Travaux dirigés C++ n°2

Informatique

—IMAC 2e année—

Introduction à l'Objet

Ce TD a pour but d'introduire le concept d'objet en C++. Les étudiants apprendront à écrire une classe, son constructeur et son destructeur.

▸ Exercice 1. *Une première classe*

Cet exercice consiste à créer une classe de vecteurs de `double` *de taille n, dans l'objectif de faire des opérations mathématiques sur ces vecteurs.*

1. *Quels peuvent être les attributs et les méthodes de cette classe ?*
2. *Créez 3 fichiers* `VectorD.cpp`, `VectorD.hpp` *et* `main.cpp`.
3. *Dans le fichier* `VectorD.hpp`, *commencez par mettre les "guards" :*

```
#ifndef SOME_UNIQUE_NAME_HERE
#define SOME_UNIQUE_NAME_HERE

// your declarations here

#endif
```

   ou bien une version plus simple, mais pas dans la norme du C++ :

```
#pragma once

// your code here
```

4. *Dotez votre classe d'un attribut public* `std::vector<double> m_data` *contenant les composantes du vecteur. Quel* `include` *faut-il rajouter?*
5. *Faites un* `main.cpp` *qui ne fait rien de particulier (mais qui inclue* `VectorD.hpp`*).*
6. *Compilez avec* `g++ -Wall -O2 -std=c++11 main.cpp VectorD.cpp -o tp2`

▸ Exercice 2. *Construisons, détruisons*

Cet exercice consiste à comprendre le mécanisme de construction-destruction.

1. *Cette classe a-t-elle besoin de paramètres à la création? Si oui, lesquels? Créez le constructeur adéquate avec paramètre(s). À savoir, la classe* `std::vector` *possède, entres autres, un constructeur par défaut* `std::vector()`*, un constructeur qui prend une taille de vecteur en paramètre* `std::vector(size)` *et un constructeur qui en plus initialise le vecteur avec une certaine valeur* `std::vector(size, value)`*.*

2. *Dans quel cas est créé le constructeur par défaut? Attribuez au(x) paramètre(s) une valeur par défaut, vous obtenez alors en bonus un constructeur par défaut.*

3. *Faites un constructeur par recopie.*

4. *Faites un destructeur.*

▸ Exercice 3. *Norme et produit scalaire*

L'objectif de l'exercice est de faire une méthode de classe et voir le mot clés `const`*.*

1. *Implantez une méthode* `display() const` *qui affiche le contenu d'un* `VectorD` *sur le terminal.*

2. *Implantez une méthode* `dot(const VectorD &v) const` *qui calcule le produit scalaire de deux vecteurs.*

3. *Implantez une méthode* `norm() const` *qui renvoie la norme du vecteur.*

4. *Quelle est la différence entre le* `const` *pour un argument d'une méthode et le* `const` *en fin de définition de méthode?*

▸ Exercice 4. *Operateur* =

L'objectif de l'exercice est de connaître les mécanismes de mise en place de l'opérateur =

1. *Testez votre programme actuel avec les instructions suivantes.*

```
VectorD b(10);
VectorD a;  // default constructor
a = b;      // operator =
```

   On s'aperçoit que ça fonctionne sans qu'on ait définit l'opérateur =*. Pourquoi ?*

2. *Pour s'exercer, implantez quand même l'operateur* =*.*