MOOC Intro POO C++
Exercices semaine 3

Exercice 9 : affichages (niveau 1)

Reprenez un de vos anciens exercices et ajoutez un opérateur d'affichage aux classes créées (par exemple les classes `Cercle`, `Point3D`, `Triangle`, `Article`, `Caddie`, ...).

Exercice 10 : nombres complexes (niveau 1)

> Cet exercice correspond à l'exercice n°52 (pages 129 et 305) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est d'implémenter la classe `Complexe` et de définir les opérateurs nécessaires pour écrire des opérations arithmétiques simples mettant en jeu des nombre complexes et réels.

1. Définir la classe `Complexe`, en utilisant la représentation cartésienne des nombres complexes (i.e. avec deux attributs `double` représentant respectivement les parties réelle et imaginaire du nombre complexe).
2. Ajouter à cette classe les constructeurs et destructeur nécessaires pour que le `main` suivant compile :

```
int main()
{
    Complexe defaut;
    Complexe zero(0.0, 0.0);
    Complexe un(1.0, 0.0);
    Complexe i(0.0, 1.0);
    Complexe j;

    return 0;
}
```

3. Définir les opérateurs nécessaires (il en faudra deux, l'un interne et l'autre externe) pour que cette suite de la fonction `main` compile et s'exécute correctement :

```
cout << zero << " ==? " << defaut;
if (zero == defaut) cout << " oui" << endl;
else cout << " non" << endl;

cout << zero << " ==? " << i;
if (zero == i) cout << " oui" << endl;
else cout << " non" << endl;
```

4. Continuer avec des opérateurs arithmétiques simples (là encore, des opérateurs externes seront nécessaires) :

```
j = un + i;
cout << un << " + " << i << " = " << j << endl;

Complexe trois(un);
trois += un;
trois += 1.0;
cout << un << " + " << un << " + 1.0 = " << trois << endl;

Complexe deux(trois);
deux -= un;
cout << trois << " - " << un << " = " << deux << endl;

trois = 1.0 + deux;
cout << "1.0 + " << deux << " = " << trois << endl;
```

5. Passer ensuite aux multiplications et divisions :

```
Complexe z(i*i);
cout << i << " * " << i << " = " << z << endl;
cout << z << " / " << i << " = " << z/i << " = ";
cout << (z/=i) << endl;
```

6. Et pour finir, les quelques opérateurs arithmétiques qui manquent encore :

```
Complexe k(2.0,-3.0);
z = k;
z *= 2.0;
z *= i;
cout << k << " * 2.0 * " << i << " = " << z << endl;
z = 2.0 * k * i / 1.0;
cout << " 2.0 * " << k << " * " << i << " / 1 = " << z << endl;
```

Indication : Soient $z_1 = (x_1, y_1)$ et $z_2 = (x_2, y_2)$ deux complexes ; on a :
$z_1 \cdot z_2 = (x_1 \cdot x_2 - y_1 \cdot y_2\ ,\ x_1 \cdot y_2 + y_1 \cdot x_2)$ et
$\frac{z_1}{z_2} = \left(\frac{x_1 \cdot x_2 + y_1 \cdot y_2}{x_2 \cdot x_2 + y_2 \cdot y_2}\ ,\ \frac{y_1 \cdot x_2 - x_1 \cdot y_2}{x_2 \cdot x_2 + y_2 \cdot y_2} \right)$.

Exercice 11 : encore un peu plus de polynômes (désolé !) (niveau 2)

Cet exercice correspond à l'exercice n°54 (pages 133 et 314) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est de faire de façon propre et complète une classe permettant la manipulation de polynômes (sur le corps réels).

Pour ceux qui n'ont pas fait le tutoriel de cette semaine :

- définissez la classe `Polynome` comme un tableau dynamique de `double` ;
- ajoutez-y la méthode `degre()` qui donne le degré du polynôme ; définissez à cette occasion le type `Degre` utilisé pour représenté le degré d'un polynôme ;
- ajoutez un constructeur par défaut qui crée le polynôme nul, un constructeur plongeant le corps des réels dans celui des polynômes (`Polynome(double);`) et un constructeur de copie ;
- ajoutez aussi un constructeur permettant de construire facilement des monômes ($a\ X^n$), en précisant le coefficient (`a`) et le degré (`n`) : `Polynome(double a, Degre n)`
- passez ensuite à l'opérateur (externe) << d'affichage dans un *ostream* ;
- Pour finir cette partie introductive, ajoutez les opérateurs de multiplication (par un polynôme et aussi par un réel).

```
Polynome operator*(Polynome const& q) const;
Polynome operator*(double) const;
Polynome& operator*=(Polynome const& q);
Polynome& operator*=(double);
```

et en externe

```
Polynome operator*(double, Polynome const&);
```

Pour tout le monde :

1. Ajoutez les opérateurs pour l'addition et la soustraction.
   Pour la soustraction, faire attention à ce que les polynômes restent toujours « bien formés », c'est-à-dire que le coefficient de plus haut degré soit non nul (sauf pour le polynôme nul).
   Il pourra, à ce sujet, être utile de faire une méthode privée `simplifie()` qui supprime les 0 inutiles (de degré trop élevé).
2. Ajoutez les opérateurs de comparaison == et `!=`.
3. Ajoutez une méthode `top` qui retourne la valeur du coefficient de plus haut degré du polynôme.

4. On s'intéresse pour finir à la division. On va pour cela implémenter une méthode (privée) `divise` qui effectue la division euclidienne de deux polynômes. Cette méthode permet de calculer à la fois le quotient et le reste de la division. Elle sera donc utile pour ensuite

implémenter les deux opérateurs `/` et `%`.

L'algorithme de la division euclidienne pour les polynômes qui, étant donné deux polynômes `N` (numérateur) et `D` (dénominateur), produit les deux polynômes `Q` (quotient) et `R` (reste) tels que `N = Q * D + R`, avec le degré de `R` strictement inférieur à celui de `D`, est le suivant :

```
Q = 0, R = N et δ = deg(R) - deg(D)
Tant que δ ≥ 0 et R ≠ 0
   a = rtop / dtop
   Q = Q + aX^δ
   R = R - aX^δ x D
   δ = deg(R) - deg(D)
```

où `rtop` est le coefficient de plus haut degré de `R` et `dtop` celui de `D`.

5. Terminer en implémentant les opérateurs de division et modulo à l'aide de la méthode `divise`.
