TP4 - Surcharge des opérateurs

Rappel : **avant** d'écrire un sous-programme ou une méthode, il est conseillé de définir son jeu de tests et de préparer les tests automatiques (pour les fonctions/méthodes ne faisant pas d'entrée/sortie) dans le fichier `main.cpp`.

Vous devez utiliser la décomposition en 3 fichiers.

Introduction

Vous disposez d'une classe `Rationnel` permettant de représenter des fractions **sous forme irréductible**.

Les fractions sont représentées par leur numérateur `_num` et leur dénominateur `_den`. Pour obtenir la fome irréductible, une méthode privée `_reduire` vous est fournie. Cette méthode devra être appelée à chaque fois que nécessaire (mais pas plus) afin de toujours produire des rationnels réduits. Comme elle est privée, elle ne peut être appelée directement que dans la classe : en dehors, **elle peut être appelée indirectement au travers du constructeur**.

On vous demande d'implanter les fonctionalités suivantes et de les tester au fur et à mesure. Selon les cas, il pourra s'agir de méthodes de la classe `Rationnel` ou de fonctions externes (à vous d'identifier les cas). Vous pourrez être amenés à devoir ajouter des méthodes supplémentaires non explicitement demandées. Tous les calculs doivent être exacts : **il ne faut donc jamais convertir un objet de type `Rationnel` en flottant**.

Partie 1

1. Compilez et exécutez le code ; regardez les tests qui sont effectués.
2. Définissez l'opérateur `*` qui calcule le produit de deux rationnels. Testez.
3. Définissez l'opérateur de comparaison `==`. Testez.
4. Définissez l'opérateur `<<` pour l'affichage. Remplacez tous les appels à la méthode `affiche` par des utilisation de cet opérateur.
5. Définissez l'opérateur `-` **unaire** (pour réaliser l'instruction `r1 = -r2` où `r1` et `r2` sont des objets de la classe `Rationnel`). Testez.
6. Définissez l'opérateur `+` qui calcule la somme de deux rationnels. Testez.
7. Définissez l'opérateur de comparaison `<`. Testez. (Rappel : ne pas convertir en flottant.)

Partie 2

1. Ajouter la fonction membre `saisir` prenant en paramètre un flot d'entrée.

```
void Rationnel::saisir(istream & in){
  cout << "numerateur ?";
  in>> _num;
  do{
   cout << "denominateur ?";
   in >> _den;
  }while(_den == 0);
  if (_den < 0) {
    _num = -_num;
    _den = -_den;
  }
  _reduire();
}
```

   Utiliser la fonction membre `saisir` pour redéfinir l'opérateur `>>` pour la saisie. Testez.
2. Ajoutez au programme principal le calcul `r2 = r1 * 3` (avec `r1` et `r2` objets de la classe `Rationnel`). Cette instruction compile-t-elle ? Donne-t-elle le bon résultat ? Pourquoi ?
3. Ajoutez au programme principal le calcul `r2 = 3 * r1`. Cette instruction compile-t-elle ? Donne-t-elle le bon résultat ? Pourquoi ?
4. Écrivez si besoin le prototype, puis le corps des fonctions nécessaires à la bonne exécution des instructions des questions précédentes.
5. Testez votre programme en calculant le produit et le maximum d'un vecteur de rationnels.

Partie 3

1. Définissez l'opérateur `!=` en utilisant les opérateurs déjà définis. Testez.
2. Définissez l'opérateur `<=` en utilisant les opérateurs déjà définis. Testez.
3. Définissez l'opérateur `++` pour pouvoir écrire `r++` Testez.
4. Définissez l'opérateur `++` pour pouvoir écrire `++r`Testez.

?

Modifié le: jeudi 15 décembre 2022, 12:16