UNIVERSITÉ LIBRE DE BRUXELLES ULB

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Langages de programmation 2 : Projet C++

Yves Roggeman *

Année académique 2016–2017 (2e session)

Résumé

Ceci est un des deux énoncés de l'épreuve de seconde session qui se déroule en août. 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 simple ou multiple.

Le problème posé consiste à définir une structure de données — que nous appellerons *un dictionnaire* — constituée d'une famille de chaînes de caractères toutes distinctes et stockées par ordre lexicographique. Cette structure sera implantée concrètement de deux façons différentes. On réalisera également une fusion de deux telles structures, indépendamment de l'implantation choisie pour chacune.

1 Le dictionnaire

Le dictionnaire est un type abstrait de conteneur, c'est-à-dire un type dont les seules opérations sont ici : insérer une (nouvelle) chaîne, retirer une chaîne (existante) et rechercher si une chaîne est présente. Bien sûr, les opérations standard de copie et de transfert de ce conteneur seront également définies.

Toutes les formes concrètes d'un dictionnaire hériteront de ce type abstrait de base.

Trois implantations sont ici envisagées : comme une liste de chaînes et comme un *trie*, qui hériteront donc chacune également de l'un de ces types (c'est un héritage multiple), et une troisième stockant ses éléments dans un conteneur prédéfini « `std::vector` » (ce n'est pas un héritage).

Il est évident que les chaînes contenues dans un dictionnaire y sont stockées de manière unique. Elles ne peuvent y être dupliquées ni stockées autrement que dans la structure choisie pour l'implantation concrète ; c'est une règle essentielle en algorithmique ! À aucun moment, il ne peut donc être créé, par exemple, de tableau de chaînes supplémentaire ou toute autre forme de copie des données, même de manière temporaire.

1.1 Liste de chaînes

Un dictionnaire peut être implanté comme héritier d'une liste triée de chaînes de caractères. Ces chaînes sont ici de simples objets du type prédéfini `std::string`. Cette structure hérite d'un type générique (un canevas, un **`template`**) de liste simplement liée et triée dont le type de base des éléments est un paramètre.

Vous devez définir explicitement cette structure (*i.e.* ne pas utiliser un conteneur de la bibliothèque) avec pour seules opérations l'insertion d'un élément, la suppression, les opérations de parcours (`begin()`, `end()`, **`operator`**++ préfixe et suffixe) et l'accès à l'élément courant (**`operator`*** unaire). Vous définirez donc un itérateur permettant ce parcours et l'accès à un élément. Vous pouvez vous contenter d'adapter simplement l'exemple de liste fourni sur le site de cours.

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

1.2 Trie

Un *trie* est un arbre n-aire dont les nœuds contiennent un seul caractère, à l'exception de sa racine qui est vide. Chaque nœud peut donc représenter une chaîne correspondant à la suite des caractères du chemin de la racine à ce nœud. Chaque nœud, y compris la racine, contiendra donc également un booléen indiquant s'il correspond ou non à une chaîne à prendre en considération. Des nœuds frères sont triés par ordre croissant et contiennent toujours des caractères différents.

La façon la plus simple de réaliser un arbre n-aire est de le traduire comme une structure d'arbre binaire où le pointeur « gauche » mène au premier fils et le pointeur « droit » au frère (*i.e.* le fils suivant du même père). Cette implantation vous est imposée : vous ne pouvez définir de liste ou de tableau de fils, par exemple.

Comme l'on doit retrouver le chemin vers la racine depuis un nœud, chacun contiendra donc également un pointeur vers son père (au sens de l'arbre n-aire, bien sûr).

Vous implanterez directement cette structure à l'aide d'une classe « nœud » imbriquée et privée de la classe « trie » pour laquelle vous définirez les opérations nécessaires pour permettre son usage comme réalisation du dictionnaire.

1.3 Vecteur

Cette forme contient un attribut de type « `std::vector<std:string>` » et implante ses opérations à l'aide ce celle de ce conteneur de façon masquée.

1.4 Fusion

Vous écrirez un opérateur polymorphe « += » qui réalise une fusion de deux dictionnaires, quels que soient les types concrets de ses opérandes.

Cet opérateur doit donc accepter tant comme opérande de gauche que de droite un dictionnaire générique et doit donc être défini dès la classe abstraite de base. Dans tous les cas, il ne peut dupliquer les éléments de l'opérande de gauche ; il peut seulement y insérer ceux de l'opérande de droite, celui-ci restant inchangé.

2 Réalisation

Il vous est demandé d'écrire en `C++` la définition de toutes les classes nécessaires ainsi qu'un programme principal « `main` » servant de test de plusieurs instanciations d'objets de ces classes et d'appels à l'opérateur de fusion. Vous fournirez également un schéma, un diagramme « à la UML » de la hiérarchie des différentes classes définie.

L'évaluation portera le respect strict des structures décrites dans l'énoncé et 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, 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 mercredi 16 août 2017 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-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.