Université Paris Diderot **Langages à objets avancés**
M1 **Année 2016-2017**

TP n° 8 : généricité, redéfinition d'opérateurs

Note : pour gagner du temps, vous pouvez ne vous intéresser qu'à l'opération plus.

Exercice 1 On souhaite écrire une classe `Fraction` qui représente les nombres en fractions entières. L'un des constructeurs de cette classe prendra en entrée deux entiers n et d (numérateur et dénominateur) et stockera les deux entiers correspondant à la fraction irréductible : (c.à.d tel que seul n peut être négatif, et tel que n et d ont été divisés par le pgcd)

1. Définissez cette classe et écrivez la méthode statique privée qui calcule le pgcd (le code du pgcd est fourni pour gagner du temps).
2. Redéfinissez l'opérateur de sortie `<<` pour permettre un affichage.
3. Surchargez les opérateurs `+`, `-` sur les fractions. **Rappel :** $\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}$
4. Vérifiez qu'une opération $\frac{1}{2} + 1$ ne compile pas. Ajoutez simplement un constructeur de fractions à un seul argument. Vérifiez que l'expression précédente est à présent évaluée. Quel est le mécanisme qui a été mis en oeuvre ?
5. Complétez éventuellement afin de pouvoir évaluer également $1 + \frac{2}{4}$.
6. Relisez vos déclarations de fonctions pour utiliser au maximum le mot clé `const`.

Exercice 2 On utilisera comme point de départ pour implémenter une pile générique le code suivant : [1]

```
template <class T> class Stack
{
private:
  int size;
  int top;
  T* stacktab;
  .....
};
```

1. Écrivez un constructeur vide et un autre avec comme argument la taille, des méthodes pour tester si la pile est vide, si elle est pleine, une méthode pour empiler (seulement lorsqu'il y a de la place, ne rien faire sinon) et une pour dépiler (on suppose qu'il y a bien des éléments).
2. Vous redéfinirez également l'opérateur `<<` pour permettre un affichage rapide.
3. Testez avec `Stack<int>` et `Stack<char>`.

1. Pourquoi mes templates ne sont ils pas reconnus à l'édition des liens ? La réponse sur : `http://cpp.developpez.com/faq/cpp/?page=Les-templates`

4. Vous redéfinirez également l'opérateur `<<` pour permettre un affichage rapide. Pour ce faire, comme vous aurez besoin de spécialiser l'opérateur (suivant que l'on ait une pile de pointeurs ou pas), et qu'apparement, il n'est pas possible de spécialiser une fonction amie, il ne faudra pas déclarer l'opérateur comme ami (`friend`), mais redéfinir l'opérateur à l'extérieur de la classe.
5. Testez avec `Stack<int>` et `Stack<char>`.
6. On veut maintenant empiler des `Fraction`s, mais `Fraction` n'a pas de constructeur vide. Que peut-on empiler ? Faites le nécessaire pour que l'affichage de `Fractions` soit convenable.

Exercice 3 On définit maintenant une classe abstraite `Element` qui prendra comme sous-classes `Operateur` et `Nombre`, les sous-classes de `Operateur<T>` seront `Plus<T>` et `Moins<T>` (binaire), et pour `Nombre`, on définira une seule sous-classe `Fraction` (On modifiera en conséquence le code précédent).

— Les opérateurs redéfiniront l'opérateur `()`. L'idée est que le code suivant doit fonctionner :

```
Fraction f1{2,4};
Fraction f2{9,-12};
Plus<Fraction> pl;
Fraction f4 = pl(f1, f2);
```

Bien sûr, créer un objet de type `Plus<Stack<int>> plus;` n'aura pas de sens et la ligne `plus(pile_entier, autre_pile_entier)` ne compilera pas.

— Écrivez maintenant une fonction `Fraction calcul(Element*[])` qui calcule le résultat de l'expression postfixée contenue dans le tableau en utilisant une pile d'`Element`s.