Projet n°1 : Sérialisation dans un fichier

Le but de ce projet est d'implémenter des classes pour lire et écrire des fichiers binaires, pour réaliser une bibliothèque de sérialisation. La sérialisation consiste à stocker des données en format binaire portable. Elle est utilisé pour la sauvegarde de fichiers, les protocoles réseaux, etc.

Étape 1 : Fichier binaire

Dans cette étape, vous devrez implémenter deux classes, `OBinaryFile` pour les fichiers en écriture, et `IBinaryFile` pour les fichiers en lecture. Comme il s'agit de classes représentant des ressources, il sera nécessaire d'appliquer la *Rule of Five*.

L'implémentation de ces deux classes utilisera l'API C qui se trouve dans `<cstdio>`, il est donc formellement interdit d'utiliser toute autre API.

Pour permettre une meilleure compatibilité, le format du fichier binaire doit respecter ces règles :

1. Les données sont stockées dans l'ordre d'ajout : *First In First Out (FIFO)*
2. Les collections d'objets doivent stocker d'abord le nombre d'éléments puis les objets dans le même ordre que leur itérateur.

Étape 2 : Opérateurs de sérialisation

Dans cette étape, vous utiliserez les deux classes définies précédemment pour sérialiser et désérialiser un ensemble de types de base en surchargeant les opérateurs `<<` et `>>`. Attention, le format de sérialisation devra être en *big endian*. On ne prendra pas en charge les pointeurs et les références.

Puis vous aurez à sérialiser des types conteneurs de la bibliothèque standard de manière générique.

Exemple d'utilisation

```
#include <cassert>
#include "Serial.h"

struct Foo {
  int16_t i;
  double d;
  std::string s;
};

serial::OBinaryFile& operator<<(serial::OBinaryFile& file,
    const Foo& foo) {
  return file << foo.i << foo.d << foo.s;
}

serial::IBinaryFile& operator>>(serial::IBinaryFile& file,
    Foo& foo) {
  return file >> foo.i >> foo.d >> foo.s;
```

```
}

int main() {
  Foo foo;
  foo.i = 42;
  foo.d = 69.0;
  foo.s = "Hello";

  {
    serial::OBinaryFile out("foo.bin");
    out << foo;
  }

  struct Foo copy;

  {
    serial::IBinaryFile in("foo.bin");
    in >> copy;
  }

  assert(foo.i == copy.i);
  assert(foo.d == copy.d);
  assert(foo.s == copy.s);
}
```