Auteur : Marco Lavoie
Instructeur: Sébastien Bois

Devoir No.7

L'instructeur a présenté en classe des bonnes pratiques de programmation en C++ que des conventions d'écriture. Assurez-vous d'appliquer des pratiques et conventions pour ne pas voir votre travail pénalisé inutilement.

À effectuer

Créez une classe appelée **Rationnel** pour effectuer des opérations arithmétiques avec des fractions. L'instructeur vous fournie un programme principal pour tester votre classe.

Rehaussez la solution de l'exercice 8.3, portant sur la classe **Rationnel**, en surchargeant les opérateurs suivants dans la classe :

- Les opérateurs arithmétiques (**+**, **-**, ***** et **/**) pour fractions et entiers (ex: f3 = f1 + 6 + f2).
- Les opérateurs relationnels (**==**, **!=**, **<**, **<=**, **>** et **>=**) pour fractions et entiers (ex: f2 < f1 && f3 >= 7 || 8 < f4).
- Les opérateurs d'affectation (**=**, **+=**, **-=**, ***=** et **/=**) pour fractions et entiers (ex: f2 /= f1).
- Les opérateurs d'incrémentation (**++**) et décrémentation (**--**) en format préfixe (ex: --f1) et suffixe (ex: f2++).

Voici un programme principal testant tous les opérateurs que vous devez surcharger dans la classe **Rationnel** ; ce programme affiche « Ok » lorsque l'opérateur réussi le test, « Fail » sinon :

```
// Programme principal : teste la classe Rationnel
int _tmain(int argc, _TCHAR* argv[]) {
    Rationnel r1( 1, 2 ),
              r2( 3, 4 ),
              r3( 5, 6 ),
              r4;

    // Tester l'addition
    cout << "\nAddition :       ";
    r4 = r1 + r2 + r3;  cout << ( r4 == Rationnel( 25, 12 ) ? "Ok " : "Fail " );
    r4 = 1 + r1;        cout << ( r4 == Rationnel(  3,  2 ) ? "Ok " : "Fail " );
    r4 = r1 + 1;        cout << ( r4 == Rationnel(  3,  2 ) ? "Ok " : "Fail " );

    // Tester la soustration
    cout << "\nSoustraction :   ";
    r4 = r3 - r2 - r1;  cout << ( r4 == Rationnel( -5, 12 ) ? "Ok " : "Fail " );
    r4 = 1 - r1;        cout << ( r4 == Rationnel(  1,  2 ) ? "Ok " : "Fail " );
    r4 = r1 - 1;        cout << ( r4 == Rationnel( -1,  2 ) ? "Ok " : "Fail " );

    // Tester la multiplication
    cout << "\nMultiplication : ";
    r4 = r3 * r2 * r1;  cout << ( r4 == Rationnel(  5, 16 ) ? "Ok " : "Fail " );
    r4 = 2 * r1;        cout << ( r4 == Rationnel(  1,  1 ) ? "Ok " : "Fail " );
    r4 = r1 * 2;        cout << ( r4 == Rationnel(  2,  2 ) ? "Ok " : "Fail " );

    // Tester la division
```

```
cout << "\nDivision :       ";
r4 = r3 / r2 / r1;  cout << ( r4 == Rationnel( 40, 18 ) ? "Ok " : "Fail " );
r4 = 2 / r1;        cout << ( r4 == Rationnel(  4,  1 ) ? "Ok " : "Fail " );
r4 = r1 / 2;        cout << ( r4 == Rationnel(  1,  4 ) ? "Ok " : "Fail " );

// Opérateurs relationnels
cout << "\n\n== : ";
cout << ( Rationnel( 1, 2 ) == Rationnel( 1, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 2, 2 ) == 1 ? "Ok " : "Fail " );
cout << ( 2 == Rationnel( 4, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) == Rationnel( 3, 2 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 2, 2 ) == 2 ? "Fail " : "Ok " );
cout << ( 2 == Rationnel( 5, 2 ) ? "Fail " : "Ok " );

cout << "\n!= : ";
cout << ( Rationnel( 1, 3 ) != Rationnel( 1, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) != 1 ? "Ok " : "Fail " );
cout << ( 2 != Rationnel( 5, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 3 ) != Rationnel( 1, 3 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 2, 2 ) != 1 ? "Fail " : "Ok " );
cout << ( 2 != Rationnel( 4, 2 ) ? "Fail " : "Ok " );

cout << "\n< :  ";
cout << ( Rationnel( 1, 3 ) < Rationnel( 1, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) < 1 ? "Ok " : "Fail " );
cout << ( 2 < Rationnel( 5, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) < Rationnel( 1, 2 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 3, 2 ) < 1 ? "Fail " : "Ok " );
cout << ( 2 < Rationnel( 1, 2 ) ? "Fail " : "Ok " );

cout << "\n<= : ";
cout << ( Rationnel( 1, 3 ) <= Rationnel( 1, 3 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) <= 1 ? "Ok " : "Fail " );
cout << ( 2 <= Rationnel( 4, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) <= Rationnel( 1, 3 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 3, 2 ) <= 1 ? "Fail " : "Ok " );
cout << ( 2 <= Rationnel( 1, 2 ) ? "Fail " : "Ok " );

cout << "\n> :  ";
cout << ( Rationnel( 1, 2 ) > Rationnel( 1, 3 ) ? "Ok " : "Fail " );
cout << ( 1 > Rationnel( 1, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 5, 2 ) > 2 ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 2 ) > Rationnel( 1, 2 ) ? "Fail " : "Ok " );
cout << ( 1 > Rationnel( 3, 2 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 1, 2 ) > 2 ? "Fail " : "Ok " );

cout << "\n>= : ";
cout << ( Rationnel( 1, 3 ) >= Rationnel( 1, 3 ) ? "Ok " : "Fail " );
cout << ( 1 >= Rationnel( 1, 2 ) ? "Ok " : "Fail " );
cout << ( Rationnel( 4, 2 ) >= 2 ? "Ok " : "Fail " );
cout << ( Rationnel( 1, 3 ) >= Rationnel( 1, 2 ) ? "Fail " : "Ok " );
cout << ( 1 >= Rationnel( 3, 2 ) ? "Fail " : "Ok " );
cout << ( Rationnel( 1, 2 ) >= 2 ? "Fail " : "Ok " );

// Opérateurs d'affectation
cout << "\n\n= :  ";
r4 = Rationnel( 1, 2 );  cout << ( r4 == Rationnel( 1, 2 ) ? "Ok " : "Fail " );
r4 = 2;                  cout << ( r4 == Rationnel( 4, 2 ) ? "Ok " : "Fail " );

cout << "\n+= : ";
r4 += Rationnel( 1, 2 ); cout << ( r4 == Rationnel( 5, 2 ) ? "Ok " : "Fail " );
r4 += 2;                 cout << ( r4 == Rationnel( 9, 2 ) ? "Ok " : "Fail " );

cout << "\n-= : ";
r4 -= Rationnel( 1, 2 ); cout << ( r4 == Rationnel( 8, 2 ) ? "Ok " : "Fail " );
r4 -= 2;                 cout << ( r4 == Rationnel( 4, 2 ) ? "Ok " : "Fail " );

cout << "\n*= : ";
r4 *= Rationnel( 1, 2 ); cout << ( r4 == Rationnel( 2, 2 ) ? "Ok " : "Fail " );
r4 *= 2;                 cout << ( r4 == Rationnel( 4, 2 ) ? "Ok " : "Fail " );

cout << "\n/= : ";
r4 /= Rationnel( 1, 2 ); cout << ( r4 == Rationnel( 8, 2 ) ? "Ok " : "Fail " );
r4 /= 2;                 cout << ( r4 == Rationnel( 4, 2 ) ? "Ok " : "Fail " );

// Opérateurs d'incrémentation et décrémentation
cout << "\n\n++ :  ";
r4 = Rationnel( 1, 2 );
cout << ( r4++ == Rationnel( 1, 2 ) && r4 == Rationnel( 3, 2 ) ? "Ok " : "Fail " );
r4 = Rationnel( 1, 2 );
cout << ( ++r4 == Rationnel( 3, 2 ) && r4 == Rationnel( 3, 2 ) ? "Ok " : "Fail " );
```

```
    cout << "\n-- :  ";
    r4 = Rationnel( 3, 2 );
    cout << ( r4-- == Rationnel( 3, 2 ) && r4 == Rationnel( 1, 2 ) ? "Ok " : "Fail " );
    r4 = Rationnel( 3, 2 );
    cout << ( --r4 == Rationnel( 1, 2 ) && r4 == Rationnel( 1, 2 ) ? "Ok " : "Fail " );

    // Attendre confirmation pour fermer la console
    std::cout << "\n\nPressez une touche pour terminer..." << std::endl;
    _getch();

    return 0;
}
```

Source: Marco Lavoie