Université de Nice-Sophia Antipolis — POLYTECH
ELEC4 — 2022–2023

Examen de C++

Durée : 1h00
Aucun document autorisé
Mobiles interdits

On souhaite représenter l'ensemble $\mathbb{Q}$ des nombres rationnels. On rappelle que : $\mathbb{Q} = \{\frac{m}{n} | (m,n) \in \mathbb{Z} \times \mathbb{Z}^*\}$.

► 1. Écrivez la classe `rationnel` qui possède :

- 2 variables membres privées `num` et `dem` ;
- des constructeurs ;
- le destructeur (qui ne fera rien) ;
- les accesseurs (*get...* et *set...*) ;
- la méthode `toString` qui renvoie une chaîne de caractères représentation du rationnel courant au format *num/dem* ;
- la surcharge de l'opérateur `<<` pour écrire un rationnel dans un `std::ostream`.
- la surcharge de l'opérateur `>>` pour lire un rationnel depuis un `std::ifstream`.

```
#pragma once
/*
 * la classe rationnel représente l'ensemble mathématique ℚ
 * des nombres rationnels tels que ℚ = {m/n | (m,n) ∈ ℤ × ℤ*}
 */
#include <iostream>
#include <cassert>

class rationnel {
private:
  int num, dem;
public:
  //
  static const rationnel zero; // (0,1)
  // les constructeurs
  rationnel(const int n=0, const int d=1) : num(n), dem(d) { assert(d!=0); }
  // le destructeur
  ~rationnel() {}
  // les accesseurs
  int getNumerateur() const { return this->num; }
  int getDenominateur() const { return this->dem; }
  void setNumerateur(const int n) { this->num = n; }
  void setDenominateur(const int d) { assert(d!=0); this->dem = d; }
  // la méthode toString renvoie une chaîne de caractères représentation
  // du rationnel courant au format num/dem ;
  std::string toString() const {
    return std::to_string(this->num) + "/" + std::to_string(this->dem);
  }
  // surcharge de l'opérateur « pour écrire un rationnel dans un std::ostream.
  friend std::ostream& operator<<(std::ostream &f, const rationnel &r) {
```



```
    return f << r.toString();
  }
  // surcharge de l'opérateur » pour lire un rationnel depuis un std::ifstream.
  friend std::ifstream& operator>>(std::ifstream &f, const rationnel &r) {
    f.read((char *) &r, sizeof(rationnel));
    return f;
  }
};
```

.....

► 2. Complétez la classe `rationnel` en écrivant la méthode `simplifier` qui renvoie (si possible) le rationnel simplifié du rationnel courant. Par exemple, 21/70 est simplifié en 3/10 et 42/6 en 7/1. Vous pourrez écrire des méthodes auxiliaires.

```
// rôle : renvoie le rationnel simplifié du rationnel courant
rationnel simplifier() const {
  if (this->num == 0) return rationnel::zero;
  if (this->num == this->dem) return rationnel(1,1);
  if (this->num == -this->dem) return rationnel(-1,1);
  // chercher les diviseurs et réduire
  return this->reduire();
}

// rôle : renvoie la réduction du rationnel courant à partir de ses diviseurs
rationnel rationnel::reduire() const
{
  // calculer le pgcd
  int div = rationnel::pgcd(std::abs(this->num), std::abs(this->dem));
  return rationnel(this->num/div, this->dem/div);
}

// rappel du calcul du pgcd
int pgcd(int a, int b) {
  while (a != b) {
    // pgcd(a,b) = pgcd(a-b,b) et a>b
    // = pgcd(a,b-a) et a<b
    if (a>b)
      // pgcd(a,b) = pgcd(a-b,b) et a>b
      a -= b;
    else
      b -=a;
    // pgcd(a,b) = pgcd(a,b-a) et a<b
  }
  return a;
}
```

.....

► 3. Écrivez la méthode `plus` qui renvoie la somme du rationnel courant et du rationnel passé en paramètre.

```
/*
 * Rôle : renvoie la somme *this + r
```

```
 */
rationnel plus(const rationnel &r) const {
  if (*this==rationnel::zero) return r;
  if (r==rationnel::zero) return *this;
  //
  int new_num, new_dem;
  if (this->dem == r.dem) {
    // même dénominateur
    new_num = this->num + r.num;
    new_dem = r.dem;
  } else {
    new_num = this->num*r.dem + r.num*this->dem;
    new_dem = this->dem*r.dem;
  }
  return rationnel(new_num, new_dem).simplifier();
}
```

.....

▶ 4. Écrivez la surcharge de l'opérateur `+` qui renvoie la somme du rationnel courant et du rationnel passé en paramètre.

```
/*
 * Rôle : renvoie la somme *this + r
 */
rationnel operator+(const rationnel &r) const {
  return this->plus(r);
}
```

.....

▶ 5. Un fichier de nom *fr* contient des nombres rationnels (de type `rationnel`, bien sûr). Écrivez la fonction `main` qui lit tous les rationnels contenus dans le fichier *fr* et qui écrit sur la sortie standard leur somme.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include "rationnel.hpp"

int main() {
  std::ifstream ios("fr");
  if (!ios.is_open()) {
    perror("fr");
    exit(EXIT_FAILURE);
  }
  // lire le fichier fr de rationnels et calculer leur somme
  rationnel r, somme=rationnel::zero;
  while (ios >> r)
    somme = somme + r;
  //
  ios.close();
  std::cout << somme << std::endl;
  return EXIT_SUCCESS;
```



```
}
```

.....