Programmation orientée objet (C++) – ING2-GSI

TP7 : Gestion des exceptions & Bibliothèque STL

Exercice 7.1 : Tableau à intervalle d'indices

71.1. Les cases des tableaux C ou C++ sont numérotées à partir de 0. On veut ici écrire une classe permettant d'utiliser des tableaux dans lesquels l'intervalle des indices est quelconque. Déclarer une telle classe pour manipuler des tableaux d'entiers à indices quelconques, appelée `ArrayInt`. Vous pouvez utiliser un attribut de type `vector` ou une zone allouée dynamiquement. Munir cette classe d'un constructeur permettant de choisir l'indice minimum et l'indice maximum du tableau, ces indices étant des entiers signés. Définir deux accesseurs permettant d'accéder à l'indice minimum et à l'indice maximum du tableau. Définir un constructeur par recopie.

7.1.2. Définir une classe `ExceptionArrayInt`, sous-classe de `std::exception` (défini dans `<exception>`). Cette classe représentera les accès incorrects à un `ArrayInt`, c'est-à-dire l'accès à une valeur d'indice qui n'est pas incluse dans l'intervalle d'indices. Une instance mémorisera un message d'erreur (`std::string`) et l'indice erroné. Définir un constructeur permettant d'initialiser ces deux attributs. Définir la méthode `what`.

7.1.3. Écrire une méthode `at` de `ArrayInt` prenant comme paramètre un indice et retournant l'entier situé à cet indice dans le tableau. Si l'indice passé ne fait pas partie de l'intervalle d'indices, lever une `ExceptionArrayInt`. Écrire l'opérateur d'accès `[]` faisant la même chose que `at` et permettant d'accéder au contenu d'un `ArrayInt` de la même façon qu'à un tableau C/C++ (ex : `cout << a[4];`).

7.1.4. Écrire une méthode `set` prenant comme paramètre un indice et une valeur et modifiant la valeur mémorisée dans le tableau pour l'indice passé.

7.1.5. Écrire l'opérateur d'affectation =, et l'opérateur de comparaison == sur `ArrayInt`.

7.1.6. Écrire l'opérateur de sortie sur un flux <<. À titre d'exercice, cet opérateur utilisera l'accesseur à l'indice minimum du tableau, mais n'utilisera pas l'accesseur à l'indice maximum.

Exercice 7.2 : Conteneur liste std::list et algorithmes de la bibliothèque standard

7.2.1.Écrire une classe `Resultat` qui mémorise le résultat d'un tirage aléatoire. Un `Resultat` sera formé de 5 entiers compris entre 1 et 49. Les entiers seront stockés dans une liste d'entiers (`std::list`), qui sera un attribut de la classe. Définir un constructeur sans argument, et un constructeur par recopie.

7.2.2. Définir une méthode `ajouterNumero` qui ajoute au `Resultat` le numéro passé en paramètre. Cette méthode lèvera une exception dans le cas où l'entier passé en paramètre n'est

pas compris entre 1 et 49, s'il figure déjà dans la liste des numéros du *Resultat* (pour cela, on utilisera l'algorithme *std::find*), ou si le *Resultat* contient déjà 5 numéros.

7.2.3. Écrire une méthode *trier* de *Resultat* qui trie dans l'ordre croissant les numéros stockés dans la liste. Écrire un opérateur de sortie sur un flux.

7.2.4. Écrire une classe *Date* formée de trois attributs entiers : *année*, *mois*, *jour*. Définir constructeur(s), accesseurs et mutateurs, opérateur d'affectation =, opérateur de sortie sur un flux, opérateurs de comparaison *==, !=, <, <=, >, >=.*

7.2.5. Rajouter à *Resultat* un attribut *Date* pour mémoriser la date du tirage.

7.2.6. Définir une classe *EnsembleResultat* stockant une liste de *Resultat*, et munir cette classe d'un constructeur par défaut, d'une méthode permettant de rajouter un *Resultat* (et levant une exception si la date du résultat à ajouter figure déjà dans la liste des résultats stockés. Réfléchir à la façon d'utiliser *find* pour faire cela. Noter que la comparaison entre le *Resultat* à ajouter et les résultats stockés se fait uniquement sur la date, et non sur les numéros stockés dans les *Resultat*), et d'une méthode triant la liste des résultats dans l'ordre chronologique (là aussi, on réfléchira à la façon d'utiliser sort pour faire cela).

Références

std::list

http://www.cppreference.com/wiki/stl/list/start

http://www.cplusplus.com/reference/stl/list/

algorithmes

http://www.cppreference.com/wiki/stl/algorithm/start

http://www.cplusplus.com/reference/algorithm/