R1.01 : Initiation au développement (partie 2)
Feuille TD n° 2

Modularité

Objectifs :
1.- Apprendre à utiliser un module.
2.- Apprendre à concevoir un module réutilisable

Rappel – Nombre rationnel – Fraction rationnelle

Un nombre rationnel est, en mathématiques, un nombre qui peut s'exprimer comme le quotient de 2 entiers relatifs.
On peut écrire les nombres rationnels sous la forme de fraction **a/b**, où **a**, le **numérateur**, est un entier relatif et **b**, le **dénominateur¸est un entier relatif** non nul.
Chaque nombre rationnel peut s'écrire d'une infinité de manières différentes sous forme de fractions, comme 1/2= 2/4 =3/6 = …
Mais il existe une forme privilégiée, **normalisée**, d'écriture:
Tout nombre rationnel non nul s'exprime de manière unique comme une fraction :

- Dont le numérateur et le dénominateur sont premiers entre eux
- Dont le dénominateur est positif.

On appelle cette expression une **fraction irréductible.**

Exercice – Module de 'gestion' des nombres rationnels

On souhaite créer un module contenant toutes les éléments / sous-programmes nécessaires permettant à un programmeur de :

- Déclarer des variables de type Fraction
- Écrire des programmes faisant des opérations sur des fractions, de manière compatible avec la définition mathématique

Le module doit être aussi complet que possible pour pouvoir être utilisé dans différentes contextes = différents programmeurs dans différents programmes.

Travail à faire

Définir le Type Abstrait de Données `Fraction`

1. Définir le type **`Fraction`** le plus approprié pour implémenter les nombres rationnels en accord avec la définition et propriétés mathématiques
2. ***Lister*** tous les sous-programmes et autres éléments qu'il vous semble nécessaire de mettre à disposition d'un programmeur pour qu'il/elle puisse manipuler des fractions dans un programme. Argumenter rapidement, pour chaque sous-programme, le choix d'une procédure / fonction.
3. Organiser ces sous-programmes par 'familles' de buts

Créer l'Interface du module `Fractions`

4. Écrire le lignes **les plus importantes l'interface** du module Fractions, c'est-à-dire celles qui structurent / organisent son contenu. Vous écrirez la totalité de l'Interface en TP.

   Préciser les pré et post-conditions des sous-programmes identifiés.

Préparer un jeu d'essai

5. Lister des valeurs constituant un jeu d'essai qui vous permettra de tester et valider le bon fonctionnement du module. Écrire le corps du main() qui permettra de tester ces valeurs.

Développer le corps du module

6. Ecrire les algorithmes suivants, en les accompagnant des dictionnaires associés :

a) D'un opérateur de comparaison

b) De l'opérateur additionner()

c) De l'opérateur diviser()

d) De la primitive reduire()

e) D'une fonction **`unsigned int pgcd (int entierA, int entierB)`**, qui retourne un nombre entier > 0 correspondant au plus grand entier > 0 divisant à la fois entierA et entierB.

Euclide (3eme siècle avant JC) a fourni une méthode pour calculer le PGCD de deux nombres.
Cette méthode est connue sous le nom d'algorithme d'Euclide.
Données : entierA, entierB, entierA >= entierB > 0
Résultat : pgcd, entier > 0

Elle se base sur le constat suivant :
La division entière de entierA par entierB s'écrit :
entierA = quotient * entierB + reste, avec 0 <= reste < entierB

En itérant cette division, et en remplaçant entierA par entierB et entierB par reste, on aboutira finalement à un reste nul.

➔ Le pgcd est alors le dernier reste non nul.

Exemple : PGCG(42, 30)

42 = 1 * 30 + 12
30 = 2 * 12 + 6 ◄
12 = 2 * 6 + 0 → le PGCD est 6, le dernier reste non nul

Nous généraliserons l'algorithme : il calculera le pgcd de 2 nombres entiers, positifs ou négatifs, avec la pré-condition suivante : les 2 entiers sont tels qu'au moins 1 d'eux est différent de 0.
Le PGCD calculé sera > 0.