Rapport TP4 C++

3IF 2019-2020

Application analog

INSA Lyon

Enzo Boscher

Tuoyuan Tan

[Conception et spécification 1](#_Toc30517180)

[Gestion des erreurs 2](#_Toc30517181)

# Conception et spécification

1. Schéma de données

|  |  |
| --- | --- |
| Doc demandé A |  |
| Doc demandé B |  |
| Doc demandé C |  |
| Doc demandé D |  |
| Doc demandé … |  |

Catalogue

|  |  |
| --- | --- |
| Referer A |  |
| Referer B |  |
| Referer C |  |
| Referer D |  |
| Referer … |  |

Association

|  |
| --- |
| Requête A |
| Requête B |
| Requête C |
| Requête D |
| Requête … |

Requêtes

logfile

Le programme fourni a deux fonctionnalités principales. Par défaut, il affiche les 10 documents les plus consultés. De plus, il fournira une description textuelle d’un graphe à partir du fichier de log quand l’utilisateur choisit l’option -g. Pour cette dernière fonctionnalité, il faut afficher tout d’abord tous les nœuds existant dans le fichier original et puis les relations de visite entre des nœuds.

Pour faciliter le stockage des relations entre ces nœuds, on décide de créer un couple. Le premier élément est un nœud (un document demandé), le deuxième est une autre structure qui permet de stocker tous les référenceurs qui ont visité ce document (pour les nœuds qui n’ont jamais été visités, cette structure sera vide). Chaque couple doit avoir une et une seule place dans notre conteneur. Donc on a besoin d’un conteneur qui contient des paires et qui assure que chaque nœud est unique. De plus, l’insertion dans ce conteneur doit être assez rapide. Par conséquent, on choisit un **map** pour stocker des paires qui prennent le document demandé comme la clé.

Ensuite, il faut trouver une structure qui permet de stocker les référenceurs qui référent sur un même document. A priori, chacune ligne du fichier log est transformée en une **Requête**, ce dernier est un objet que l’on définit. Même s’il suffit juste de stocker le nombre de visites entre deux nœuds, on décide de stocker toutes les **Requêtes** dans notre structure pour faciliter l’implémentation de nouvelles fonctionnalités dans le futur. Pour cela, on crée de nouveau un couple. Cette fois-ci, le premier élément devient un référenceur et le deuxième devient une autre structure qui stocke tous les **Requêtes** qui témoignent de la relation entre ce référenceur et le document demandé. En comptant le nombre de ces **Requêtes,** on obtient le nombre de visites de ce référenceur au document demandé. De même, on a besoin d’un conteneur qui contient des paires et qui assure que chaque référenceur est unique. Donc on choisit un **map** en prenant le nom de référenceur comme la clé.

Enfin, on a besoin d’un conteneur qui stocke les **Requêtes**. En effet, ce conteneur ne sert à rien, il n’est pas utilisé dans la suite. Il faut juste choisir un conteneur qui permet d’insérer une **Requête** à la fin très rapidement. Donc on choisit un **vector**. Il peut y avoir plusieurs solutions possibles, on le choisit juste parce que l’on connait très bien cette structure-là.

1. Cas normaux

|  |  |  |  |
| --- | --- | --- | --- |
| Numéro cas | Description | Entrées | Sorties attendues |
| 0 | Utilisation standard sans option | ./analog <filename.log> | Top10 des objets les plus demandés |
| 1 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Gestion des erreurs