Université de Nice-Sophia Antipolis POLYTECH
ELEC4 2019–2020

Examen de Programmation C++

Durée : 1h
Aucun document autorisé
Mobiles interdits Nom : Prénom :

---

1 Elève

On souhaite modéliser les élèves d'une classe d'une école. Écrivez en C++ la classe `Eleve` qui possède :

- la variable privée `nom` de type `string` ;
- la variable privée `notes` de type `vector` de **double** ;
- un/des constructeur(s) pour initialiser les deux variables privées ;
- deux accesseurs `getNom` et `getNotes` ;
- la méthode `moyenne` qui renvoie la moyenne des notes de l'élève (-1 si pas de notes) ;
- la surcharge de l'opérateur `<<` pour écrire le nom de l'élève avec toutes ses notes (ou *pas de notes*) sur un `ostream`.

```
#pragma once

#include <vector>
#include <string>
#include <iostream>

class Eleve {
private:
  std::string nom;
  std::vector<double> notes;
public:
  // constructeur
  Eleve(const std::string s,
        const std::vector<double> &n = {}) : nom(s), notes(n) {}
  // accesseurs
  const  std::string &getNom() const {
    return this->nom;
  }
  const std::vector<double>&getNotes() const {
    return this->notes;
  }

  double moyenne() const {
```



```
    if (this->notes.empty())
      // pas de notes
      return -1;
    // calculer la moyenne
    double moy = 0.0;
    for (double n : this->notes)
      moy+=n;
    return moy/this->notes.size();
  }
  //
  const std::string toString() const {
    std::string s = this->nom + " :";
    if (this->notes.empty())
      s+=" pas de notes";
    else
      for (double n : this->notes)
        s+=" " + std::to_string(n);
    return s;
  }
  friend std::ostream&operator<<(std::ostream &f, const Eleve& e) {
    return f << e.toString();
  }
};
```

.....

▶ 1. Écrivez la méthode `main` qui déclare deux élèves *Paul* et *Isabelle*. Le premier n'a pas de notes, et la seconde a les 4 notes suivantes : 12.0, 9.0, 13.5 et 19.0. Vous écrirez sur la sortie standard les deux élèves avec leur moyenne. Votre programme écrira donc :

```
Paul :  : pas de notes
Isabelle :  12.000000 9.000000 13.500000 19.000000 : Moy = 13.375
```

```
void printlnEleve(const Eleve &e) {
  // afficher le nom de l'élève avec ses notes
  std::cout << e;
  // afficher sa moyenne s'il a des notes
  double m = e.moyenne();
  if (m!=-1)
    std::cout << " : Moy = " << std::to_string(m);
  //
  std::cout << std::endl;
}

int main() {
  Eleve e1("Paul"), e2("Isabelle", {12, 9, 13.5, 19.0} );
  printlnEleve(e1);
  printlnEleve(e2);
  return EXIT_SUCCESS;
}
```

2 Classe

Maintenant, on modélise une classe d'élèves. Écrivez la classe `Classe` (🙂) qui possède :

- la variable privée `nom` de type `string` ;
- la variable privée `lesEleves` de type `vector` de pointeur sur `Eleve` ;
- un constructeur pour initialiser la variable privée `nom` ;
- le constructeur de copie ;
- le destructeur ;
- l'accesseur `getnom` ;
- la méthode `addEleve` pour ajouter un élève à la fin du vecteur `lesEleves` ;
- la surcharge de l'opérateur `<<` pour écrire le nom de la classe avec les noms de tous les élèves et leur moyennes sur un `ostream` ;
- la surcharge de l'opérateur d'affectation `=`.

```
#include "Eleve.hpp"
class Classe {
private:
  std::string nom;
  std::vector<Eleve *> lesEleves;

  void dupliquer(const  Classe &c) {
    this->nom = c.nom;
    for (Eleve *e : c.lesEleves)
      this->lesEleves.push_back(new Eleve(*e));
  }

public:
  // constructeurs
  Classe(const std::string n) : nom(n) {}
  Classe(const Classe &c) { dupliquer(c); }
  // destructeur
  ~Classe() {
    for (Eleve *e : this->lesEleves) delete e;
  }
  // accesseurs
  const std::string &getNom() const { return this->nom; }
  //
  void addEleve(Eleve *e) {
    this->lesEleves.push_back(e);
  }
  //
  const std::string toString() const {
    std::string s ="-- Classe " + this->nom + " --";
    for (Eleve *e : this->lesEleves) {
      s+="\n" + e->getNom() + " : ";
      double m = e->moyenne();
      s+= (m==-1) ? " abs" : std::to_string(m);
```



```
    }
    return s;
  }

  Classe &operator=(const Classe &c) {
      dupliquer(c);
      return *this;
  }
  friend std::ostream&operator<<(std::ostream &f, const Classe& c) {
    return f << c.toString();
  }
};
```

▶ 2. Écrivez la méthode `main` qui déclare la variable `elec4` de type `Classe`. Dans cette classe, vous ajouterez les deux élèves *Ali* et *Mei*. Le premier n'a pas de notes, et la seconde a 3 notes : 15.5, 14.0 et 9.0. Puis, écrivez la variable `elec4` sur la sortie standard, ce qui produira :

```
-- Classe Elec4 --
Ali : abs
Mei : 12.833333
```

```
int main() {
  Classe elec4("Elec4");
  elec4.addEleve(new Eleve("Ali"));
  elec4.addEleve(new Eleve("Mei", {15.5, 14.0, 9.0} ));
  std::cout << elec4 << std::endl;
  return EXIT_SUCCESS;
}
```