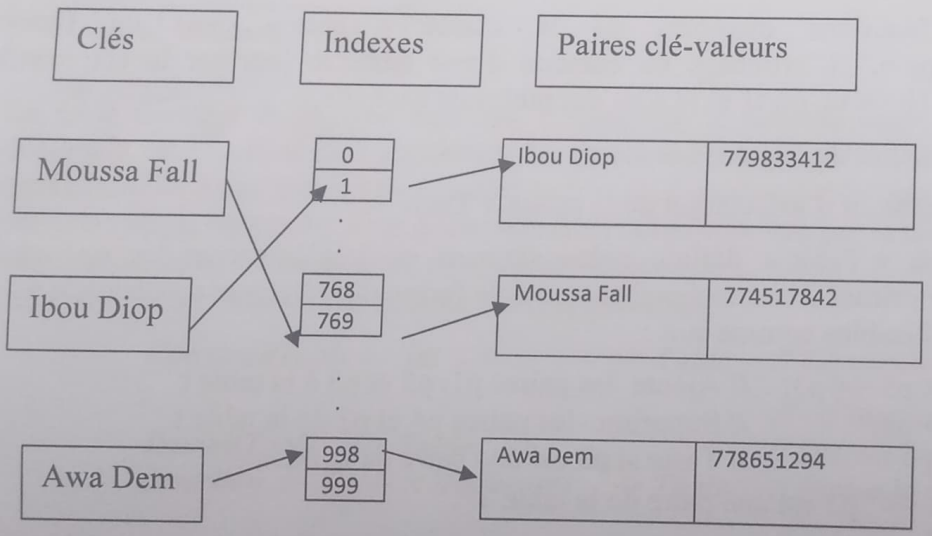
UNIVERSITE GASTON BERGER
DE SAINT-LOUIS
UFR SAT
SECTION INFORMATIQUE

ANNEE 2017-2018

Programmation Orientée Objet
Contrôle Continu
Licence 3 INFO
Durée : 2h
Documents non autorisés

Exercice 1

Une table de hachage est, en informatique, une structure de données qui permet une association clé–valeur, c'est-à-dire une implémentation du type abstrait tableau associatif ; en particulier, l'implémentation d'une table de symboles lorsque les clés sont des chaînes de caractères. Il s'agit d'un tableau ne comportant pas d'ordre (contrairement à un tableau ordinaire qui est indexé par des entiers). On accède à chaque valeur du tableau par sa clé. L'accès s'effectue par une fonction de hachage qui transforme la clé en une valeur de hachage (un nombre) indexant les éléments de la table, ces derniers étant appelés des buckets. En voici une illustration :



Ce schéma est un annuaire représenté comme une table de hachage. La fonction de hachage transforme les clés (à gauche) en valeurs de hachage (au milieu) indexant les éléments de la table composés de paires clé–valeur (à droite).
Ainsi, la table de hachage peut être vue comme un tableau de pointeurs sur des paires clé-valeurs dont les indexes (indices) sont obtenues par hachage des clés.
Quand la fonction de hachage renvoie le même nombre pour deux clés différentes, on dit qu'il y a une collision. Dans un premier temps, les collisions ne seront pas gérées. Autrement dit, dans un tel cas, la place sera considérée comme occupée et l'insertion impossible.
Les chaînes de caractères seront considérées au sens de c.
On veut implémenter une table de hachage avec les opérations suivantes :

ConstruitTable : **->** ***Table*** *// Construit une table de hachage vide*
DetruitTable : ***Table*** **->** *// Detruit la table de hachage*
insertion ***: Paire x Table*** **->** ***Table*** *// Insère la paire clé-valeur de clé c et*

```
                                                    // de valeur v dans la table si cette
                                                    // dernière ne s'y trouve pas
getV : Table x c          ->    V                   // Permet de récupérer la valeur d'une
                                                    // paire à partir d'une clé c de la table
supprimer : Table x c     ->    Table  // Retire la paire clé-valeur de clé c de
                                       // la table
rechercher : Table x c    ->    Booléen  // Recherche la paire de clé c
                                         // dans la table
afficher : Table          ->             // Affiche le contenu de la table
```

1. Définissez dans un fichier « TableHachage.h » une classe « Table » correspondant au type précisé ci-dessus.
   Indication : déclarer dans un premier temps une structure *Paire* pour les paires clés-valeur avant de définir la classe.
   La fonction de hachage à utiliser est la fonction suivante : SommeLettres%100, *où* SommeLettres *est la somme des codes ASCII des différentes lettres de la chaîne de caractères.*
2. Définissez les fonctions membres de la classe « Table » dans un fichier « TableHachage.cpp ». L'affichage du contenu d'une table se fera par la commande suivante : cout << t1 << t2, où t1 et t2 sont des tables de hachage.
3. Ajoutez un constructeur de recopie à la classe « Table ».
4. Surdéfinissez l'opérateur d'affectation de la classe « Table ».
5. Modifiez la classe « Table » définie précédemment en implémentant les opérations « insertion », « supprimer » et « rechercher » sous la forme d'opérateurs surdéfinis « << », « >> » et « % » utilisables comme suit :

```
t << p1 << p5 << p3;   // Ajoute les paires p1, p5 et p3 à la table t
t >> p4 >> p2;         // Supprime les paires p4 et p2 de la table t
if (t % p3)            // Teste si p3 est une paire de la table
    cout << "p3 est une paire de la table T";
```

6. Vous testerez votre implémentation du type abstrait « Table » en construisant et manipulant des tables de hachage dans le fichier « main.cpp ». Vous donnerez des situations d'invocation du constructeur de recopie et ferez l'affectation entre tables.
7. On veut maintenant gérer les collisions. Pour cela, on utilise maintenant une fonction de hachage-compression *h* qui va borner la valeur de hachage aux indices du tableau. Ainsi, l'indice d'un élément *i*, donné par la fonction *h*, est défini comme suit :

$$h(i) = h_c(h_h(i)) \text{ où}$$

$h_h(i) = 3*i+14$ *est la fonction de hachage et* $h_c(i) = i \% 10$ *la fonction de compression*
Lorsqu'on a une collision, à partir de l'indice, on parcourt le tableau jusqu'à trouver une case libre. Les indices parcourus peuvent être écrits sous la forme suivante, avec M la taille du tableau, h la fonction de hachage-compression, et i un indice qui varie de 0 à M :

indice = (h(k) + i) % M)

Réécrire la méthode d'insertion d'une paire dans le tableau.

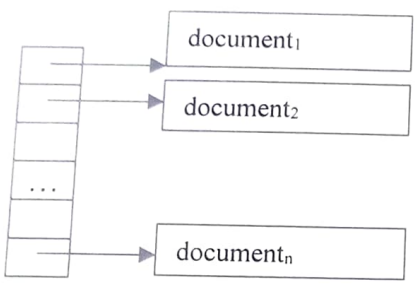
UNIVERSITE GASTON BERGER
DE SAINT-LOUIS
UFR SAT
SECTION INFORMATIQUE

ANNEE 2019-2020

Programmation Orientée Objet
Contrôle Continu
Licence 3 INFO
Durée : 2h
Documents non autorisés

Exercice 1

I. On considère une table représentée comme suit :



La table contient le nombre maximal d'éléments, le nombre courant et les éléments. Un document est constitué de 3 attributs : une clé (un entier) qui permet de l'identifier, une valeur (une chaîne de caractères correspondant au titre du document) et la liste de ses mots-clés. Ainsi, par exemple, la recherche d'un document dans la table se fera par la clé.
Les chaînes de caractères seront considérées au sens de c.
Les opérations possibles sur la table sont les suivantes :

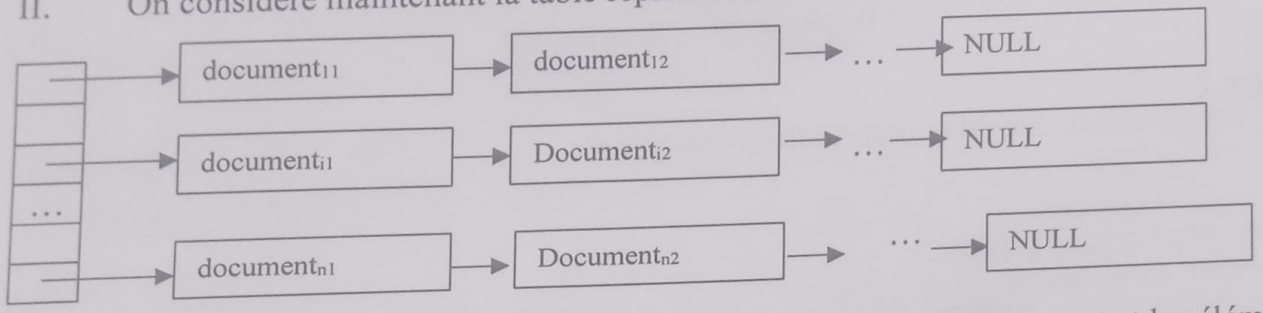
ConstruitTable : int	***-> Table***	*// Construit une table vide // d'au plus max éléments*
DetruitTable : Table ->		*// Detruit une table*
insertion : Table x Document	***-> Table***	*// Insère le document de clé i et de // valeur v dans la table si ce // dernier ne s'y trouve pas*
supprimer : Table x i	***-> Table***	*// Retire le document de clé i de // la table*
rechercher : Table x i	***-> Booléen***	*// Recherche le document de clé i // dans la table*

1. Définissez dans un fichier « Table.h » une classe « Table » correspondant au type précisé ci-dessus.
2. Définissez les fonctions membres de la classe « Table » dans un fichier « Table.cpp ». L'affichage du contenu d'une table se fera au format « d1 -> d2 -> d3 » si d1, d2 et d3 sont les documents ajoutés successivement.
3. Ajoutez un constructeur de recopie à la classe « Table ».
4. Surdéfinissez l'opérateur d'affectation de la classe « Table ».

5. Modifiez la classe « Table » définie précédemment en implémentant les opérations « insertion », « supprimer » et « rechercher » sous la forme d'opérateurs surdéfinis « << », « >> » et « % » utilisables comme suit :

```
t << d1 << d5 << d3;    // Ajoute les documents d1, d5 et d3 à la table t
t >> d4 >> d2;          // Supprime les documents d4 et d2 de la table t
if (t % d3)             // Teste si d3 est un document appartenant à la table t
        cout << "d3 est un document de la table t";
```

II. On considère maintenant la table représentée comme suit :



Liste chaînée i

La table contient toujours le nombre maximal d'éléments, le nombre courant et les éléments

1. Définissez dans un fichier « Table.h » une classe « Table » correspondant au type précisé ci-dessus.
2. Définissez les méthodes « ConstruitTable », « DetruitTable », « insertion » et « rechercher ». Pour l'ajout comme pour la recherche, le numéro de la liste chaînée utilisée est obtenu à partir de la clé par l'opération suivante : clé % max, où max est le nombre maximal d'éléments du tableau. L'ajout se fait en tête.
3. Vous testerez votre implémentation du type abstrait « Table » en construisant et manipulant des tables de documents dans le fichier « main.cpp ».

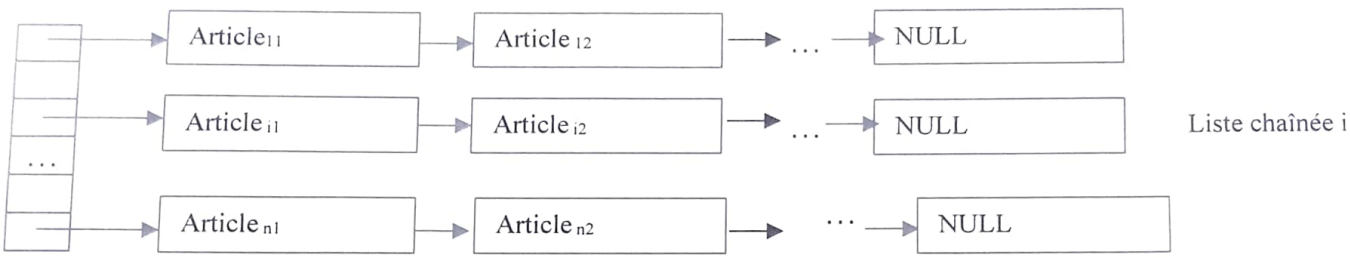
UNIVERSITE GASTON BERGER
DE SAINT-LOUIS
UFR SAT
SECTION INFORMATIQUE

ANNEE 2020-2021

Programmation Orientée Objet
Contrôle Continu
Licence 3 INFO – L3 MIAGE
Durée : 2h
Documents non autorisés

Exercice

On considère la table représentée comme suit :



La table contient le nombre maximal d'éléments, le nombre courant et les éléments.
Un article est constitué des attributs suivants : une clé (une chaîne de caractères) qui permet de l'identifier, sa valeur représentée par les caractéristiques suivantes : son url (une chaîne de caractères), le titre, l'auteur et la liste de ses mots-clés.
Les chaînes de caractères seront considérées au sens de c.
Les opérations possibles sur la table sont les suivantes :

ConstruitTable : int -> Table *// Construit une table vide*
// d'au plus max éléments
DetruitTable : Table -> *// Detruit une table*
insertion : Table x Article -> Table *// Insère l'article de clé i et de*
// valeur v dans la table si ce
// dernier ne s'y trouve pas
supprimer : Table x i -> Table *// Retire l'article de clé i de*
// la table
rechercher : Table x i -> Booléen *// Recherche l'article de clé i*
// dans la table

1. Définissez dans un fichier « Table.h » une classe « Table » correspondant au type précisé ci-dessus.
2. Définissez les fonctions membres de la classe « Table » dans un fichier « Table.cpp ». Pour l'insertion, la recherche comme pour la suppression, le numéro de la liste chaînée utilisée est obtenu à partir de la clé par l'opération suivante : SommeLettres%max, *où* SommeLettres *est la somme des codes ASCII des différentes lettres de la chaîne de caractères*, et max est le nombre maximal d'éléments du tableau. L'ajout se fait en queue.

3. L'affichage du contenu d'une table se fera au format « a1 -> a2 -> a3 » si a1, a2 et a3 sont les articles ajoutés successivement.
4. Ajoutez un constructeur de recopie à la classe « Table ».
5. Surdéfinissez l'opérateur d'affectation de la classe « Table ».
6. Modifiez la classe « Table » définie précédemment en implémentant les opérations « insertion », « supprimer » et « rechercher » sous la forme d'opérateurs surdéfinis « << », « >> » et « % » utilisables comme suit :
7.
```
t << a1 << a5 << a3;          // Ajoute les articles d1, d5 et d3 à la table t
t >> a4 >> a2;                // Supprime les articles a4 et a2 de la table t
if (t % a3)                   // Teste si a3 est un article appartenant à la table t
    cout << "a3 est un article de la table t";
```
8. Vous testerez votre implémentation du type abstrait « Table » en construisant et manipulant des tables d'articles dans le fichier « main.cpp ».