Examen partiel 443 – Programmation orientée objet en C++

Université Paris Saclay – Master E3A

7 mars 2025

Les documents ne sont pas autorisés. Le barème donné est indicatif. Toute question peut être traitée en admettant les résultats précédents.

Exercice 1 — Questions de compréhension

1. Écrire la forme canonique de Coplien (FCC) de la classe A. Donnez des exemples des appels (à partir de la fonction `main` d'un programme) qui utilisent tout ce que vous avez implémenté pour la FCC.
2. Quelle sont les particularités d'un membre de classe statique ?
3. Quelle sont les particularités d'une méthode statique ? Donnez un exemple très simple pour illustrer l'intérêt de disposer d'une méthode statique.
4. Est-ce que cela a du sens de déclarer une méthode `static` et `const` à la fois ?
5. Expliquez brièvement en quoi consiste le problème du "diamant" qui peut apparaître suite à un héritage multiple.

Exercice 2 — Utilisation de base

1. En supposant qu'il n'y a pas de problème de dépassement de la capacité des types (donc vous pouvez utiliser int et float), ecrivez une fonction pour calculer, à partir de la valeur `int n` donnée en paramètre, la somme :

$$1 - \frac{1}{2!} + \frac{1}{3!} - \ldots + (-1)^{n+1}\frac{1}{n!}$$

2. Quel est le problème avec le code suivant, et qu'est-ce qu'on devrait changer ?

```
class A
{
public:
A() {}
~A(){}
};

class B: public A
{
public:
B():A(){}
~B(){}
};

int main(void)
{
  A* a = new B();
  delete a;
  return 0;
}
```

Exercice 3 — Utilisation de base

1. Quel est le problème avec le code suivant, et qu'est-ce qu'on devrait changer ?

```
class A
```

```
{
public:
A() {}
~A(){}
};

class B: public A
{
public:
B():A(){}
~B(){}
};

int main(void)
{
  A* a = new B();
  delete a;
  return 0;
}
```

Exercice 4 — Utilisation de base

1. Implémenter une fonction `void F(int* A, int* B, int N)` qui prend en paramètre deux tableaux A, B et leur taille N, et qui va écrire dans la case $B[i]$ le produit de toutes les valeurs présentes en A, sauf $A[i]$. Exemple : si A contient $A = \{2, 1, 5, 9\}$, on doit écrire en B les valeurs $B = \{45, 90, 18, 10\}$.
   Attention aux cas particuliers. Essayez (si vous avez le temps) de réfléchir à la façon la plus rapide d'effectuer les calculs nécessaires.

Exercice 5 — Utilisation de base

1. Expliquer ce qui se passe dans les deux programmes suivants. Est-ce que la sortie console sera la même dans les deux cas ?
   Programme 1 :

```
#include <iostream>
using namespace std;

int main()
{
  int count = 1;
  for (; count <= 5 ; count++)
  {
    int count = 1;
    cout << count << "\n";
  }
  return 0;
}
```

   Programme 2 :

```
#include <iostream>
using namespace std;

int main()
{
  int count = 1;
  while (count <= 5)
  {
    int count = 1;
    cout << count << "\n";
    count++;
  }
  return 0;
}
```

Exercice 6 — Design de classe (8 points)

Un polynôme à coefficients entiers est une fonction $P(x) = a_0 + a_1x^1 + \ldots + a_nx^n$, où $n \geq 0$ est le degré du polynôme, et les coefficients du polynôme $a_i \in \mathbb{Z}, \forall i \in \{0, \ldots, n\}$ sont tous des entiers (positifs ou négatifs).
L'objectif de l'exercice est d'implémenter une classe `Poly` qui modélise un polynôme et qui implémente quelques opérations de base.

1. Proposez et justifiez un choix pour les membres privés de la classe `Poly` qui nous permettraient de modéliser un polynôme quelconque.
2. Quel est le contenu des membres privés que vous avez proposés pour les polynômes suivants : $P_1(x) = 3$, $P_2(x) = 2x^2$, $P_3(x) = -2x^2 + x - 1$?
3. Ecrire la déclaration de la classe, et implémenter le constructeur sans paramètres de la classe qui initialise l'objet courant à un polynôme de votre choix ; justifiez ce choix.
4. Implémenter le constructeur

   ```
   Poly(int _deg, int* _coeffs)
   ```

   qui prend en paramètre le degré du polynôme, ainsi qu'un tableau d'entiers représentant les coefficients a_i, et qui initialise correctement les membres privés de l'objet. Grâce a ce constructeur, on pourra initialiser dans la fonction `main()` les objets de type `Poly` correspondant à $P_1(x), P_2(x), P_3(x)$ (question 2) de la manière suivante :

   ```
   int valeurs1[]={3};
   Poly p1(0,valeurs1);
   int valeurs2[]={0, 0, 2};
   Poly p2(3,valeurs2);
   int valeurs3[]={-1, 1, -2};
   Poly p3(2,valeurs3);
   ```

   Pendant la construction, on vérifie la condition suivante, sans laquelle le programme doit s'arrêter : la valeur du coefficient dominant a_n ne doit pas être zéro, sauf si le degré du polynôme est zéro (c'est à dire $P(x) = 0$ est valide).
5. Déclarer et implémenter le constructeur de copie.
6. Déclarer et implémenter l'opérateur d'affectation et le destructeur.
7. Surcharger l'opérateur `+` qui vous permettra d'obtenir la somme de deux polynômes. La surcharge de l'opérateur `+` nous permettra d'écrire dans la fonction `main()` :

   ```
   Poly sum;
   sum = p2 + p3;
   ```

   Attention : le degré du polynôme résultat doit être calculé en fonction de la même condition précédente sur le coefficient dominant. Quel est la valeur des membres privés de l'objet `sum` de dessus ?
8. Surcharger l'opérateur `*` qui vous permettra de multiplier le polynôme par **un entier** (on ne s'intéresse pas à la multiplication entre polynômes). La surcharge de l'opérateur `*` nous permettra d'écrire dans la fonction `main()` :

   ```
   Poly mul;
   mul = p2 * 3;
   ```

   Attention : le degré du polynôme résultat doit être calculé en fonction de la même condition précédente sur le coefficient dominant. Quel est la valeur des membres privés de l'objet `mul` de dessus ?
9. Si dans la fonction `main()` on écrit le code suivant :

   ```
   Poly mul;
   mul = 3 * p2;
   ```

   justifiez pour quelle raison cela renvoie une erreur de compilation. Proposez une solution pour que ce code fonctionne avec le même résultat que celui de la question précédente.
10. Surcharger l'opérateur `-` qui vous permettra d'obtenir la différence de deux polynômes. La surcharge doit être implémentée uniquement à partir des autres opérateurs déjà surchargés. La surcharge de l'opérateur `-` nous permettra d'écrire dans la fonction `main()` :

    ```
    Poly dif;
    dif = p2 - p3;
    ```

11. Surcharger l'opérateur `<<` qui vous permettra d'insérer dans le flot de sortie des objets comme `p1, p2, p3` créés précédemment :

```
std::cout << p1 << std::endl;
std::cout << p2 << std::endl;
std::cout << p3 << std::endl;
```

Suite aux instructions précédentes, on devrait voir apparaître dans la console :

```
3
2x^2
-2x^2+x-1
```

Attention : pour avoir un affichage "propre" comme celui de l'exemple, il y a beaucoup de cas particuliers qui doivent être pris en compte. Plutôt que d'écrire beaucoup de code incorrect, essayez d'être succincts et très clairs (et commentez votre code) pour avoir des points pour les cas que vous prenez en compte correctement, même s'il vous reste des cas non-traités.

12. On s'intéresse maintenant à l'évaluation de $P(x)$ pour une valeur donnée de x. En supposant que vous avez accès à une fonction $pow(x, n)$ qui calcule x^n avec une implémentation naïve (c.a.d. $n-1$ multiplications), combien d'additions et de multiplications sont nécessaires pour évaluer $P(x)$ pour une valeur donnée de x ?
13. Si on écrit $P(x)$ de la manière suivante :

$$P(x) = a_0 + x\Big(a_1 + x\big(a_2 + \ldots + x(a_{n-1} + xa_n)\big)\Big)$$

proposez une stratégie pour évaluer $P(x)$ en moins d'opérations. Combien d'additions et de multiplications sont nécessaires maintenant ?
14. Implémentez une méthode `int eval(int v)` qui evalue $P(x)$ en $x = v$.