Travaux dirigés C++ n°5

Informatique

—IMAC 2e année—

Gestion d'erreur en C++

Le but de ce TD est de se familiariser avec la gestion d'erreur en C++ au travers des assertions et des exceptions, leur levée et leur gestion.

▸ Exercice 1. *Tester ce qui ne rate jamais*

Objectif de l'exercice : Savoir utiliser le mot clé assert

1. *Téléchargez le code disponible sur le site de l'enseignant et allez dans le répertoire* `Assertion`*. Compilez le et exécutez le.*
2. *modifier le code en ajoutant un* `assert` *pour s'assurer que l'argument de la méthode* `myPrivatePlop` *soit toujours positif ou nul. Cet* `assert` *sera assorti du message d'erreur "error: Plop::myPrivatePlop: 'value' should be positive". Pourquoi est-ce légitime de mettre un* `assert` *plutôt qu'une exception pour gérer cette erreur?*
3. *Modifiez le constructeur par défaut de la classe* `Plop` *avec un* `static_assert` *pour s'assurer que la taille du tableau* `m_data` *est toujours inférieur à 6. Ce* `static_assert` *sera assorti du message d'erreur "m_data should be lower than 6". A quel moment peut-on voir ce message d'erreur s'afficher? Faites un test.*
4. *Les* `assert` *ralentissent-ils le code? Et les* `static_assert`*?*
5. *Recompilez avec l'option* `-DNDEBUG` *afin d'ignorer les* `assert` *(pas les* `static_assert`*). Vérifiez que l'assertion est maintenant bien ignorée.*

▸ Exercice 2. *Les exceptions de la STL*

Objectif de l'exercice : Savoir attraper les exceptions des outils de la STL.

1. *Récupérez le code dans le répertoire* `STL` *et lisez le.*
2. *Décommentez tour à tour chacune des instructions commentées dans la fonction* `main()`*. Chacune de ces fonctions lève une exception. Modifiez le code pour attraper l'exception afin de ne pas interrompre le programme (inutile de trouver une solution au problème levé).*

▸ **Exercice 3.** ***Lancer des choses***
Objectif de l'exercice : Savoir lever une exception

1. *Récupérez le code dans le répertoire* `VectorD` *et (re)lisez le.*
2. *Changer la gestion des erreurs en utilisant des exception pour les méthodes :*
   - `operator+(const VectorD &v)`
   - `operator-(const VectorD &v)`
   - `double dot(const VectorD &v)`

   Vous utiliserez l'include `#include <stdexcept>` *et pour lever des exceptions de la stl, par exemple (*`std::length_error`*).*
3. *Changer la gestion des erreurs des fonctions suivantes :*
   - `VectorD::save(const std::string &filename)`
   - `VectorD::load(const std::string &filename)`

▸ **Exercice 4.** ***Classe d'exception***
Objectif de l'exercice : Créer ses propres classes d'exception

La classe exception est la classe de base de toutes les exceptions lancées par la bibliothèque standard (inclure l'entête `exception`*). Elle est définie comme suit :*

```
class exception
{
public:
  exception(){} //Constructor.
  virtual ~exception(); //Destructor.
  virtual const char* what() const; // Error message
};
```

`what()` *: renvoie le message d'erreur sous la forme d'une chaîne de caractères (type* `char *`*).*

1. *Dans le répertoire* `MyExceptions`*, créez votre propre classe d'exception que vous appellerez* `VectorDException` *(dans des fichiers* `VectorDException.hpp` *et* `VectorDException.cpp`*) qui doit hériter de la classe* `std::exception`*. Votre classe d'exception devra afficher :*
   - *une phrase décrivant l'erreur (type* `string`*)*
   - *le numéro de l'erreur (type* `int` *ou* `class enum`*)*
   - *le niveau de l'erreur (erreur fatale, erreur mineure, ...), en utilisant un* `int` *ou mieux* `enum class`
2. *Ajouter les méthodes suivantes votre classe* `VectorDException` *:*
   - *Un constructeur prenant en paramètre le message d'erreur, le numéro et le niveau de l'erreur.*

- *Un constructeur par défaut avec des valeurs par défaut (niveau = minor, code= 0 et message= " ")*
- *Une méthode* `what()` *redéfinie et retournant le message d'erreur.*
- *Un desctructeur* `virtual` ∼`VectorDException()`*.*

Une exception doit renvoyer un texte du genre

```
Exception launched :
   Level   : major
   Code    : 1
   Message : my message ...
```

3. *Modifiez votre classe* `VectorD` *de telle sorte qu'elle utilise vos exceptions.*
4. *Faites des tests.*