Projet de C++

2024-2025

Énoncé

On souhaite développer une application en C++ permettant de gérer un index d'éléments. Cet index est composé de nœuds, organisant des éléments selon une clé de recherche (i.e. une valeur à partir de laquelle les éléments indexés dans l'index peuvent être retrouvés).

Par exemple, la Figure 1 représente un index (représenté par le premier niveau en partant du haut) dont les nœuds (situés au deuxième niveau) ont pour clé de recherche une lettre (ici une lettre initiale) et dont les éléments (situés au dernier niveau) sont des chaînes de caractères (ici des prénoms par exemple). Le dernier nœud (situé à droite), par exemple, a deux éléments (le nombre d'éléments du nœud est indiqué en haut du nœud) et comme clé de recherche associée la lettre 'z'. Le deuxième élément (situé à droite) correspond à la chaîne de caractères "Zoé" (la clé de recherche 'z' est indiquée au dessus de la chaîne de caractères).

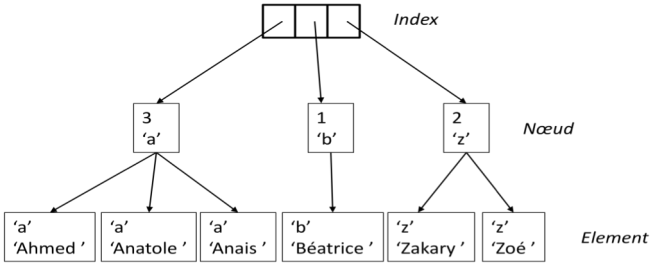

Figure 1 - Un index dont la clé de recherche est une lettre et dont les éléments sont des chaînes de caractères.

La Figure 2 représente un index dont les nœuds ont pour clé de recherche un entier (ici une note par exemple) et dont les éléments sont des chaînes de caractères (ici des prénoms correspondant aux prénoms des élèves ayant obtenus cette note par exemple). Le dernier nœud (situé à droite) contient donc les éléments correspondent aux prénoms des 2 élèves ayant obtenus la note 12.

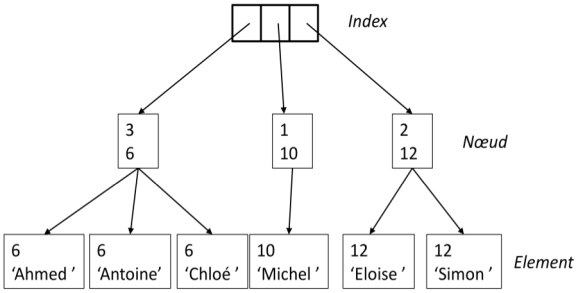

Figure 2 - Un index dont la clé de recherche est un entier (une note) et dont les éléments sont des prénoms.

La Figure 3 représente, quant à elle, un index dont les nœuds ont pour clé de recherche un entier (correspondant à une borne inférieure) et dont les éléments sont aussi des entiers. Le dernier nœud (situé à droite) contient les éléments correspondant aux 2 entiers indexés (i.e. enregistrés dans l'index) qui sont supérieurs à 200.

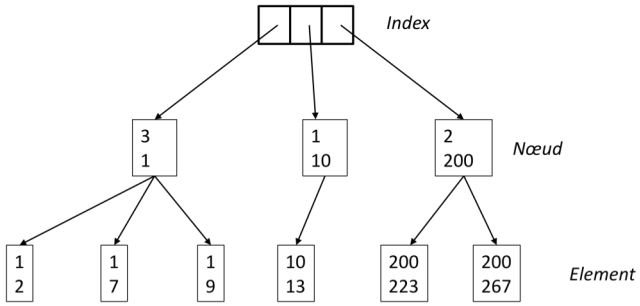

Figure 3 - Un index dont la clé de recherche est un entier et dont les éléments sont aussi des entiers.

L'objectif du projet est de définir des classes permettant de gérer ce type d'index. La Figure 4 donne le diagramme UML correspondant. Vous pouvez ajouter à ce modèle tous les attributs ou méthodes qui vous sembleraient pertinents pour le projet.

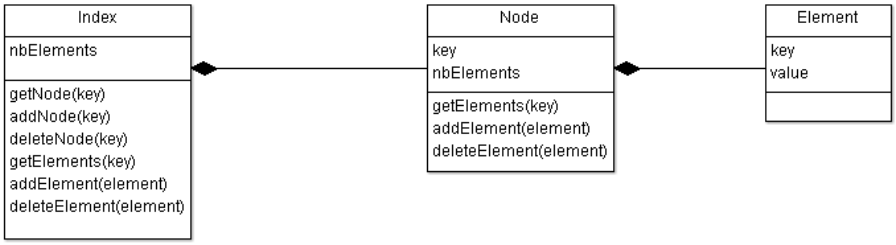

Figure 4 - Diagramme de classes UML de l'application.

L'application doit permettre :

1. De construire un index, tel que défini plus haut, à partir d'un fichier, donc les lignes sont de la forme : `key ; value`.

   Il est à noter que les types de clés de recherche et les types des valeurs pouvant varier, l'utilisation de patron de classes est vivement recommandée.

   Le nombre de nœuds et d'éléments étant dépendants du contenu du fichier en entrée, l'utilisation de pointeurs est également fortement recommandée, afin de n'utiliser que l'espace mémoire réellement nécessaire.

   Idéalement, les nœuds de l'index doivent être triés en fonction de leur clé d'index et les éléments d'un même nœud doivent être triés en fonction de leur valeur. L'algorithme de tri utilisé n'a aucune importance.

2. De pouvoir rechercher dans l'index, créé à l'étape 1, des nœuds (et leurs éléments associés) en fonction d'une certaine valeur de clé de recherche, ou des éléments en fonction de leur clé de recherche.

3. De pouvoir insérer ou supprimer des éléments dans l'index.

   Il est à noter que si l'élément ajouté a une clé de recherche qui ne correspond à aucun nœud, il faut créer ce nœud. De même, si un élément est supprimé et que le nœud correspondant se retrouve sans élément, il est nécessaire de supprimer le nœud.

Organisation du projet

Le projet est à effectuer **individuellement.**

Travail à réaliser

Vous devez développer l'application en C++. Votre application devra au minimum contenir les classes telles qu'elles sont décrites sur la Figure 4. Vous pouvez ajouter toute autre classe (à justifier dans le rapport final) de votre choix ou compléter les classes de la Figure.

Ce projet donnera lieu à une présentation power point**,** contenant au moins les parties suivantes :

- La description générale de vos classes C++ (attributs et signatures des méthodes, associées à un court descriptif de chaque méthode en français).
- Le descriptif détaillé des algorithmes[1] permettant de réaliser les fonctionnalités 1 à 3 mentionnées plus haut.
- Des exemples d'exécution commentés.
- Une conclusion décrivant les limites de votre travail, ainsi qu'une évaluation du temps passé.

Le code source de vos programmes (fichiers d'extension .h et .cpp uniquement) devra être remis le jour de la soutenance. Un programme test devra être fourni afin que le correcteur puisse tester la totalité de votre programme. Vos classes doivent fonctionner avec votre programme test ou tout autre programme test que le correcteur souhaitera développer.

La présentation et la soutenance sont programmés pour le 2/04 suivant l'ordre alphabétique. La démonstration ainsi que l'exposé devront se faire en 5 minutes

Un mini rapport devra être fourni à la même date qui résumera l'ensemble de votre réalisation. Tous les documents à savoir présentation, codes et rapport seront à déposer sur l'Eprel.

En cas de problème de compréhension ou d'ordre fonctionnel, n'hésitez pas à me contacter par mail (jalel.ben-othman@u-pec.fr).

[1] Merci si vous vous êtes inspiré de codes pris sur internet, de citer vos sources !