Travaux dirigés C++ n°6

Informatique

—IMAC 2e année—

Les exceptions

L'objectif de ce TD est de découvrir et comprendre le mécanisme de la gestion des exceptions en C++.

▸ Exercice 1. *Une situation exceptionnelle !*

1. *Exécuter le code suivant :*

```
#include <iostream>
int main(int argc, char const *argv[])
{
int a = 10;
int b = 0;

std::cout << "a/2 = " << a/2 << std::endl;
std::cout << "a/b = " << a/b << std::endl;
std::cout << "a/5 = " << a/5 << std::endl;
std::cout << "Terminé !! " << std::endl;

return 0;
}
```

2. *Est ce que le programme plante à l'étape de la compilation ? à l'étape de l'exécution ? Si oui, c'est au niveau de quelle ligne?*

▸ Exercice 2. *Intercepter une exception : try-catch*

1. *Pour gérer l'exception, identifier d'abord la partie du code qui doit être surveillée, pour cela, utiliser le bloc* ***try****.*

   Pour créer une exception amenant l'instruction try à réagir, on utilise le mot-clé ***throw****. Ainsi, lorsqu'une exception survient au cours de l'exécution (on dit l'exception est "lancée"), elle peut être interceptée (capturée) avec le bloc* ***catch****.*

2. *Si le dénominateur (variable b) est nul, lever (throw) une exception* ***entière*** *vue comme un code d'erreur (la valeur du dénominateur par exemple).*

3. *On souhaite maintenant que throw lance une chaine de caractère au lieu d'un entier. Modifier le programme précédant pour lever une exception de type* ***std::string****.*

4. *Et si on souhaite capturer toute exception, quelle qu'elle soit. Modifier le bloc catch pour qu'il désigne n'importe quelle exception, et qu'il signale une erreur dans le flux d'erreur (cerr), de telle sorte à afficher :*

```
a/2 = 5
Exception inconnue dans votre programme !!
a/5 = 2
Terminé !!
```

5. *Expliquer pourquoi le programme termine son exécution malgré la division par zéro.*

6. *Qu'est ce qui ce passe si le bloc try {... throw....} existe alors que le bloc catch(...){..} n'existe pas ?*

▸ **Exercice 3.** ***La classe exception***

Dans les l'exercices précédants, on a lancé des exceptions de type int et type string. On peut aussi créer une classe bien plus complexe pour lancer un objet contenant plusieurs attributs:

- *une phrase décrivant l'erreur*
- *le numéro de l'erreur*
- *le niveau de l'erreur (erreur fatale, erreur mineure...)*
- *l'heure à laquelle l'erreur est survenue ...*

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() throw(){ } //Constructeur.
virtual  ~exception() throw(); //Destructeur.

virtual const char* what() const throw();
};
```

what() : renvoie les informations sur l'erreur sous la forme d'une chaîne de caractères.

On peut alors créer sa propre classe d'exception en la dérivant grâce à un héritage.

1. *Créer votre classe* ***Erreur*** *qui dérive de la classe* ***std::exception*** *et qui a les attributs suivants :*

   - *Un entier décrivant le niveau de l'erreur*
   - *Un entier décrivant le code de l'erreur*
   - *Un message (string) décrivant l'erreur*

2. *Ajouter les méthodes suivantes à votre classe* ***Erreur*** *:*

   - *Un constructeur initialisant les valeurs par défaut (niveau = 0, code= 0 et message= " ")*
   - *Une méthode getNiveau() retournant le niveau de l'erreur.*
   - *Une méthode getCode() retournant le code de l'erreur.*
   - *Une méthode virtuelle what() redéfinie et retournant le message d'erreur.*
   - *Un desctructeur virtual ∼Erreur().*

3. *Modifier votre programme de l'exercice précédant de telle sorte à utiliser cette fois votre classe* ***Erreur*** *et obtenir l'affichage suivant :*

   -
```
a/2 = 5
Exception lancée :
    Niveau : 1
    Code : 2
    Message : Dénominateur nul !!
a/5 = 2
Terminé !!
```