Rapport Projet SE3

Sujet : Application d'Analyse de vols

Université de Lille

POLYTECH LILLE

SOMMAIRE 2

I. Présentation du sujet

Ce projet a pour objectif la réalisation d'une application qui permet d'analyser 58592 vols aux États Unis en 2014 dans le but de charger les informations des fichiers pour effectuer un certain nombre de requêtes par exemple lister des vols selon plusieurs critères, lister l'aéro ligne avec plus de retards ou bien trouver un itinéraire .
Cette année , pendant les séances de programmation avancée, nous avons travaillé sur les structures de données en langage C. Ce projet consiste à charger des informations depuis des fichiers CSV ,ces informations sont sous forme des données de 58592 vols aux États Unis en 2014 , ces données se répartissent en trois fichiers CSV . Après le chargement de données nous allons réaliser des fonctions qui traitent les données collectées , ces fonctions vont nous permettre d'obtenir des informations sur les vols. Ce projet peut être divisée en trois parties :
• La lecture des données et les mettre dans des structures.
• L'implémentation des fonctions permettant le traitement des données .
• Mise en œuvre d'une application de commande permettant l'accès aux informations souhaitées .

II. Description du cahier des charges

Dans ce projet il est demandé de réaliser un programme qui charge les fichiers de données CSV qui sont disponibles dans le répertoire data et qui permet de les interroger facilement.

À cause du volume important de données, un soin particulier devra être apporté à l'implémentation aux structures de données utilisées , de plus à la rapidité d'exécution du programme.

Le programme devra fonctionner de la manière suivante :

1. charger le fichier de données
2. attendre une commande
3. traiter la commande
4. afficher le résultat de cette commande
5. revenir à l'étape 2

Les commandes sont les suivantes:

- **show-airports <airline_id>** : affiche tous les aéroports depuis lesquels la compagnie aérienne <airline_id> opère des vols
- **show-airlines <port_id>**: affiche les compagnies aériens qui ont des vols qui partent de l'aéroport <port_id>
- **show-flights <port_id> <date> [<time>] [limit=<xx>]** : affiche les vols qui partent de l'aéroport à la date, avec optionnellement une heure de début, et limité à xx vols

- **most-delayed-flights** : donne les 5 vols qui ont subis les plus longs retards à l'arrivée
- **most-delayed-airlines** : donne les 5 compagnies aériennes qui ont, en moyenne, le plus de retards
- **delayed-airline <airline_id>** : donne le retard moyen de la compagnie aérienne passée en paramètre
- **most-delayed-airlines-at-airport <airport_id>** : donne les 3 compagnies aériennes avec le plus de retard d'arrivé à l'aéroport passée en paramètre
- **changed-flights <date>** : les vols annulés ou déviés à la date (format M-D)
- **avg-flight-duration <port_id> <port_id>**: calcule le temps de vol moyen entre deux aéroports
- **find-itinerary <port_id> <port_id> <date> [<time>] [limit=<xx>]** : trouve un ou plusieurs itinéraires entre deux aéroports à une date donnée (l'heure est optionnel, requête peut être limité à xx propositions, il peut y avoir des escales)
- **find-multicity-itinerary <port_id_depart> <port_id_dest1> <date> [<time>] <port_id_dest2> <date> [<time>] ... <port_id_destN> <date> [<time>]** : trouve un itinéraire multiville qui permet de visiter plusieurs villes (il peut y avoir des escales pour chaque vol intermediaire)
- **quit** : quit

1) Objectifs du projet

Ce projet a pour objectif la réalisation d'une application qui permet d'analyser 58592 vols aux États Unis en 2014 dans le but de charger les informations des fichiers pour effectuer un certain nombre de requêtes par exemple lister des vols selon plusieurs critères, lister l'aéro ligne avec plus de retards ou bien trouver un itinéraire .

III. Solutions techniques envisageables et choix de développement :

1) Choix de structures de données

Pour recueillir les données, j'ai créé trois structures à savoir: s_flight, s_airport et s_airlines, elles sont un assemblage de variables de différents types. À partir des fichiers csv à ma disposition, j'ai pu remplir les structures avec les données décrites dans les fichiers, ce qui m'a permis de stocker les informations pour une future utilisation, manipuler les informations avec aisance et extraire d'autres informations en effectuant quelques requêtes sur la combinaison de ces structures.

Chaque structure contient des informations différentes des autres et ça se peut qu'une information se répète dans une autre structure par exemple (iata_code qui se trouve dans la structure s_airport et s_airline) .

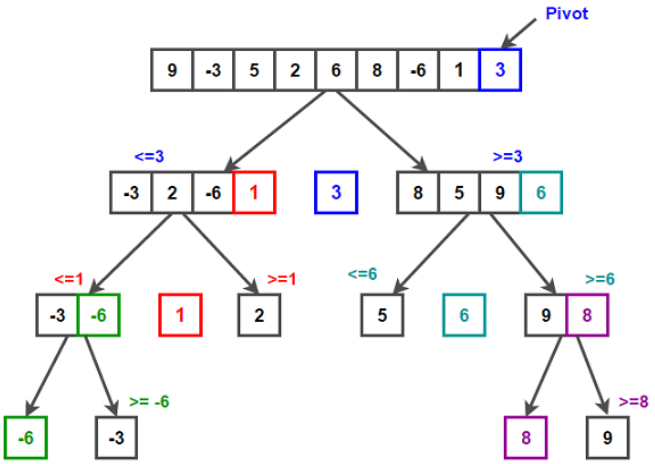
De plus j'ai défini un pointeur de fonction dans chaque structure , ce dernier (**to_string**) est un pointeur qui pointe sur une fonction (**nom_de_structure**_to_string) qui convertit une structure donnée à une chaîne de caractère à l'aide de la fonction définie dans ma librairie **string_builder_formated** . En plus j'ai défini aussi une structure intitulée s_data qui regroupent tous les structures (airports , flights , airlines et errors) .

2) Choix de conception algorithmique:

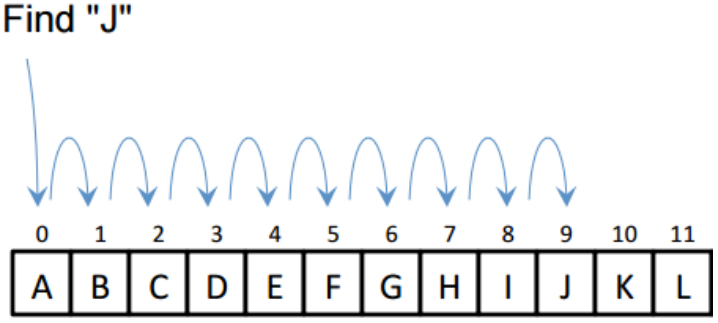
_____Pour trier la data et permettre aussi le filtrage de la data, plusieurs solutions sont envisageables comme le tri à bulles , le tri par insertion ou le tri pivot. Au niveau des algorithmes de recherche il y'a la recherche dichotomique , la recherche ou la recherche linéaire.

La solution la plus adaptée pour le projet est :

~Pour Trier la data : Le tri pivot(quick sort)
exemple:



~Pour le Filtrage data : La recherche linéaire (Linear search)
exemple :



3) les principales étapes du déroulement du programme :

- ➢ Initialiser le guide d'usage (liste des commandes possibles à exécuter)

- ➢ Initialiser les commandes pour pouvoir lancer des requêtes.

- ➢ Allouer une place pour la data et allouer une place pour les listes data (airlines,flights,airports et errors) en utilisant la fonction définie dans ma librairie new_array_list .
- ➢ Le parse de fichiers permet la lecture des fichiers CSV et de vérifier s'il y'a des erreurs dans les fichiers de données csv et les retourner (càd s'il y'a un erreur dans une ligne on retourne (invalid line) dans la liste de data errors), s'il y'a pas d'erreurs on ajoute la ligne lu à notre liste de data du fichier concerné.

- ➢ Vérifier s'il y'a des erreurs dans la liste de data errors et les afficher en utilisant la fonction défini foreach qui permet l'affichage de toutes les erreurs existantes dans le data errors.

- ➢ Sort data :
  - Trier la liste data flights du arr_delay inférieur jusqu'à arr_delay supérieur .
  - Calculer le retard moyen et trier la liste data airlines du retard moyen inférieur jusqu'au retard moyen supérieur.
- ➢ Afficher "**>**" afin que l'utilisateur comprenne qu'il doit écrire une commande.

- ➢ Lire la commande entrée par l'utilisateur et la traiter :
  - Si la commande est vide : réécrire "**>**" afin de permettre l'écriture d'une nouvelle commande.

  - Si la commande est "**quit**" le programme s'arrête.

  - Si la commande entrée ne correspond pas aux commandes définies, le guide d'usage est affiché pour que l'utilisateur puisse trouver facilement ce qu'il cherche.

  - Si la commande entrée correspond à une des commandes définies, à l'aide de **excecute_command**, le programme détermine la commande entrée et cherche l'indice de la bonne commande avec la fonction **excecute**, celle-ci permet aussi de retourner une fonction qui exécute cette commande .

  - Après l'exécution de cette fonction , **excecute** nous retourne une ligne du coup **excecute_command** permet d'afficher cette ligne c'est-à-dire le résultat de la fonction exécuté .Cependant si **excecute_command** retourne FALSE alors le programme affiche "missing args" et affiche l'usage .

IV. Limites de solution

Je me suis rendu compte que choisir d'être seul dans ce projet, n'est pas une décision judicieuse. Car le fait d'avoir une charge de travail sur moi seul , m'a posé un problème en ce qui concerne le temps. Du coup je n'ai pas pu finir les deux dernières requêtes à savoir : **find-itinerary** et **find-multicity-itinerary .**

V. CONCLUSION

La réalisation du projet de programmation avancée m'a permis de consolider mes compétences en programmation et de développer mon autonomie et mon savoir-faire.C'était l'occasion pour mettre en pratique tout ce que j'ai appris en théorie et en ajoutant toutes les connaissances que j'ai appris à travers mes recherches sur internet.

Je suis satisfait, car le programme marche correctement malgré la non-implémentation de deux requêtes, surtout j'ai pris du temps et du plaisir pour le réaliser malgré les problèmes de débogage que j'ai rencontré pendant la conception et la réalisation du projet.

VI. ANNEXE

Bibliographie :

- Strdup: https://man7.org/linux/man-pages/man3/strdup.3.html#:~:text=The%20strdup()%20function%20returns,copies%20at%20most%20n%20bytes.

- Strchr: https://man7.org/linux/man-pages/man3/strchr.3.html#:~:text=The%20strchr()%20function%20returns,c%20in%20the%20string%20s.&text=The%20strchrnul()%20function%20is,of%20s%2C%20rather%20than%20NULL.

- Strncmp: https://linux.die.net/man/3/strncmp