Les exceptions

Travaux dirigés et pratiques 7 - Programmation par objets 2 (UE ULIN606)

1 Exceptions prédéfinies en C++

La figure 1 vous présente les exceptions standard prévues en C++. La classe mère, `exception`, a l'interface suivante :

```
class exception {
  public:
    exception() throw();
    exception(const exception&) throw();
    exception& operator=(const exception&) throw();
    virtual ~exception() throw();
    virtual const char* what() const throw(); // affiche le type d'exception
};
```

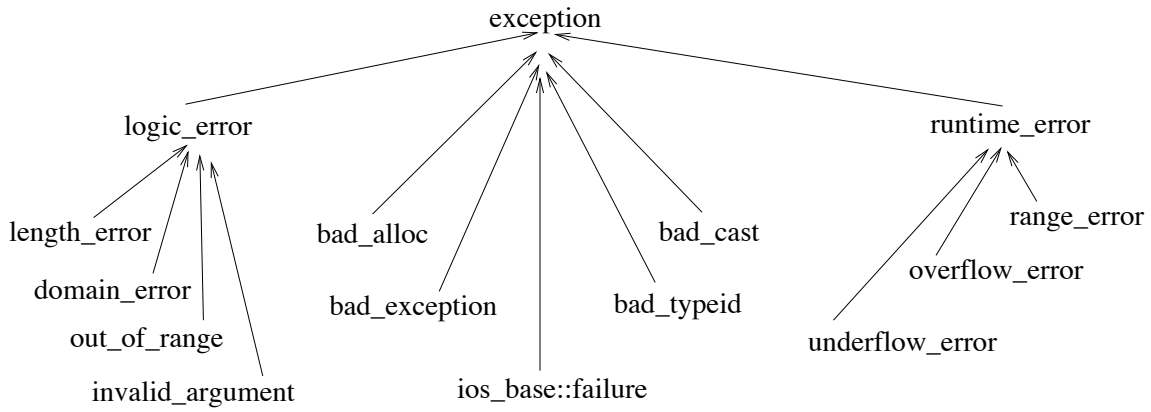

FIGURE 1 – Hiérarchie des exceptions en C++

Certaines exceptions sont signalées par les méthodes de la librairie standard C++. Lorsque vous aurez terminé les exercices suivants en TP, vous pouvez essayer de les faire apparaître.

2 File et file bornée

1. Proposez une classe paramétrée `File`, représentant les files d'attente (premier entré, premier sorti). La file sera implémentée à l'aide d'un `vector` et prévoyez une gestion d'exceptions pour cette classe en étudiant les pré-conditions et post-conditions des opérations. Les exceptions signalées par la file seront également paramétrées par le type des éléments stockés dans la pile ; dans le verdict, affichez le type de l'exception (utilisez la méthode `what`), le type des éléments stockés dans la file et le

contexte (nom de la méthode) où s'est produite l'erreur ; placez les exceptions dans la hiérarchie déjà existante. Les opérations à prévoir sont les suivantes :
- création,
- consultation du nombre d'éléments stockés,
- prédicat permettant de savoir si la file est vide,
- retrait et retour de l'élément le plus ancien,
- ajout d'un élément,
- consultation de l'élément le plus ancien.

2. Proposez une classe paramétrée `FileBornée`, spécialisation de `File` et complétez la gestion des exceptions. La capacité d'une file bornée doit être pour cet exercice un paramètre de généricité.
3. Ecrivez une opération (méthode ou fonction ?) permettant de transvaser des éléments d'une file bornée d'un certain type T (ex. Enfant) dans une file bornée du même type.
4. Même question en transvasant dans une file bornée d'un type plus général (ex. Personne).
5. Etudiez l'opération inverse (transvaser une file de personnes dans une file d'enfants) et le cas général où les types des deux files sont quelconques.
6. Ecrire un petit programme incluant des manipulations de files et de files bornées et des récupérations d'exceptions.

3 Distributeur de friandises

Nous étudions le fonctionnement d'un distributeur de friandises relativement standard. Le distributeur est décrit par une marque et un numéro de série et se compose de différents compartiments numérotés de taille fixe qui se comportent comme des files d'attente bornées (la taille ne peut changer d'un compartiment à un autre). Un compartiment stocke des friandises qui doivent être toutes de même type. Les friandises sont décrites par le code de l'atelier de fabrication, une date de péremption et un type décrit par un nom ("nougat", "nuts", etc.) et un prix qui ne doit pas dépasser la constante `prixMaximum`.

Sur le distributeur, on peut :
- associer (par une sorte d'étiquette) un type de friandise à un compartiment de numéro donné, pour cela le compartiment doit être vide,
- désaffecter un compartiment : aucun type de friandise n'y est associé,
- modifier le prix d'un type de friandise stocké dans un compartiment de numéro donné,
- ajouter une friandise dans un compartiment de numéro donné,
- regarnir de friandises un compartiment de numéro donné,
- examiner (sans la retirer) la friandise qui est sur le devant d'un compartiment de numéro donné,
- retirer une friandise dans un compartiment de numéro donné,
- vider un compartiment de numéro donné.

Ces méthodes seront réalisées par délégation du traitement au compartiment donné.

1. Proposez un schéma des classes. Prévoyez les exceptions associées à cette modélisation : attachez-vous à signaler, dans les méthodes d'une classe, des erreurs dont la signification est en rapport avec la sémantique de la classe. Par exemple `indiceHorsBornes` a un sens pour un tableau, mais pas pour un distributeur de friandises, pour ce dernier, c'est au contraire une erreur `compartimentInexistant` qui aurait un sens.
2. Ecrivez le code des méthodes en récupérant les exceptions d'un niveau interne de description (par exemple des exceptions qui ont un sens pour une file bornée) pour signaler des exceptions du niveau adéquat (par exemple des exceptions qui ont un sens pour un compartiment).
3. Votre solution convient-elle pour traiter des distributeurs dans lesquels la taille pourrait varier d'un compartiment à un autre ?