Correction

Remaniement ou refactorisation (*refactoring*) : les développeurs n'hésitent pas à revenir sur le code écrit pour le rendre plus «propre», le débarrasser d'éventuelles parties inutilisées, et le préparer à l'ajout de la fonctionnalité suivante. D'une manière plus générale, cette pratique propose une démarche de conception continue qui fait émerger la structure de l'application au fur et à mesure du développement.

Exemple:

```
typedef int BOOLEEN;

BOOLEEN estAnneeBissextile(int annee)
{
   return   ( ( (annee % 400) == 0 ) ||
            ( ( (annee % 4) == 0 )   &&  ( (annee % 100) != 0 ) ) );
}
```

Points positifs:

- respect des règles de codage (nommage variable et fonction)
- type booléen défini
- code « présenté » pour améliorer la lecture

Points Négatifs:

- commentaires manquants, donc code difficilement compréhensible

Amélioration proposée:

- mettre des commentaires, mais il y a mieux ...

Remaniement :

Il existe beaucoup de refactorisation, voici quelques exemples :

- **Extraction de fonction ou de méthode** : transformer une longue partie de code d'une fonction (ou méthode) en une fonction (ou méthode) plus courte en factorisant une partie de celle-ci dans une fonction (ou méthode privée) auxiliaire

- **Utiliser une variable explicative** : placer le résultat d'une expression dans une variable temporaire de façon à expliquer la finalité de l'expression :

```
BOOLEEN estMultipleDeQuatre = ( (annee % 4) == 0);
```

Exercice n° 1: ré écrire la fonction en utilisant des variables explicatives

```
typedef int BOOLEEN;

BOOLEEN estAnneeBissextile(int annee)
{
   BOOLEEN estMultipleDe4   = ( (annee % 4)   == 0);
   BOOLEEN estMultipleDe400 = ( (annee % 400) == 0);
   BOOLEEN estMultipleDe100 = ( (annee % 100) == 0);

   return ( estMultipleDe400 || ( estMultipleDe4   &&  !estMultipleDe100 ) );
}
```

Bilan:

L'utilisation des variables explicatives clarifie, simplifie et réduit la nécessité de commenter le code.

Exercice n°2: Extraction de méthode

1 . Ecrire le code qui permet le lancer des dés

```
class Joueur
{
  private:
   Pion pion;
   Plateau plateau;
   De des[];

  public:
   void prendreTour()
   {
     //lancer les dés
     int totalDuLancer = 0;
     for(i=0; i<NOMBRE_DE_DES; i++)
     {
        des[i].lancer();
        totalDuLancer += des[i].getValeur();
     }

     Case nouvellePosition = plateau.getCase(pion.getPosition(), totalDuLancer);

     pion.setPosition(nouvellePosition);
   }
};
```

2 . Remanier la méthode **`prendreTour()`** en utilisation la technique de l'extraction

Réponse:

```
class Joueur
{
  private:
   Pion pion;
   Plateau plateau;
   De des[];

   int lancerDes()
   {
     int totalDuLancer = 0, i;
     for(i=0; i<NOMBRE_DE_DES; i++)
     {
        des[i].lancer();
        totalDuLancer += des[i].getValeur();
     }
     return totalDuLancer;
   }

  public:
   void prendreTour()
   {
     int totalDuLancer = lancerDes();

     Case nouvellePosition = plateau.getCase(pion.getPosition(), totalDuLancer);

     pion.setPosition(nouvellePosition);
   }
};
```