JUNIA ISEN

CIR 2

Partiel Informatique / POO C++

15/12/2022 / Sans Documents et Calculette

1. Questions de cours (8 points):

a) Considérant le code suivant, à l'aide de trois méthodes différentes, « fonction », « functor » et « lambda », définir la fonction « comp » afin d'effectuer un tri dans l'ordre inverse des valeurs absolues de « v ».

```
#include <vector>
#include <random>
int main() {
    std::default_random_engine rng(std::random_device{}());
    std::uniform_int_distribution<int> dist(-100, 100);  //(min, max)
    std::vector<int> v;
    v.reserve(100);
    for (int i = 0; i < 100; ++i) {
        v.push_back(dist(rng)); //génération aléatoire de valeurs
    }
    std::sort(v.begin(), v.end(), comp); //comp à compléter
    return EXIT_SUCCESS;
}
```

b) Ecrire une fonction *template* qui renvoie la moyenne des éléments d'un vecteur passé en paramètre.

c) Que doit respecter une classe (ex. Fraction) pour être compatible avec cette fonction ?

d) Donner un exemple d'un pointeur intelligent et citer au moins un avantage de son utilisation en remplacement d'un simple pointeur. Donner un exemple de son utilisation pour allouer un tableau de « n » flottants.

2. : Ecriture de code C++ (12 points)

a) Considérant la déclaration de la structure `Monomial` et de la classe `My_polynomial` ci-dessous permettant de modéliser respectivement, un monôme composé d'un coefficient et de son degré, et un polynôme tel que :

$$f = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$$

(Rappel : La fonction *pow* (fournie par *<cmath>*) permet de calculer x^y

```
double pow (double x, double y);)
```

```cpp
#include <vector>
#include <iostream>
#include <complex>
#include <string>

template <typename _Type>
struct Monomial
{
   size_t expo_; // Degré du monôme
   _Type coeff_; // Coefficient du monôme de type « _Type »
// ===> coeff_ * X^expo_
};

template <typename _Type>
class My_polynomial
{
private:
   std::vector<_Type> monomial_; // vecteur stockant les monômes de type « _Type » dans l'ordre croissant de leur degré associé (0..n)

public:
   size_t get_size(); // retourne la taille du vecteur monomial_

   My_polynomial();                                    // Constructeur 1
   My_polynomial(const My_polynomial<_Type> &other); // Constructeur 2
   My_polynomial(My_polynomial<_Type> &&other);      // Constructeur 3

   _Type &operator[](const size_t &ind); // retourne une référence sur le ième (ind) élément du vecteur monomial_

   _Type operator()(const _Type &val); // retourne la valeur du polynôme pour la valeur x = val

   My_polynomial<_Type> &operator+=(const Monomial<_Type> &m); // Addition d'un monôme avec un polynôme

   My_polynomial<_Type> operator+(const My_polynomial<_Type> &p); // Addition de deux polynômes
   My_polynomial<_Type> operator*(const My_polynomial<_Type> &p); // Multiplication de deux polynômes

   template <typename _Typel>
   friend std::ostream &operator<<(std::ostream &os, const My_polynomial<_Typel> &p); // Surcharge de l'opérateur <<

   template <typename _Typel>
   friend Monomial<_Typel> &operator>>(Monomial<_Typel> &m, My_polynomial<_Typel> &p); // Surcharge de l'opérateur >> attention il s'agit ici d'un monôme et pas de la classe
```

```
std::istream. A l'aide de cette méthode, le monôme est ajouté au polynôme
```

et considérant que la fonction « main » suivante :

```
int main()
{
   std::cout << "-------------------------------------------------------" << std::endl;
   My_polynomial<float> poly2;

   for (size_t i = 0; i < 3; ++i)
   {
       Monomial<float> m(i, static_cast<float>(i + 1));
       poly2 += m;
   }

   std::cout << "poly2" << poly2 << std::endl;
   std::cout << "-------------------------------------------------------" << std::endl;

   std::cout << "-------------------------------------------------------" << std::endl;
   My_polynomial<float> poly3;

   for (size_t i = 0; i < 3; ++i)
   {
       Monomial<float> m(i, static_cast<float>(i + 2));
       poly3 += m;
   }

   std::cout << "poly3" << poly3 << std::endl;
   std::cout << "-------------------------------------------------------" << std::endl;

   std::cout << "-------------------------------------------------------" << std::endl;
   auto poly4 = poly2 + poly3;

   std::cout << "poly4" << poly4 << std::endl;
   std::cout << "-------------------------------------------------------" << std::endl;

   std::cout << "-------------------------------------------------------" << std::endl;
   auto poly5 = poly2 * poly3;

   std::cout << "poly5" << poly5 << std::endl;

   std::cout << "-------------------------------------------------------" << std::endl;
   std::cout << "-------------------------------------------------------" << std::endl;
   std::cout << "poly3(2) = " << poly3(2) << std::endl;
   std::cout << "-------------------------------------------------------" << std::endl;

   std::cout << "-------------------------------------------------------" << std::endl;
```

```
    std::cout << "indice n°2 of poly3 = " << poly3[2] << std::endl;
    poly3[5] = 15;
    std::cout << "indice n°5 of poly3 = " << poly3[5] << std::endl;

    std::cout << "poly3" << poly3 << std::endl;
    std::cout << "-----------------------------------------------------" << std::endl;

    std::cout << "-----------------------------------------------------" << std::endl;
    My_polynomial<float> poly6;
    std::cout << "poly6" << poly6 << std::endl;

    Monomial<float> m0(0, 1);
    m0 >> poly6;
    Monomial<float> m1(1, 2);
    m1 >> poly6;
    Monomial<float> m5(3, 4);
    m5 >> poly6;

    std::cout << "poly6" << poly6 << std::endl;
    std::cout << "-----------------------------------------------------" << std::endl;

    return EXIT_SUCCESS;
}
```

Affiche:

```
---------------------------------------------------
poly2 = +1 X^0 +2 X^1 +3 X^2
---------------------------------------------------
---------------------------------------------------
poly3 = +2 X^0 +3 X^1 +4 X^2
---------------------------------------------------
---------------------------------------------------
poly4 = +3 X^0 +5 X^1 +7 X^2
---------------------------------------------------
---------------------------------------------------
poly5 = +2 X^0 +7 X^1 +16 X^2 +17 X^3 +12 X^4
---------------------------------------------------
---------------------------------------------------
poly3(2) = 24
---------------------------------------------------
---------------------------------------------------
indice n°2 of poly3 = 4
indice n°5 of poly3 = 15
poly3 = +2 X^0 +3 X^1 +4 X^2 +0 X^3 +0 X^4 +15 X^5
```

```
----------------------------------------------------
----------------------------------------------------
poly6 =
poly6 = +1 X^0 +2 X^1 +0 X^2 +4 X^3
----------------------------------------------------
```

a) Implémenter les trois constructeurs et la méthode « `size_t get_size() ».`
b) Implémenter la méthode permettant la surcharge de l'opérateur « += » (`My_polynomial<_Type> &operator+=(const Monomial<_Type> &m)).`
c) Implémenter la méthode permettant la surcharge de l'opérateur « + » (`My_polynomial<_Type> operator+(const My_polynomial<_Type> &p)`)
d) Implémenter la méthode permettant la surcharge de l'opérateur « * » (`My_polynomial<_Type> operator*(const My_polynomial<_Type> &p)`)
e) Implémenter les deux méthodes permettant la surcharge des opérateurs « >> » et « << » telles quelles sont définies ci-dessus.