTolerance`). Vous pourrez comparer deux valeurs quelconques à une tolérance près en utilisant la fonction `bool checkClose(…)` définie dans les mêmes fichiers que le patron MeanValue.
- `bool operator !=(const MeanValue<T, R> & m)` est l'opérateur de différence. Il réalise l'opération inverse de l'opérateur ==. Il est donc conseillé d'utiliser l'opérateur précédent dans cet opérateur.

Et enfin l'opérateur `ostream & operator <<(ostream & out, const MeanValue<T, R> & mv)` permet d'envoyer mv dans le flux de sortie standard (`out`) en affichant :

`<mean value> ± <std value> [<min value>...<max value>](<count value>)`

(les <> indiquent les valeurs à afficher et ne doivent pas faire partie de l'affichage)

Rappels mathématiques :

Soit X_n une suite de n valeurs quelconques $\{x_1, \cdots, x_n\}$.
Moyenne :

$$E(X_n) = \frac{1}{n}(x_1 + \cdots + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$E(X_0) = 0$$
$$E(X_n^2) = \frac{1}{n}\sum_{i=1}^{n}(x_i)^2$$

Écart Type :

$$\sigma(X) = \sqrt{E(X^2) - E(X)^2}$$

Pour calculer une moyenne ou un écart type dans la classe `MeanValue<T, R>`, on peut utiliser la somme des valeurs $\sum_{i=1}^{n} x_i$ et la somme des valeurs au carré $\sum_{i=1}^{n}(x_i)^2$ (avec le type `T`) mais ces sommes peuvent atteindre des valeurs importantes lorsque le nombre de valeurs n devient grand ce qui risque de provoquer un dépassement (et incidemment une inversion de signe) si `T` est de type entier. Une autre approche consiste à calculer $E(X_n)$ et $E(X_n^2)$ de manière incrémentale :

$$E(X_{n+1}) = \frac{1}{n+1}\sum_{i=1}^{n+1} x_i = \frac{1}{n+1}\left[\sum_{i=1}^{n} x_i + x_{n+1}\right] = \frac{1}{n+1}\left[nE(X_n) + x_{n+1}\right]$$
$$E(X_{n+1}^2) = \frac{1}{n+1}\left[nE(X_n^2) + (x_{n+1})^2\right]$$

Travail à effectuer :

Complétez la classe `MeanValue<T, R>` et faites tourner les programmes de test contenu dans TestMeanValue.cpp et/ou bien dans main.cpp (Voir l'annexe page 3).

Si vous ne voulez (ou ne pouvez) pas utiliser le programme de test contenu dans TestMeanValue.cpp qui utilise la librairie Boost, supprimez « TestMeanValue » du Makefile (ligne 112). Lorsque vous utilisez le programme principal (main.cpp), vous pouvez décommenter les sorties de la fonction checkClose (clog << …) pour voir les valeurs comparées.

Lorsque vous aurez fini votre projet vous pourrez le déposer sur le dépôt « ilo-meanvalue » du serveur exam.ensiie.fr jusqu'au 28/04/2018.

Recommandations

- Réutilisez autant que faire se peut les méthodes déjà présentes dans la classe MeanValue<T, R>. Ou plus exactement ne réécrivez jamais deux fois la même chose !
- Lorsque vous avez fini d'implémenter une méthode, supprimez le commentaire contenant le TODO, cela nous permettra de corriger votre code plus rapidement.
- N'attendez pas la fin du TP pour compiler votre projet et lancer les programmes de test : Faites-le dès le début (la plupart des tests échoueront) et régulièrement (jusqu'à ce que l'ensemble des tests réussissent).
- Ne modifiez en aucun cas les définitions de `MeanValue<T, R>` dans MeanValue.h ou les programmes de test (TestMeanValue.cpp, ou main.cpp)
- Respectez les mentions « Ne rien écrire ici » dans le code à compléter.
- Pour fabriquer votre archive de code lors du rendu, placez-vous à la racine du projet et tapez la commande « make clean archive » qui fabriquera un fichier zip dans le sous-répertoire « archives » contenant tous les fichiers sources de votre projet.

Annexes

Importer un projet

- Se mettre dans la perspective C/C++ et fermer la perspective Java.
- Import … → Existing Projects into Workspace → Select root directory : naviguez jusqu'au répertoire où vous avez dézippé l'archive, le projet devrait apparaître dans la liste des projets (il n'est pas nécessaire de copier le projet dans le workspace, vous pouvez le laisser là où il se trouve).
- L'étape de build n'est pas automatique en C++, il faut la lancer explicitement avec Crtl-B.

Archiver un projet (pour le déposer sur exam)

- Renommez votre projet :TP MeanValues (nom.prenom). Ceci nous facilitera l'importation de vos projets.
- Exportez votre projet sous forme d'archive : Export…→Archive File, puis sélectionnez les fichiers à exporter en ne sélectionnant **aucun** sous répertoire et **que** les fichiers mentionnés ci-dessous :
    - .cproject
    - .project
    - Makefile
    - MeanValue.cpp
    - MeanValue.h
    - TestMeanValue.cpp
    - main.cpp
- Puis sélectionnez le nom et l'emplacement de votre archive.
- Vous n'avez pas besoin d'ajouter votre nom à l'archive lors du dépôt le système de gestion des dépôt ajoutera vos prenom.nom au nom de l'archive déposée.

Créer un projet (Si besoin uniquement)

- Se mettre dans la perspective C/C++ et fermer la perspective Java.
- Dans le Projet Explorer : Menu Contextuel → New → C++ Project
- Dans C++ Project :
    - Donner un nom au projet
    - Location, Browse : indiquez le répertoire où vous avez dézippé le code fourni
    - Project Type : Makefile project → Empty Project
    - Tool Chains : Linux GCC
- L'étape de build n'est pas automatique en C++, il faut la lancer explicitement avec Crtl-B.

Lancer le programme de test (lorsque la compilation a réussi). Vous pouvez lancer soit le programme principal (main) qui utilise de simples assertions (comme en C), soit le programme de test (TestMeanValue) qui utilise la librairie Boost (vérifier la présence de /usr/include/boost) ainsi que le plugin « C/C++ Unit Testing Support » (à vérifier dans le Menu Help→Installation Details).

- Pour lancer le programme principal :
    - Dans le Project Explorer → Binaries → main → Menu contextuel → Run As … → Local C/C++ Application
- Pour lancer le programme de test
    - Lorsque le plugin « C/C++ Unit Testing Support » est installé : Dans le Project Explorer → Binaries → TestMeanValue → Menu contextuel → Run As … → Run Configuration…
        - Dans Run Configuration :
            - Double cliquer sur « C/C++ Unit » pour créer une nouvelle configuration de test.
            - Dans l'onglet « C/C++ Testing » de la configuration, choisir le « Boost Test Runner ».
            - Run
    - Lorsque le plugin « C/C++ Unit Testing Support » n'est pas installé : procédez comme pour le programme principal mais avec le binaire TestMeanValue.