IN 101 - TD 08
4 novembre 2011

Énoncé

ENSTA ParisTech

Dans ce TD nous allons commencer par programmer une bibliothèque de gestion de nombre rationnels et ensuite l'utiliser pour calculer des racines carrées.

Exercice 1 : Bibliothèque pour gérer les rationnels

Dans cet exercice, vous devez écrire 3 fichiers différents : un fichier `ratio.c` contenant le code des fonctions pour manipuler les rationnels, un fichier `ratio.h` contenant les prototypes des fonctions utilisables à l'extérieur de la bibliothèque, et un fichier `main.c` (ou un nom qui vous arrange) qui contiendra juste une fonction `main` pour tester/utiliser votre bibliothèque de rationnels.

1.a] Commencez par définir un type "rationnel" à l'aide d'une structure C adaptée. Dans quel fichier la structure (et éventuellement un `typedef`) doit-elle être définie ? Écrivez une fonction d'initialisation d'un rationnel qui prend en argument deux entiers a et b et retourne un pointeur vers une structure de rationnel représentant $\frac{a}{b}$. Réfléchissez au fichier dans lequel doit aller la fonction et où doit aller son prototype. Pensez à allouer la mémoire correctement pour pouvoir réutiliser le rationnel dans une autre fonction.

1.b] Écrivez une fonction prenant en argument un pointeur vers un rationnel et qui affiche ce rationnel. Écrivez une petite fonction `main` pour tester vos fonctions d'initialisation et d'affichage. Pensez bien à inclure tous les fichiers nécessaires lors de la compilation.

1.c] Écrivez maintenant deux autres fonctions permettant d'additionner et de multiplier deux rationnels (sans les simplifier pour l'instant) et une autre fonction permettant de désallouer un rationnel. Utilisez ces nouvelles fonctions dans le `main` et vérifiez qu'elles fonctionnent bien.

1.d] On veut maintenant écrire une fonction de simplification d'un nombre rationnel. Pour cela il faut être capable de calculer un pgcd. Écrivez donc une fonction pour calculer le pgcd de deux entiers et une fonction qui simplifie un rationnel passé en argument (et ne retourne rien, elle modifie directement son argument). La fonction de calcul du pgcd est une fonction *interne* de la bibliothèque, elle n'a donc pas besoin d'avoir son prototype dans le fichier `.h`. Modifiez vos fonctions d'addition et de multiplication pour qu'elles simplifient automatiquement le résultat avant de le retourner et vérifiez qu'elles fonctionnent encore correctement.

1.e] Pour compléter votre bibliothèque de manipulation de nombres rationnels, programmez les quelques fonctions simples suivantes :
- une fonction pour "cloner" un rationnel,
- une fonction qui convertit un rationnel en `double`,
- une fonction qui inverse le rationnel qu'on lui passe en argument,
- une fonction pour ajouter un `int` à un rationnel et retournant un pointeur vers la somme.

Exercice 2 : Utilisation de votre bibliothèque de rationnels

2.a] Maintenant que vous avez une bibliothèque de gestion de rationnels, vous allez pouvoir l'utiliser ! On sait que le nombre $\sqrt{2}$ peut s'écrire sous forme de fraction continue :

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \dots}}}}.$$

Cette fraction continue permet de calculer une approximation rationnelle de $\sqrt{2}$ mais aussi une suite de `double` qui converge vers $\sqrt{2}$. Écrivez donc une fonction qui calcule cette fraction continue à l'ordre n (où n est un `int` passé en argument) et retourne un pointeur vers le rationnel équivalent (faite bien attention aux fuites mémoires : pensez à libérer les rationnels utilisés dans des calculs intermédiaires). En utilisant les fonctions de votre bibliothèque, affichez le `double` correspondant au résultat renvoyé pour différents ordres d'approximation et vérifiez que cela converge bien vers $\sqrt{2}$.

Compilation séparée

Cette nouvelle fonction n'a pas sa place dans la bibliothèque de rationnels, elle doit donc aller dans le même fichier que le `main`. C'est une bonne occasion d'essayer la compilation séparée. Au lieu de compiler directement le code C en exécutable (comme vous faite habituellement), vous pouvez passer par des fichiers objet `.o` intermédiaires qu'il suffit ensuite de lier entre eux pour créer l'exécutable. L'étape coûteuse est la compilation, l'édition des liens est plus rapide. Donc cette méthode vous permet de ne compiler qu'une fois votre bibliothèque de rationnels en `.o` et ensuite de ne recompiler que votre fichier `main.c` en refaisant l'édition de liens. Voici les étapes de la compilation :

- `gcc -Wall -c ratio.c` : compile le fichier `ratio.c` en `ratio.o`
- `gcc -Wall -c main.c` : compile le fichier `main.c` en `main.o`
- `gcc -Wall ratio.o main.o -o main` : lie les fichiers objet `ratio.o` et `main.o` en un exécutable `main`
- `./main` : exécute le programme.

2.b] Programmez maintenant une fonction calculant la somme $\sum_{i=1}^{n} \frac{1}{i^2}$ et vérifiez qu'elle converge vers $\frac{\pi^2}{6}$. Pour des valeurs de n supérieures à 11 votre somme va certainement prendre des valeurs bizarres, à quoi est-ce du et comment peut-on améliorer cela ?

2.c] En utilisant une technique similaire au calcul de $\sqrt{2}$ il est possible de calculer n'importe quelle racine carrée. Soit a le plus grand entier tel que $a^2 \leq N$, alors on peut écrire $N = a^2 + b$ et on a :

$$\sqrt{N} = a + \frac{b}{2a + \frac{b}{2a + \frac{b}{2a + \dots}}}.$$

Vérifiez que cette formule est vraie en calculant le point fixe $x = \frac{b}{2a+x}$, puis programmez une fonction qui calcule une valeur approchée de la racine carrée d'un entier N en utilisant cette technique (la fonction peut renvoyer un `double`).