

# Cours de programmation séquentielle

## Librairie de fractions

### 1 Buts

- Introduction au langage C.
- Utilisation de structures.
- Implémentation de l'algorithme du PGCD.
- Utilisation de fonctions.
- Écriture d'une librairie de plusieurs fichiers et son utilisation dans un programme.
- Bonus: Utilisation de la ligne de commande.

### 2 Énoncé

Il s'agit d'écrire une librairie pour gérer des fractions. Cette librairie offrira notamment comme fonctionnalités la saisie, l'affichage, l'addition, la soustraction, la multiplication et la division de fractions. Les fractions devront toujours être stockées sous forme irréductible. Il faudra également gérer la division par zéro qui produira une erreur.

Ensuite, il faudra écrire un programme utilisant la librairie et permettant de vérifier que votre code se comporte comme attendu.

#### 2.1 Exercice supplémentaire

Lorsque vous aurez fini cette première partie, ajoutez à votre programme la possibilité de passer un calcul en argument de votre programme à la ligne de commande et retourner le résultat dans le terminal.

### 3 Cahier des charges

La librairie sera constituée **au moins** de:

- Une `struct fraction` composée de 2 champs: le numérateur et le dénominateur;
- 4 fonctions utilitaires obligatoires et une fonction bonus:
  - Une fonction qui affiche une fraction à l'écran.
  - Une fonction qui retourne une fraction irréductible.
  - Une fonction qui calcule le PGCD de deux nombres entiers positifs.
  - Une fonction qui élève une fraction à une puissance entière (positive ou négative) et retourne une fraction irréductible.

- Une fonction bonus qui lit une fraction à la ligne de commande en gérant les erreurs de saisie (voir plus bas);
- 5 fonctions de manipulation de fractions en place:
  - La fonction `fraction_add_inplace()` qui additionne deux fractions et stocke le résultat dans une autre fraction passée en argument.
  - La fonction `fraction_sub_inplace()` qui soustrait deux fractions et stocke le résultat dans une autre fraction passée en argument.
  - La fonction `fraction_mul_inplace()` qui multiplie deux fractions et stocke le résultat dans une autre fraction passée en argument.
  - La fonction `fraction_div_inplace()` qui divise deux fractions et stocke le résultat dans une autre fraction passée en argument (attention à la division par zéro).
  - La fonction `fraction_neg_inplace()` qui stocke le négatif d'une fraction dans une autre fraction passée en argument.
- 6 fonctions de manipulation de fractions:
  - La fonction `fraction_add()` qui additionne deux fractions et retourne une fraction irréductible.
  - La fonction `fraction_sub()` qui soustrait deux fractions et retourne une fraction irréductible.
  - La fonction `fraction_mul()` qui multiplie deux fractions et retourne une fraction irréductible.
  - La fonction `fraction_div()` qui divise deux fractions et retourne une fraction irréductible (attention à la division par zéro).
  - La fonction `fraction_neg()` qui retourne le négatif d'une fraction.
  - La fonction `fraction_to_double()` qui retourne la valeur en à virgule flottante de la fraction.

Une fois votre librairie de fraction écrite, vous pouvez l'utiliser pour évaluer le nombre pi. Les trois formules suivantes peuvent vous être utiles:

$$\sum_{n=1}^{\infty} \frac{1}{n^4} = 1 + \frac{1}{16} + \frac{1}{81} + \dots = \frac{\pi^4}{90}, \quad (1)$$

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^2} = 1 - \frac{1}{4} + \frac{1}{9} + \dots = \frac{\pi^2}{12}, \quad (2)$$

$$\prod_{n=1}^{\infty} \left( \frac{2n}{2n-1} \right) \left( \frac{2n}{2n+1} \right) = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \dots = \frac{\pi}{2}. \quad (3)$$

Quelle est la façon qui *converge* le plus rapidement?<sup>1</sup> Que se passe-t-il lorsque  $n$  devient trop grand et qu'on utilise que des entiers? Que se passe-t-il si on convertit tout en nombres à virgule flottante?

### 3.1 Syntaxe pour la partie bonus

La syntaxe pour la lecture des fractions à la ligne de commande, ainsi que de l'opération à effectuer doit avoir la syntaxe suivante: Les fractions se représentent

<sup>1</sup>Quelle est la méthode qui a besoin du moins d'itération pour atteindre une précision donnée?

comme `num denum` (notez l'espace entre `num` et `denum`), puis un opérateur (+, -, x, /, ^), puis une autre fraction également représentée comme `num denum`. Un nombre entier se représentera également sous la forme d'une fraction. La seule exception est le calcul du PGCD où on passe en argument deux nombres et PGCD. Le PGCD de ces deux nombres s'affiche comme résultat.

Pour la partie bonus la ligne de commande pourrait ressembler à ce qui suit.

```
> ./executable 3 10 + 8 15
5 6
> ./executable 3 10 x 8 15
4 25
> ./executable 3 1 x 8 11
24 11
> ./executable 3 10 x 8 1
12 5
> ./executable 3 10 ^ 2
9 100
> ./executable 15 10 PGCD
5
```

Afin de lire les arguments à la ligne de commande votre fonction `main()` doit avoir la forme suivante:

```
int main(int argc, char *argv[])
```

où `argc` est le nombre d'arguments de votre programme **y compris** le nom du programme et `*argv[]` un tableau de chaînes de caractères. Ainsi dans l'exemple

```
./executable 3 10 + 8 15
```

`argc == 6`, et `*argv[] == {"executable", "3", "10", "+", "8", "15"}`. Afin de transformer une chaîne de caractères en entier il faut utiliser la fonction `atoi()`. Pour déterminer si deux chaînes de caractères sont égales, vous pouvez utiliser la fonction `strcmp()`.

## 4 Remarques

- Pour le calcul de la fonction PGCD, utiliser l'algorithme de division d'Euclide.
- Les exemples ci-dessus peuvent être également utilisés pour tester votre programme.