Nombres rationnels

Exercice 1 Le but de cet exercice est de simuler les nombres rationnels. Ils sont caractérisés par un numérateur et un dénominateur, tous deux étant des nombres entiers.

1. Ecrire une classe nommée **Rat** contenant les deux membres **num_** et **den_** dans le fichier d'en-tête **rational.h**.
2. Ecrire dans le fichier source **rational.cpp** les méthodes suivantes :
   (a) Le constructeur. Il faudra faire en sorte que si le numérateur est nul, le rationnel 0/1 soit renvoyé, le dénominateur doit être toujours positif, et il faudra gérer le cas où le rationnel n'est pas sous forme irréductible.
   (b) Les accesseurs **get_num()**et **get_den()** pour récupérer les membres de donnée de la classe.
   (c) Les setters **set_num()** et **set_den()** pour définir les membres de donnée de la classe. On ne mettra pas le rationnel sous forme irréductible.
   (d) Le setter **set()** pour définir d'un coup les deux membres de donnée de la classe. On mettra le rationnel sous forme irréductible.
   (e) La surcharge des opérateurs internes suivants :
      i. **operator+=** pour un rationnel et un entier
      ii. **operator-=** pour un rationnel et un entier
      iii. **operator*=** pour un rationnel et un entier
      iv. **operator/=** pour un rationnel et un entier
   (f) La surcharge des opérateurs externes suivants :
      i. **operator<<** pour l'affichage (le rationnel n/1 aura pour affichage n seulement).
      ii. **operator+**
      iii. **operator-**
      iv. **operator***
      v. **operator/**
      vi. **operator==**
      vii. **operator !=**
      viii. **operator<**
      ix. **operator<=**
      x. **operator>**
      xi. **operator>=**
3. Ecrire dans le fichier source **main.cpp** un exemple d'utilisation des rationnels.

Dans tous les cas, utiliser le namespace **ensiie** ainsi que les exceptions quand elles sont nécessaires.

Vecteur

Exercice 2 Le but de cet exercice est de simuler des vecteurs de taille quelconque. On utilisera les pointeurs pour les données. Ils sont donc caractérisés par un pointeur et une taille (le nombre d'éléments).

1. Ecrire une classe nommée **Vector** contenant les deux membres **data_** et **size_** dans le fichier d'en-tête **vector.h**.
2. Ecrire dans le fichier source **vector.cpp** les méthodes suivantes :
   (a) Le constructeur, qui prendra la taille du vecteur en argument.
   (b) Le destructeur.
   (c) Le constructeur de copie.
   (d) **get_size()** qui retourne le nombre déléments du vecteur.
   (e) La surcharge des opérateurs internes suivants :
      i. **operator[ ]** pour récupérer un coefficient.
      ii. **operator[ ]** pour définir la valeur d'un coefficient.
   (f) La surcharge des opérateurs externes suivants :
      i. **operator<<** pour l'affichage sous la forme "(v1,...,vn)".
      ii. **operator+**
      iii. **operator-**
      iv. **operator*** pour le produit scalaire de deux vecteurs
      v. **operator*** pour le produit d'un vecteur par un scalaire (et inversement)
      vi. **operator/** pour la division d'un vecteur par un scalaire
   (g) **norm()** pour la norme (inclure le fichier **cmath** pour utiliser la fonction **sqrt()**). On passera un argument pour la norme $p \geqslant 1$, aucun pour la norme 2 (on pourra utiliser la fonction **pow()**).
   (h) **cross()** pour le produit scalaire de deux vecteurs de dimension 3.
3. Ecrire dans le fichier source **main.cpp** un exemple d'utilisation des vecteurs.

Dans tous les cas, utiliser le namespace **ensiie** ainsi que les exceptions quand elles sont nécessaires.