UNIVERSITÉ LIBRE DE BRUXELLES ULB

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Langages de programmation 2 : Projet C++

Yves Roggeman *

INFO-F202 — Année académique 2018–2019

Résumé

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 donc de démontrer une connaissance approfondie et un usage adéquat des constructions du langage de programmation C++, en particulier de l'usage des « **`template`** » et du mécanisme d'héritage.

Le problème posé consiste à définir une structure de données de *vecteur creux*, c'est-à-dire un tableau de grande taille mais dont les valeurs sont majoritairement nulles. Il s'agit de définir une structure de données abstraite correspondant à un tableau, puis une réalisation concrète héritière d'un conteneur adéquat que vous définirez également.

1 Le conteneur

Le tableau est une simple structure homogène à n composantes indicées de 0 à $n - 1$. Le type de ces composantes doit être paramétrique.

Il vous est demandé de définir un type abstrait correspondant à ce simple type. Outre les opérations « spéciales » habituelles de construction, destruction, copie, transfert, etc. il vous est simplement demandé de définir l'opérateur `[]` d'indiçage. Le nombre de composantes du tableau est fourni à la construction ; les accès lèvent une exception en cas de tentative d'accès à une position au-delà de cette valeur.

Comme on suppose que la dimension est grande et que les éléments non nuls sont peu nombreux, le type concret qui vous est demandé sera celui d'un arbre binaire dont les sommets (nœuds et feuilles) comprennent deux informations : l'indice de l'élément et sa valeur. L'arbre est bien sûr ordonné par indices croissants. On suppose qu'une valeur nulle n'est jamais insérée dans un tel arbre, mais si une composante le devient, elle n'est pas supprimée de la structure.

Un tel type concret de vecteur creux est bien sûr également héritier d'un type d'arbre binaire ordonné dont le type des éléments est paramétrique et pour lequel les opérations d'insertion et de recherche sont définies. Cette dernière utilise donc les opérateurs < et == qui doivent être définis pour le type de ses éléments.

Vous définirez également un itérateur de cet arbre binaire qui le parcourt dans l'ordre logique (donc l'ordre symétrique ou *inorder*).

2 Implantation

Il vous est demandé de réaliser une hiérarchie de *templates* de classes permettant d'instancier des vecteurs creux ainsi implantés. Vous pouvez bien sûr définir des classes ou méthodes « techniques »

*Université libre de Bruxelles (ULB) <yves.roggeman@ulb.ac.be>

qui vous sont utiles pour cela, mais en veillant bien à respecter le principe général d'encapsulation des données et entités.

Lorsqu'on consulte un élément, s'il n'est pas présent mais correspond à un indice possible, l'opération doit renvoyer une valeur nulle. Inversement, lorsqu'on tente de modifier un élément encore absent, la valeur est insérée dans le conteneur.

Quel que soit le type des éléments, leur valeur nulle est celle qui est implicitement convertible comme **`false`** dans le type **`bool`** et réciproquement.

On suppose bien sûr que les valeurs sont insérées dans un ordre aléatoire ; en effet, l'insertion dans l'ordre séquentiel (indices croissants ou décroissants) serait tout à fait inefficace, raison pour laquelle les copies se font sur base de la structure de l'arbre, ce qui doit rester transparent grâce à la définition de méthodes polymorphes à cet effet.

Afin de tester le bon fonctionnement de ces classes, vous écrirez également deux fonctions génériques de parcours qui affichent les composantes : l'une d'un tableau abstrait (donc y compris ses composantes nulles, même si elles sont physiquement absentes), l'autre d'un arbre binaire ordonné (donc seules les composantes présentes seront affichées).

3 Réalisation

Il vous est demandé d'écrire en C++ la définition de toutes les classes et fonctions nécessaires ainsi qu'un programme principal « `main` » servant de test de plusieurs instanciations d'objets de ces classes et d'appels à leurs méthodes pour au moins deux types de base différents. Vous fournirez également un schéma, un diagramme « à la UML » de la hiérarchie des différentes classes définies.

L'évaluation portera essentiellement sur la pertinence des choix effectués dans l'écriture : la codification, la présentation et l'optimisation du programme justifiées par la maîtrise des mécanismes mis en œuvre lors de la compilation et l'exécution du code. D'une manière générale, le respect strict des directives, la concision, la précision, la lisibilité (clarté du texte source), l'efficacité (pas d'opérations inutiles ou inadéquates) et le juste choix des syntaxes typiques de C++ seront des critères essentiels d'appréciation. De brefs commentaires dans le code source sont souhaités pour éclairer les choix de codification.

Votre travail doit être réalisé pour le vendredi 21 décembre 2018 à 10 heures au plus tard. Vous remettrez tout votre travail empaqueté en un seul fichier compacté (« .zip » ou autre) *via* le site du cours (INFO-F202) sur l'Université Virtuelle (https ://uv.ulb.ac.be/). Ceux-ci devront contenir en commentaire vos matricule, nom et prénom. 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.