FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Auteur : Yves Roggeman

ÉNONCÉ DE PROJET

ULB
UNIVERSITÉ LIBRE DE BRUXELLES

Date : 05/10/2014
Document : Cpp_2015-01
N° de version : 16
Destinataires : Inscrits INFO-F-202

Confidentialité :
Public ☒
Interne ☐
Confidentiel ☐
Ne pas diffuser sans autorisation ☐
Autre ☐

Réf. : 2014-009-001

Objet : Langages de progammation 2

Pièces jointes :

Commentaires : Projet de C++

Ceci est un des deux énoncés de l'épreuve de première session qui se déroule en janvier. Il sert de base à l'épreuve orale ; l'évaluation finale porte sur l'acquisition des concepts démontrée à cette occasion.

Le but de cet exercice de programmation est de démontrer une bonne connaissance et un usage adéquat des constructions syntaxiques du langage de programmation C++, du mécanisme des **`templates`** et de la hiérarchie d'héritages. L'évaluation portera donc essentiellement sur l'écriture, la codification, la présentation, les constructions et la pertinence des choix effectués dans ce contexte.

A. PROBLÈME

Il est demandé de concevoir, en C++, un modèle de classe réalisant un type de tableau multidimensionnel. On entend par là un type composé homogène d'éléments d'un type de base donné auxquels on accède à l'aide d'opérateurs [] itérés. Le nombre d'opérateurs nécessaires est la dimension du tableau. Celle-ci peut être quelconque : au moins 1, mais sans limite *a priori*.

A.1. Problème de base

Le type de base du tableau — le type des éléments — est bien sûr un paramètre du **`template`**, de même que la dimension, *i.e.* le nombre d'indices utilisés.

À la construction d'un objet de ce type, on fournit la taille de chacune de ses composantes, c'est-à-dire les valeurs — de 0 à *n*-1, si la taille est *n* — que peuvent prendre l'indice correspondant. Ces tailles peuvent évoluer en cours de vie de l'objet, par assignation d'un autre tableau, par exemple.

Les éléments de l'objet seront placés, en privé, dans un simple vecteur « *à la C* » dans l'ordre habituel ligne par ligne et *de manière efficace et compacte*. Il ne s'agit donc pas de définir ce tableau comme un vecteur de vecteurs, ce qui induirait des indirections (par pointeurs ou références) supplémentaires.

L'interface de la classe doit permettre d'interroger sa dimension, la taille de chacune de ses dimensions, ainsi que la taille globale (le nombre total d'éléments). Il devra également être possible de réaliser un output « lisible » du contenu d'un tableau par simple appel à un opérateur « << »

L'accès aux éléments se fera exclusivement par l'usage d'opérateurs [] surchargés, donc selon une syntaxe similaire à celle d'un tableau « *à la C* », mais en garantissant un contrôle sur la valeur du paramètre fourni.

Les erreurs ou incohérences détectées par les méthodes seront traitées par un mécanisme d'exceptions.

Il est évidemment demandé de définir correctement toutes les méthodes spéciales lorsque cela se justifie.

À la limite, ce modèle de tableau doit permettre de définir un simple vecteur de dimension 1. Si nécessaire, des spécialisations de **`template`** seront définies dans ce but.

A.2. Sous-question n° 1

Une descendance de cette classe sera également conçue pour définir des tableaux dont chaque indice peut prendre ses valeurs entre deux bornes quelconques fournies à la construction.

L'interface devra permettre de questionner ces valeurs.

A.3. Sous-question n° 2

Il est également demandé de réaliser des tranches (*slices*) des objets des classes décrites ci-dessus, c'est-à-dire un type restreignant l'accès à un « sous-tableau » défini par un sous-ensemble des valeurs

pour chacun de ses indices. Ces sous-ensembles sont définis, indice par indice, par une valeur inférieure, une valeur supérieure et un pas valant 1 par défaut (on peut donc ne pas le spécifier à la construction).

Cette construction doit pouvoir s'appliquer tant à un tableau de base que celui visé à la sous-question n° 1.

Ces tranches constituent de simples accès filtrés aux éléments du tableau : toute modification via elles impacte le tableau d'origine. Leur implantation doit être efficace : aucune copie d'éléments du tableau n'est tolérée, ni à la construction ni en cours de vie de la tranche.

Par contre, il doit être possible de copier explicitement (par construction ou assignation) une tranche *de pas 1* vers un tableau tel que défini à la sous-question n° 1.

B. RÉALISATION

Il vous est donc demandé d'écrire en C++ les définitions de toute la hiérarchie des classes, des modèles et spécialisations de modèles de classes décrits ci-dessus.

Pour tester votre réalisation, vous instancierez des objets appartenant aux diverses classes concrètes auxquels les opérations définies seront appliquées. Vous vérifierez également le bon fonctionnement des opérations standard : construction, construction de copie et de transfert, assignation de copie et de transfert et destruction.

De plus, la construction devra garantir la non-accessibilité aux attributs et méthodes techniques, ainsi que la transparence et la simplicité d'usage à partir d'un programme. Si nécessaire, des définitions par « **typedef** » seront introduites.

Le programme devra montrer sa robustesse aux cas particuliers, y compris les instanciations de **template** ou les constructions d'objets erronées. Les caractéristiques « **final** » ou non seront étudiées, ainsi que le bon usage éventuel du polymorphisme. Le bon choix des tests est un élément d'évaluation.

D'une manière générale, la concision, la précision, la lisibilité (clarté du texte source), l'efficacité (pas d'opérations inutiles ou inadéquates) et le juste choix des syntaxes seront des critères essentiels d'appréciation. De brefs commentaires dans le code source sont souhaités pour éclairer les choix de codification.

C. REMISE

Votre travail doit être réalisé pour le jeudi 18 décembre 2014 à 18 heures au plus tard, mais il vous est vivement conseillé de l'achever pour le jeudi 20 novembre 2014. Vous remettrez tous vos codes sources empaquetés en un seul fichier compacté (« .zip » ou autre) *via* le site du cours (INFO-F-202) sur l'Université virtuelle (http://uv.ulb.ac.be/). Ceux-ci devront contenir en commentaire vos matricule, nom, prénom et année d'études.

Le jour de l'examen, vous viendrez avec une version imprimée — un *listing* — de ces divers fichiers. Une impression du résultat d'une exécution du programme est également demandée.