TP C++ n°3

A.Spécifications/Conception

Ce TP avait pour but d'implémenter un commande console permettant d'analyser un fichier de Log Apache. Ce fichier comporte des lignes de log dans lesquelles on retrouve un certain nombre d'information comme le fichier consulté, la personne qui l'a consulté, etc.

Notre application devait permettre d'afficher les 10 ressources les plus demandées, mais également générer un graphe des fichiers consultés. L'application devait être paramétrable à l'exécution, on peut choisir de générer ou non le fichier de graphe, d'exclure certains fichiers (images,etc.), ou encore d'afficher le classement en focntion de l'heure.

Nous avons essayer de rendre les classes qui constituent notre application réutilisable, notamment les classe Date, LogLine et LogParser.

1 - Spécifications

	Cas Normaux	Cas limite	Cas d'erreur
.log	trouvé et ouverture sans problème		pas trouvé ou erreur à l'ouverture, on retourne un message
.dot	inexistant. On le créé	Il existe, on le supprime et on en créé un	Impossible de le supprimer, renvoie un message d'erreur
type de requête	On les compte toutes		
return code HTTP	2XY (X et Y chiffres)	≠ 2XY (X et Y chiffres)	
10 docs les plus consultés	nb doc >10	- le 10ème et le 11ème sont égalités on renvoie le 10ème - nb doc <10, on donne ceux qu'on a	
Heure	heure < 24 heure ≥ 0	On ne traite pas les autres	

2 - Conception

Afin de réaliser ce TP, nous avons défini plusieurs classe.

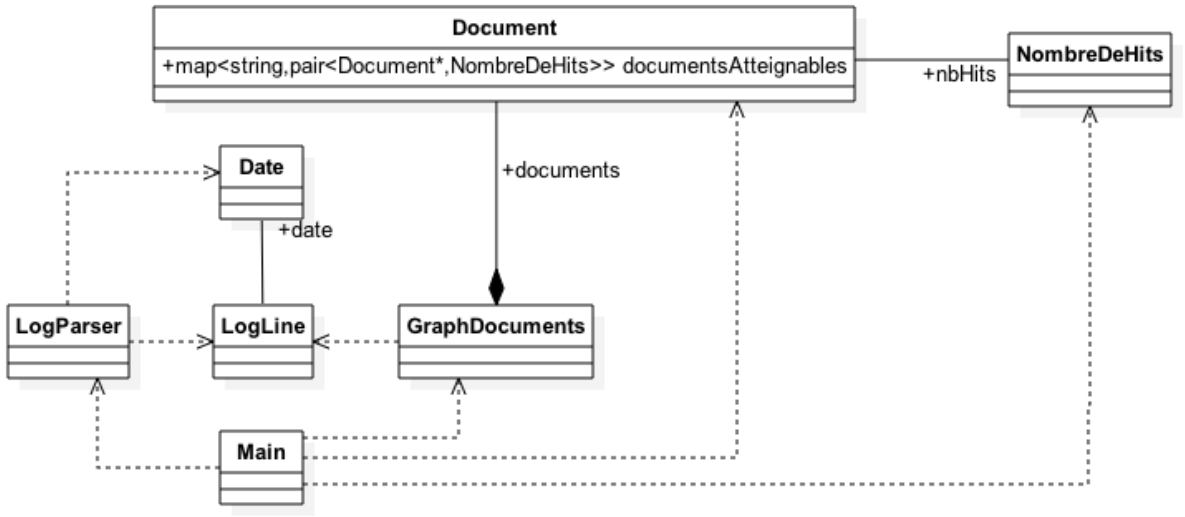


Schéma général de l'application

Classe Date :

Classe permettant de stocker une date.

Attributs :

- **int annee** : Valeur de l'année.
- **int mois** : Valeur du mois.
- **int jour** : Valeur du jour.
- **int heure** : Valeur de l'heure.
- **int minute** : Valeur de la minute.
- **int seconde** : Valeur de la seconde.
- **int gmt** : Valeur de la différence de l'heure avec celle de Greenwich.

Méthodes :

- Constructeurs (dont le constructeur de copie).
- Destructeur.
- Redéfinition de l'opérateur '**=**' pour l'affectation à partir d'une autre Date.
- Redéfinition de l'opérateur '**<<**' pour afficher la date avec un ostream

Classe NombreDeHits :

Classe permettant de stocker un nombre de hits par heure et en fonction d'un statut (réussi ou echoué).

Attributs :

- **static const int NB_HEURE_PAR_JOUR = 24** : Variable de classe contenant le nombre d'heure par jour.
- **int NombreDeHitsReussisParHeure[NB_HEURE_PAR_JOUR]** : Tableau contenant le nombre de hits réussis par heure.
- **int NombreDeHitsEchouesParHeure[NB_HEURE_PAR_JOUR]** : Tableau contenant le nombre de hits échoués par heure.

Méthodes :

- Constructeurs (dont le constructeur de copie).
- Destructeur.
- Redéfinition de l'opérateur '**=**' pour l'affectation à partir d'un autre NombreDeHits.

- **int NombreDeHitsPourUneHeure (int uneHeure, bool UniquementReussi = true)** : Méthode permettant de récupérer le nombre de hits pour une heure donnée. UniquementReussi = true si on veut uniquement le nombre de hits réussi, sinon on a le nombre total.
- **int NombreDeHitsTotal(bool UniquementReussi = true) const** : Méthode permettant de récupérer le nombre de hits. UniquementReussi = true si on veut uniquement le nombre de hits réussi, sinon on a le nombre total.
- **void MAJHits (bool status, int heure)** : Méthode permettant d'incrémenter le nombre de hits en fonction d'un statut (réussi : true, échec : false) et d'une heure. Cette méthode ne fait rien si heure < 0 et heure ≥ 24

Classe Document :

Classe permettant de stocker les informations d'un document consulté sur un serveur Apache. Permet aussi de connaître les documents atteignable à partir de celui-ci et de connaître le nombre de hits à partir de ce document;

Attributs :

- **string nomDomaine** : Nom de domaine du serveur où se trouve le document.
- **string cheminAccesRessource** : Chemin d'accès à la ressource sur le serveur (de la forme /temps/4IF15.html).
- **NombreDeHits nbHits** : Nombre d'accès (hits) à la page.
- **map<string,pair<Document*, NombreDeHits>> documentsAtteignables** : Map contenant les paires documents accessible depuis celui-ci et le nombre de hits à partir de ce document (valeur) et le chemin d'accès au document atteignable (clé) de type string.

Méthodes :

- Constructeurs (dont le constructeur de copie).
- Destructeur.
- Redéfinitions des opérateurs de comparaison qui comparent les deux documents par rapport au chemin d'accès de la ressource.
- Redéfinition de l'opérateur '**=**' pour l'affectation à partir d'un autre Document.
- **void MAJHits (int status, int heure)** : Méthode permettant de mettre à jour le nombre de hits du document en fonction du code de retour de la requête d'une ligne de log.
- **static bool CompareParNombreDeHitsReussis(Document* a, Document* b)** : Méthode de classe permettant de comparer deux documents en fonction du nombre de hits. Retourne true si le nombre de hits réussis de a est superieur au nombre de hits réussis de b, false sinon.
- **void MAJDocAtteignable(int status, int heure, Document* unDoc)** : Méthode permettant de mettre à jour les documents atteignables depuis ce document.
- **int NombreDeHitsAPartirDeCeDocument (bool uniquementReussis = true)** : Méthode permettant de récupérer le nombre de hits réussis (uniquementReussis = true) ou le nombre totals de hits (uniquementReussis = false) provenant de ce document.
- **const map<string, pair<Document*, NombreDeHits>> & DocumentsAtteignables() const** : Accesseur en lecture sur l'attribut documentsAtteignable.
- **const NombreDeHits & NbHits() const** : Accesseur en lecture sur l'attribut nbHits.
- **const string & CheminAccesRessource() const** : Accesseur en lecture sur l'attribut cheminAccesRessource.

Classe GraphDocuments :

Classe permettant de définir un graph de documents.
Attributs :

- **vector<Document*> documents** : vecteur contenant les documents du graph
- **string nomDomaine** : nom de domaine du serveur correspondant au graph.

Méthodes :

- Constructeurs (dont le constructeur de copie).
- Destructeur.
- Redéfinition de l'opérateur '**=**' pour l'affectation à partir d'une autre GraphDocuments.
- **void TraiterLogLine (LogLine l)** : Méthode permettant de traiter une ligne de log. Ajoute et/ou met a jour les documents en fonction de la ligne de log.
- **void TrierParNombreDeHitsReussis ()** : Méthode permettant de trier le vecteur 'documents' en fonction du nombre total de hits réussis de manière décroissante.
- **const vector<Document*> & Documents() const** : Accesseur en lecture sur l'attribut document.
- **const string & NomDomaine() const** : Accesseur en lecture sur l'attribut nomDomaine.
- **Document * DocumentPresent(string nomDomaine, string cheminAccesFichier)** : Méthode permettant de définir si un document ayant pour nom de domaine 'nomDomaine' et comme chemin d'accès 'cheminAccesFichier' est déjà présent dans le vecteur documents.

Structure LogLine :

Structure permettant de créer un objet regroupant tous les champs d'une ligne de log générée par un serveur Apache. Chaque attribut de la structure correspond un d'une champ de la ligne de log.
Attributs :

- **string clientIP** : Adresse IP du client émetteur de la requête.
- **string userLogName** : User Log Name, soit le nom d'utilisateur du visiteur.
- **string authenticatedUser** : Nom d'utilisateur que l'internaute s'est donné lui-même.
- **Date date** : Date et Heure de la requête avec GMT.
- **string actionType** : Type d'action exécutée (GET, POST, OPTIONS, …).
- **string requestedURL** : URL demandée, relative à l'URL stockée dans referer.
- **string httpVersion** : Version du protocole HTTP.
- **string status** : Code de retour de la réponse du serveur.
- **string bytesNumber** : Nombre d'octets de la réponse.
- **string domainName** : Nom de domaine de la source.
- **string sourceFile** : Nom du fichier de la source.
- **string navigator** : Identification du client navigateur.

Méthodes :

- Constructeurs (dont le constructeur de copie).
- Destructeur.
- Redéfinitions des opérateurs de comparaison qui comparent les deux documents par rapport au chemin d'accès de la ressource.
- Redéfinition de l'opérateur '**=**' pour l'affectation à partir d'une autre LogLine.
- **void Afficher ( )** : Méthode permettant d'afficher tous les champs de la ligne de log.

Classe LogParser :

Classe permettant de créer des objets de type LogLine. Cette classe ne contient que des méthodes de classes. Nous avons donc interdit de construire un objet de ce type. Cette classe s'utilise comme un outil.

Méthode :

- Constructeurs (dont le constructeur de copie) interdits.
- Destructeur interdit.
- Interdiction de l'opérateur '**=**' pour l'affectation à partir d'une autre LogParser.
- **static LogLine Parser (string line , string domainName)** : Méthode permettant de créer un objet de type LogLine à partir d'une string contenant une ligne de log d'un fichier de log Apache et à partir du nom de domaine du serveur apache générant le fichier
  Contrat : Il faut que la ligne donnée en paramètre soit valide. Sinon l'objet LogLine retourné ne sera pas valide.
- **static string Extension (string cheminAcces)** : Méthode de classe permettant de récupérer l'extension d'un fichier a partir d'une string contenant son chemin d'accès.
- **static Date ParserDate (string date)** : Methode permettant de créer un objet de type Date à partir d'une string contenant le champs date d'une ligne de log d'un fichier de log Apache.
- **static int Mois (string mois)** : Méthode permettant de convertir un mois anglais (sur 3 lettres, ex :Sep) en int. Retourne -1 si mois inconnu.
- **static string SplitReferer (string referer, string domainName)** : Méthode permettant de récupérer le chemin d'accès d'un fichier d'une url (referer) contenant le nom de domaine donne en paramètre. Retourne "-" si l'url (referer) ne contient pas le nom de domaine.

Main :

Constantes :

- **static const int NB_DOC = 10** : Nombre de document a afficher pour la commande par défaut.

Fonctions :

- **int main (int argc, char* argv[])** : Fonction main permettant d'éxécuter le programme. Cette fonction teste les arguments de la commandes et execute l'action appropriée.
- **void defaut (GraphDocuments & graph, int n , bool uniquementReussis = true)** : Fonction permettant d'afficher les NB_DOC les plus consultes d'un GraphDocuments
- **void creerFichierGraphe (const GraphDocuments & graph, const string & nomFichierGraphe , bool uniquementReussis = true)** : Fonction permettant de créer un fichier a partir GraphViz d'un GraphDocuments
- **void initialiserGraphe (GraphDocuments & graph, const string & fichierLog, bool exclusion, int heure)** : Fonction permettant d'initialiser un Graphdocuments a partir d'un fichier.
- **bool exclure (const LogLine & l , const vector<string> & v)** : Fonction permettant d'analyser si le fichier consultr dans une ligne de log
- **vector<string> extensionsExclus ( )** : Fonction retournant un vecteur contenant les extensions des fichiers à exclure.

3 - Données

Afin de ne pas avoir à re-développer des structures de données existantes et en accord avec le sujet, nous avons utilisé uniquement des structures de données provenant de la STL.

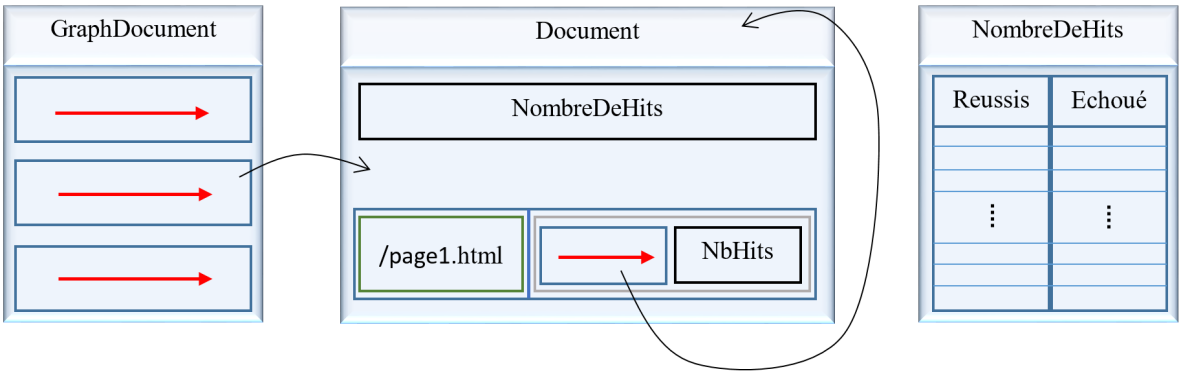
En effet, pour la classe GraphDocument, nous avons choisis de stocker tous les documents dans un vector<Document*> ,les documents étant créé par la classe GraphDocument à l'aide de la classe LogLine. Le vector permettant l'implémentation simple d'un trie grâce à l'algorithme std::sort qui provient de la STL.

Toutefois, nous aurions préféré utilisé un set<Document*, DocumentPtrComp> qui aurais permis un accès plus rapide aux documents lorsqu'on les recherche. DocumentPtrComp étant une classe de comparaison, sur l'attribut cheminAccesRessource du document, permettait de définir comment le set range ses éléments. Lors de l'implémentation du set nous nous sommes rendus compte que 2 éléments sur les 100 000 n'étaient pas pris en compte, nous avons donc laisser de côté cette structure de donnée.

La classe Document possède un attribut de type NombreDeHits, qui correspond à une structure contenant 2 tableaux d'entier de 24 cases chacun (une case par heure), nous donnant le nombre de fois où on a accédé à ce document, sans prendre en compte d'où l'on venait.

La classe Document possède également un attribut de type map<string,pair(Document*,NombreDeHits)>. La map, étant également une structure de donnée de la STL, permet un accès rapide en lecture grâce à une clef qui est ici l'attribut cheminAccesRessource du document cible (qui est le premier argument de la paire). le NombreDeHits étant le second, nous donne le nombre de fois où l'on a accédé à ce même document cible mais uniquement depuis le document dans lequel nous nous situons.

En réalité, cette clef string est inutile, on aurait très bien pu utiliser une map<Document*, NombreDeHits> étant donné que le Document* nous est donné depuis la classe GraphDocument. Mais notre map était alors triée sur l'adresse du document. Ainsi, lors de la création du fichier graph, nous avions exactement les mêmes information entre la première et la deuxième exécution mais pas forcément dans le même ordre (les pointeurs sur Document n'avaient aucune raison de garder la même adresse). Nous avons donc fait le choix de trier la map sur un critère qui ne dépend pas de l'exécution.



vector<Document> NombreDeHits et map<string, pair(Document*,NombreDeHits)>*

Tests

Nous avons effectué des testes fonctionnels en utilisant le framework de tests donnés. Chaque test teste une commande possible. (options, etc.)