Opérations sur les rationnels

2 séances de TP: 1re et 2e

Le TP peut être préparé seul ou en binôme. Le rendu doit consister en un fichier archive `<Nom1>_<Nom2>.tar.gz` ou `<Nom1>_<Nom2>.zip` contenant :

- votre code C++, commenté et facilement compilable (commande `make`)
- si la commande `make` ne suffit pas à compiler votre code, ajoutez un fichier README expliquant précisément comment compiler votre code.
- (optionnel) un compte-rendu de TP répondant aux questions qui ne demandent pas un programme et tout autre explication complémentaire.

Le TP doit être déposé sur moodle.
La date limite de rendu est la veille de la 3e séance de TP, à 23h59.

Livrable 1. Pour commencer

a. Définir une classe `Rationnel` qui comportera deux champs entiers `numerateur` et `denominateur`, un constructeur le `Rationnel(int,int)` et une méthode `afficher()` qui affiche le rationnel sous la forme p/q.

b. Implémenter et tester (progressivement) les méthodes suivantes :

- `valeur_approx()` : retourne l'approximation flottante du rationnel sous la forme d'un `double`.
- `simplifie()` : simplifie la fraction
- `ajouter(int n)` (resp. `ajouter(const Rationnel& r)`) : ajoute `n` (resp. `r`) à la valeur courante.
- Idem pour `soustraire`, `multiplier`, `diviser`
- Un test d'égalité (pour commencer : une méthode `egale`, avant d'essayer d'écrire un `operator==`)

Livrable 2. Précision des calculs numériques

On veut utiliser la classe rationnel ci dessus pour étudier le comportement des suites :

$$u_n = \begin{cases} u_0 = \frac{3}{2} \\ u_1 = \frac{5}{3} \\ u_{n+2} = 2003 - \frac{6002}{u_{n+1}} + \frac{4000}{u_n . u_{n+1}} \end{cases}, \quad v_n = \begin{cases} v_0 = 1 \\ v_{n+1} = \frac{v_n}{2} + \frac{a}{2v_n} \end{cases}$$

a. Ecrire un programme qui calcule ces suites à l'aide de `double`. Quel est le coût de vos algorithmes ?

b. Calculer par exemple u_{42} et v_{42} pour $a = 2$.

c. Ecrire un programme qui calcule ces suites à l'aide de votre classe `Rationnel` et comparer les résultats pour des valeurs de n jusqu'à 15.

d. (Bonus) Vers quelles valeurs converge v_n pour $a = 2, 4, 9, 25$? Retrouver d'où vient la formule définissant v_n (indication : c'est la méthode de la tangente de Newton)

Pour la vérification de ce livrable 2 :

u_n (en double) converge vers 2000
u_n (en rationnel) converge vers 2
v_n (en double) converge vers 1.4142
v_n (en rationnel) cesse de fonctionner correctement (overflow) quand n > 4

Livrable 3. Améliorons nos rationnels

La taille des entiers est limitée dans un ordinateur, ce qui pose rapidement des problèmes dès que le dénominateur ou le numérateur des rationnels est un peu grand. Pour cela, on va ré-écrire une classe `Entier` pour représenter des entiers non bornés.

On représente un entier en base b sous forme d'un tableau de nombres compris entre 0 et $b-1$. Dans un premier temps on pourra se contenter d'un tableau de taille fixée.

a. Quel b va-t-on choisir et quel type pour les éléments du tableau ?

b. Ecrire une classe implémentant les opérations usuelles sur les entiers (`addition`, `multiplication`...) ainsi qu'une fonction pour afficher les résultats.

c. Quel est le coût de vos algorithmes, en précisant le modèle de coût ?

d. Tester votre classe, par exemple en écrivant un programme calculant la fonction factorielle.

e. Remplacer les `int` par votre classe `Entier` dans votre classe `Rationnel`.

f. Comparer les résultats pour le calcul de u_n et v_n.

Livrable 4. (bonus) Pour aller plus loin

a. Dans l'implémentation des entiers, on s'est contenté de tableaux de taille fixe. Comment peut-on améliorer la chose ?

b. Réfléchir aux différentes optimisations que l'on pourrait faire pour améliorer la vitesse de vos algorithmes : accélérer les calculs quand les entiers sont petits (ex : $< 2^{30}$), quand les entiers changent de taille, algorithmes de multiplication...