4ETI IMI, Examen [C++]
Programmation générique en C++
CPE Lyon

2013-2014 (2eme session)
durée 2h

Tous documents autorisés. Calculatrices interdites.

Répondez aux questions sur une copie séparée

Le sujet comporte 2 pages

En cas de doute sur la compréhension de l'énoncé, explicitez ce que vous comprenez et poursuivez l'exercice dans cette logique.

Note préliminaire:

- Toutes les questions concernent le language de programmation C++ dans le cadre du developpement de logiciels sur architecture PC standard.
- Nous vous invitons à commenter le code et les réponses. En particulier, en cas d'ambiguité de compréhension d'une question, ajoutez toutes remarques et illustrations supplémentaires permettant d'expliquer votre démarche.
- Sauf mention contraire explicite, on supposera que l'on dispose d'un système Linux standard fonctionnant correctement sur un PC récent 32 ou 64 bits (identiques aux conditions de TP des PC de CPE: *int* étant encodé sur 4 octets, pointeurs étant encodés respectivement sur 4(/8) octets sur 32(/64) bits).
- On supposera que le code C++ est compilé avec une version récente de g++ sous la norme C++11 (ou ultérieure) identique aux conditions de TP des PC de CPE (avec l'option -std=c++0x).
- On supposera dans chaque cas que les `#include` des en-tête standards nécessaires à la bonne compilation et exécution des programmes décrits sont correctement installés et appelés (ex. iostream, string, etc).

1 Structure de données C++

Un polynôme creux est un polynôme possédant beaucoup de coefficients valants 0. Il n'est alors pas intéressant de stocker l'ensemble des coefficients valant 0 et on préfère stocker uniquement les coefficients non nulles et leurs places dans le conteneur.
Par exemple $P(x) = 1.2\,x^{624} + 1.1\,x^{248} + 0.6\,x^{56} + 1$, est un polynome creux: 4 coefficients suffisent à définir ce polynôme $(1.2, 1.1, 0.6, 1)$. Notez qu'un polynome non creux nécessiterait de stocker au minimum 625 coefficients.

Nous souhaitons définir une structure de donnée permettant de stocker les valeurs inutiles. D'un autre coté, nous souhaitons pouvoir récupérer de manière efficace un coefficient donné du polynome. Pour obtenir ce fonctionnement, nous utilisons un conteneur de type `std::map`. La clé de la map sera l'exposant du monome, et la valeur sera son coefficient.

Nous souhaitons avoir les caractéristiques suivants:

- La type de x doit être un paramètre template de la classe du polynome.
- Ajout et suppression d'un monome donnée dans le polynome.
- Récupération du coefficient pour une puissance donnée.
- Évaluation de la valeur du polynome pour une valeur x donnée.
- Addition, soustraction entres polynomes.
- Multiplication entre deux polynomes.
- Récupération du polynome dérivé.

Question 1 *Ecrivez les en-têtes et les implémentations de cette structure de donnée en C++ satisfaisant à l'ensemble de ces contraintes.*