IDM et Simulation TP 2

Ingénierie des modèles & Simulation

But du TP:

A) Découvrir et tester une technique de méta-programmation (génération de code par un autre programme). L'appliquer à une simulation de Monte Carlo et évaluer les performances.

B) Concevoir en C++ les bases d'un modèle capable de gérer des métadonnées pour mettre en œuvre quelques mécanismes d'introspection.

Première partie :

1. Chercher le code source en C du générateur « original » Mersenne Twister de Mastumoto (MT ci-après). Compiler ce code et **tester que les sorties sont conformes à ce qui est annoncé par l'auteur**.

2. Utiliser ce générateur pour produire un code source en C avec simplement une déclaration et initialisation de tableau de N nombres pseudo-aléatoires.

```
float tabMT[] = {
       premier nombre tiré,
       deuxième ombre,
       etc.,
       …
       } ;
```

3. Reprendre un code de simulation de Monte Carlo pour calculer PI (ZZ2) et le tester en faisant des appels au générateur MT en faisant 10 réplications avec 2 000 000 de tirages (1 000 000 de points).

4. Faire un code C qui compare, les performances en temps entre tirer des nombres pseudo-aléatoires avec MT et lire ces nombres dans le tableau généré. Reprendre le code du point 3 et intégrer le code généré au point 2, tester différentes tailles de tableau, observer les temps de compilation et comparer les 2 solutions. En fonction du nombre de tirages nécessaires, quelle technique proposeriez-vous.

5. Pour ceux qui s'ennuient : discuter de l'utilisation d'une technique à base de memory mapping, pour le contexte que nous venons de voir. Développer un code pour lire un fichier binaire de nombres générés avec MT avec la technique de memory mapping.

Deuxième partie : métamodèle du C++, introspection pour et transformation de modèle (par métaprogrammation) et génération de code C++ et Java

Pour chaque point ci-dessous faire un petit programme principal qui teste les classes. Prévoir les accesseurs – getters & setters ainsi que les méthodes que vous jugerez nécessaires. (Réutilisation de code de la 2ème année et utilisation simple de la STL)

1. En C++, concevoir une classe qui décrit les attributs d'une classe. On peut préciser simplement sous forme de chaîne de caractères :

   a. L'identificateur de l'attribut
   b. Son type (incluant ou non une * si c'est un pointeur)

2. Concevoir maintenant une classe qui décrit les méthodes d'une classe. Pour ce faire on doit pouvoir retrouver sous forme de chaînes de caractères le type de retour de la méthode, son identificateur et la liste des arguments (chaque argument étant précisé par son type)

3. Que proposez-vous pour dissocier les méthodes et attributs d'instances par rapport aux méthodes et attributs de classes.

4. Concevoir une classe C++ capable de décrire une classe C++.

5. Proposer pour ce type de classe un ensemble de méthodes qui permette de faire de l'instrospection (en clair afficher les informations que nous avons stockées mais en passant par des méthodes). On doit donc dans cette partie être capable de retrouver la liste des attributs et des méthodes (d'instance et de classe) et le nom de la classe elle même. Dans le programme principal, proposer une instance de cette classe.

6. Proposer une méthode qui grâce à ce métamodèle génère automatiquement le code C++ de déclaration d'une classe (en gros le fichier .hpp ou .hxx)

7. Reprendre l'étape 6 pour générer du code Java.