Automne 2024

Algorithmes et structures de données pour ingénieur

GLO-2100/ IFT-2008

Thierry EUDE, Ph.D
Kim Rioux-Paradis, M.Sc

Travail à faire INDIVIDUELLEMENT
À rendre avant la date indiquée
sur le portail du cours

(Voir modalités de remise à la fin de l'énoncé)

Ce travail doit être de votre propre création.
Tout étudiant qui commet une infraction au Règlement disciplinaire à l'intention des étudiants de l'Université Laval dans le cadre du présent cours, notamment en matière de plagiat, est passible des sanctions qui sont prévues dans ce règlement. Il est très important pour tout étudiant de prendre connaissance des articles 23 à 46 du Règlement disciplinaire. Celui-ci peut être consulté à l'adresse suivante:
http://ulaval.ca/reglement-disciplinaire
Tout étudiant est tenu de respecter les règles relatives au plagiat. De plus, les étudiants ou étudiantes ayant collaboré au plagiat seront soumis aux sanctions normalement prévues à cet effet par les règlements de l'Université.

Tous les travaux sont soumis à un logiciel de détection de plagiat.

Travail Pratique #3
Bottin téléphonique

UNIVERSITÉ LAVAL
Faculté des sciences et de génie
Département d'informatique
et de génie logiciel

Objectifs

Ce travail pratique a pour principal objectif de vous permettre d'approfondir l'exploitation d'autres types abstraits qui sont couramment utilisés pour la recherche et l'accès rapide aux données en l'occurrence de vous familiariser avec les tables de dispersion (*Hash Tables*).

Bottin téléphonique

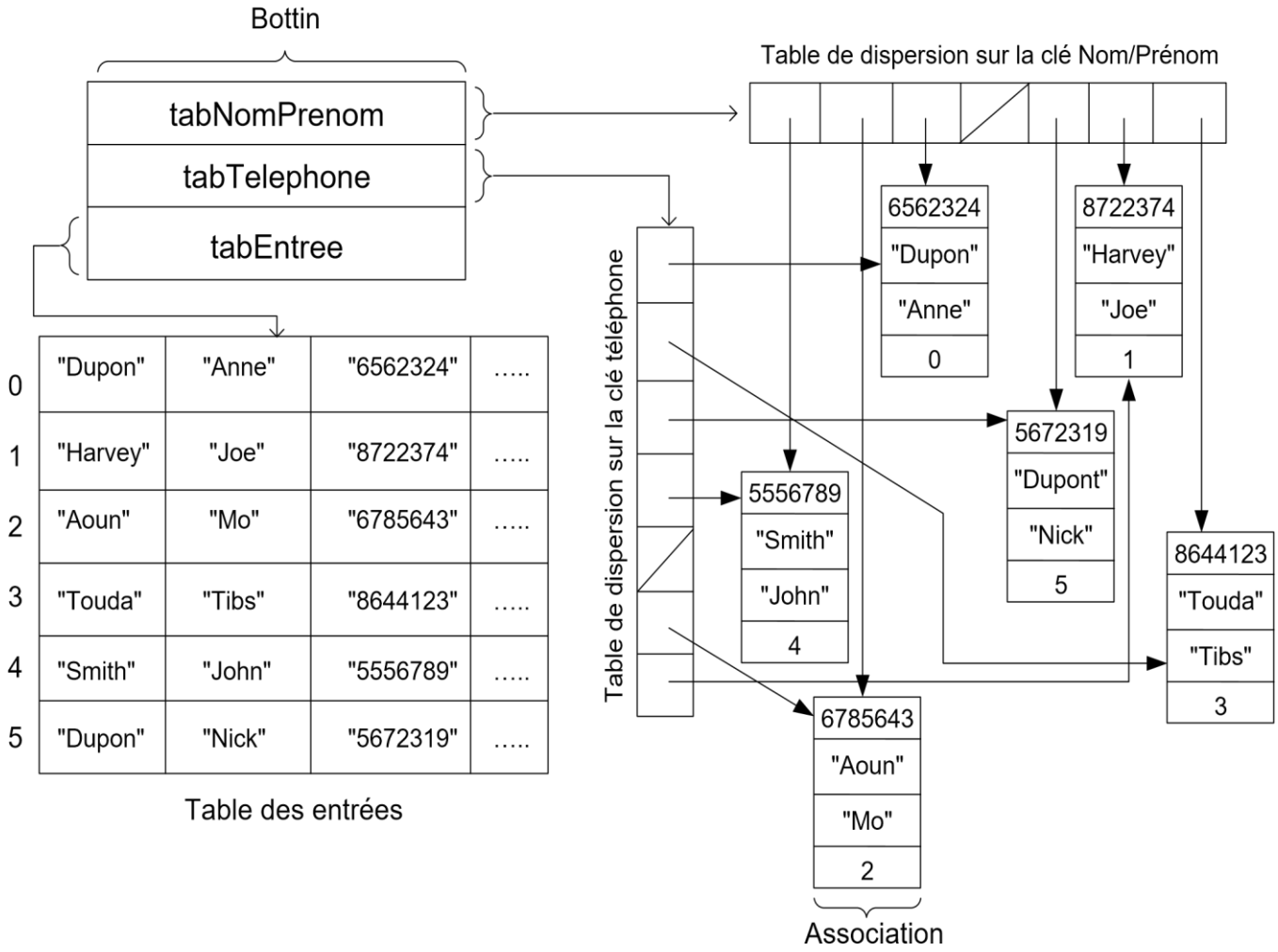
Votre mandat est de représenter l'information nécessaire à la gestion d'un bottin téléphonique. Cette information décrit un ensemble de membres du personnel dans une organisation, et pour chacun d'eux, un numéro de téléphone, un numéro de cellulaire, et une adresse courriel. Toutefois, afin de rechercher efficacement l'information pertinente à une personne, il vous faudra bâtir un index représenté par une table de dispersion dont les collisions devront être gérées par **adressage ouvert**.

Il s'agit précisément d'implanter un type abstrait **Bottin**. Ce type est constitué de plusieurs associations. Une association relie une clé primaire, par exemple une chaîne de caractère représentant un numéro de téléphone, à un nom et un prénom associés à ce numéro de téléphone, donc un couple **<clé, nom-prénom>**. Les clés d'un dictionnaire sont toutes uniques; les noms ne le sont pas (i.e. il peut y avoir des clés différentes qui ont un même nom associé, cependant, toute paire nom/prénom est unique et peut être considérée comme une seconde clé pour la recherche).

Nous désirons un accès le plus rapide possible pour trouver une association, soit par la clé numéro de téléphone, soit par la clé nom/prénom. Vous devez implanter les associations comme étant des encapsulations qui sont insérées dans deux tables de dispersion : l'une pour indexer les numéro de téléphone, l'autre, pour indexer les noms/prénoms. Ainsi, l'implantation ressemblera au graphique ici-bas. Une seule information, i.e., un entier, caractérise une association. Celui-ci désigne l'endroit dans un tableau des entrées où les informations complètes sur l'association en question peuvent être retrouvées.

I. Modèle d'implantation

Le bottin téléphonique qui sera chargé à partir du fichier texte (dont nous décrirons le format plus bas) sera maintenu dans la structure suivante :



Vous devez compléter la partie privée du type abstrait Bottin (voir **Bottin.h**) afin de refléter le modèle présenté dans le graphique précédent. Vous devez utiliser la fonction de dispersion (i.e. de hachage) implémentée dans le foncteur HString1 fourni (dans FoncteurHachage.hpp) ainsi que les politiques de résolution de collision nécessaires à la réalisation de cette implantation. Les contraintes imposées sont donc le respect du modèle graphique et la gestion des tables de dispersion qui doit être faite par **adressage ouvert**.

<u>Mises en garde</u> :

- Les tables de dispersion doivent avoir une taille adéquate.
- La fonction de hachage et de redispersions doit être celle fournie (HString1).
- Vous devez gérer les exceptions dans toutes les méthodes de la classe Bottin décrites ci-dessous.

Travail à réaliser

En plus des méthodes propres à la préoccupation de la programmation orientée objet (constructeurs, destructeur, etc.), vous devez implémenter les méthodes suivantes :

I. Fichier Bottin.cpp

Bottin (std::ifstream & p_fichierEntree, size_t p_table_size = 100)

constructeur avec fichier en entrée

Paramètres

p_fichierEntree

Précondition

le fichier est conforme au format attendu

Paramètres

p_table_size : taille des tables de hachages au départ.

Ce constructeur s'occupe de charger le bottin à partir d'un fichier en entrée. Ce constructeur doit permettre de connaitre le ratio de collisions rencontrées par rapport au nombre d'entrées, ceci dans chacune des 2 tables de dispersion. Vous pouvez utiliser le fichier **Bottin.txt** pour faire vos tests. Sur la première ligne de ce fichier figure le nombre de personnes contenues dans le bottin. Ensuite, sur les lignes subséquentes se retrouvent les personnes proprement dites, d'après le format suivant :

Nom, Prenom (666) 666-6666 (999) 999-9999 courriel

où (666) 666-6666 représente le numéro de téléphone fixe et (999) 999-9999 le numéro de téléphone cellulaire.

Dans le fichier que nous vous fournissons, les champs sont séparés par des tabulations ('\t'). Notez que les espaces ne constituent pas une délimitation de champ, mais que les noms et prénoms sont séparés par , + espace.

void Bottin::ajouter (const std::string & p_nom, const std::string & p_prenom,
const std::string & p_telephoneFixe, const std::string & p_cellulaire,
const std::string & p_courriel)

Ajoute une entrée dans le tableau des entrées et fait les associations dans les tables de hachages.

Paramètres

p_nom

p_prenom

p_telephoneFixe

p_cellulaire

p_courriel

Précondition

Les numeros de telephones sont valides

Les entrées ne sont pas vides.

La clef à insérer n'est pas déjà présente dans la table

Postcondition

La clef est insérée dans les deux tables et les données dans le tableau

const Bottin::Entree & Bottin::trouverAvecNomPrenom (const std::string & p_nom,
const std::string & p_prenom) const

Recherche une entrée dans **m_tableParNomPrenom** à partir du nom et du prénom.

Paramètres

p_nom

p_prenom

Précondition

Les entrées ne sont pas vides.

Renvoie

l'entrée trouvée, exception si non trouvé.

Cette méthode recherche le numéro de téléphone fixe, le numéro de téléphone cellulaire et l'adresse courriel d'une entrée identifiée par le nom et le prénom. Assumez que chaque paire de nom/prénom est unique. La méthode doit renvoyer toute l'entrée recherchée (y compris le nom et le prénom).

const Bottin::Entree & Bottin::trouverAvecTelephone(const std::string & p_telephoneFixe) const

Recherche une entrée dans **m_tableParTelephone** à partir du numéro de téléphone.

Paramètres

p_telephoneFixe

Précondition

L'entrée n'est pas vide.

Renvoie

l'entrée trouvée, exception si non trouvé.

Cette méthode recherche le numéro de téléphone fixe, le numéro de téléphone cellulaire et l'adresse courriel d'une entrée identifiée par le numéro de téléphone fixe. Assumez que chaque numéro de téléphone fixe est unique. La méthode doit renvoyer toute l'entrée recherchée (y compris le nom et le prénom).

void Bottin::afficherBottin (std::ostream & p_out) const

pour afficher toutes les entrées enregistrées

Paramètres

p_out un ostream contenant les lignes du bottin

int Bottin::nombreEntrees () const

Renvoie

retourne le nombre d'entrées.

double Bottin::ratioDeCollisionsNomPrenom () const

Renvoie

retourne le ratio nombre de collision / nombre d'insertions pour la table de hachage NomPrenom

double Bottin::ratioDeCollisionTelephone () const

Renvoie

retourne le ratio nombre de collision / nombre d'insertions pour la table de hachage avec telephone

int TP3::Bottin::maximumNbCollisionNomPrenom () const

Renvoie
retourne le nombre maximal de collision pour une insertion pour nom / prenom

int Bottin::maximumNbCollisionTelephone () const

Renvoie
retourne le nombre maximal de collision pour une insertion pour telephone

II. Fichier TableHachage.hpp

Vous devez modifiez ce fichier afin de compter

1) Le nombre de collisions. Le nombre de collisions doit être remis à 0 avant chaque rechaghage.
2) Le nombre d'insertions. Le nombre d'insertions doit être remis à 0 avant chaque rehachage
3) Le nombre maximal de collision pour une insertion.

Avertissements

Il est interdit de modifier le modèle d'implantation des classes tableHachage.h, Bottin.h vous avez le droit d'ajouter:

1. des méthodes privées aux deux classes afin d'optimiser votre implantation, c'est même très encouragé ;
2. des définitions de constantes;

Récapitulatif des fichiers fournis

1. Le fichier Bottin.txt contenant les données à lire sur les employés.
2. Le fichier Bottin.h avec les déclarations des méthodes publiques, à compléter.
3. Le fichier TableHachage.h à compléter.
4. Le fichier TableHachage.hpp à modifier.
5. Le fichier FoncteurHachage.hpp à ne pas modifier.

Bien livrable :

Le troisième travail pour le cours GLO-2100/IFT-2008 est **à faire INDIVIDUELLEMENT.**
Tous vos développements doivent être faits dans le dépôt github privé qui vous est créé après avoir accepté « l'assignment »
https://classroom.github.com/a/KIspisZC

RAPPEL : Vous DEVEZ ABSOLUMENT CHOISIR VOTRE COURRIEL dans la liste avant d'accepter l'assignement sinon, on ne pourra pas vous identifier lors de la correction et vous donner votre résultat.

On devra y retrouver :
Bottin.h, Bottin.cpp, TableHachage.h, TableHachage.hpp, FoncteurHachage.hpp, ContratException.h, ContratException.cpp, vos testeurs, le fichier de données test
ainsi tous les fichiers sources nécessaires à la compilation de votre projet.

Les normes de programmation

Vous devez respecter les normes de programmation du cours. Une description de ces normes se trouve dans *contenu et activités/Notes de cours/Documents divers* du site Web du cours.

Tolérance zéro vis-à-vis des erreurs de compilation

Si un travail comporte ne serait-ce qu'une erreur de compilation, il sera fortement pénalisé, et peut même se voir attribuer la note zéro systématiquement.

Erreurs à l'exécution

Testez intensivement vos programmes. Rappelez-vous qu'un programme qui ne plante pas n'est pas nécessairement sans bogue, mais qu'un programme qui plante même une seule fois comporte nécessairement un bogue. D'ailleurs, si un programme ne plante nulle part sauf sur l'ordinateur de votre ami, c'est qu'**il comporte un bogue**. Il risque donc de planter un jour ou l'autre sur votre ordinateur et, encore pire, sur celui des correcteurs.
Si vous ne testez pas suffisamment votre travail, il risque de provoquer des erreurs à l'exécution lors de la correction. La moindre erreur d'exécution rend la correction extrêmement difficile et vous en serez donc fortement pénalisés.

Portabilité des programmes

Vos programmes doivent absolument pouvoir être compilés et exécutés sans erreurs sur n'importe quel système d'exploitation. Autrement, vous obtiendrez la note zéro pour votre travail. Notamment, nous exigeons que les programmes puissent être compilés sous g++ avec la machine virtuelle du cours. Rappel : Il est donc important de vérifier la portabilité de votre code avec l'environnement de correction ceci avant la remise.

Critères d'évaluation

1) Respect des biens livrables
2) Portabilité de votre code source
3) Structure, organisation du code, lisibilité
4) Documentation
5) Exactitude du code (fonctionnalité)
6) La gestion dynamique de la mémoire.
7) La gestion des exceptions
8) La qualité de vos testeurs
9) Le respect des prototypes des méthodes quand ils sont imposés.
10) L'implémentation du contrat

 ***Particularités du barème***

- *Si des pénalités sont appliquées, elles le seront sur l'ensemble des points.*
- *Si un travail comporte ne serait-ce qu'une erreur de compilation, il sera fortement pénalisé, et peut même se voir attribuer la note zéro systématiquement.*
- *Il est très important que votre travail respecte strictement les consignes indiquées dans l'énoncé, en particulier les prototypes des fonctions, sous peine de fortes pénalités*

Plagiat

Tel que décrit dans le plan de cours, le plagiat est interdit. Une politique stricte de tolérance zéro est appliquée en tout temps et sous toutes circonstances. Tous les cas seront référés à la direction de la Faculté.

Bon travail