Compte-rendu final de projet

CUPGE S4 – Structures de Données, 2020/21

COUVRAT—PAILLE Titouan, PINÇON Pierre, RAMONDA Baptiste

Sommaire :

Lien GitHub du projet : https://github.com/BaptisteRamonda/ProjetInfoS4

Introduction

Ce rapport vise à présenter les quelques mois de travail de Titouan Couvrat--Paille, Pierre Pinçon et Baptiste Ramonda sur le projet de Structure de Données visant à la réalisation d'une application de gestion d'un réseau ferroviaire en C. Ce projet représentait pour nous trois le premier projet d'envergure dans ce langage (mis-à-part peut-être le projet d'informatique du S3 mais qui en soit était plus de l'écriture de fonctions au sein d'un corpus déjà codé le tout lié à une découverte des outils que sont GitHub et Ubuntu). Ce projet avait donc pour but de nous initier à la gestion de projet puisque pour la première fois nous allions être obligé de travailler séparément sur des parties et des fichiers qui au final s'assemblerait pour former un seul et unique programme qui serait compilable et exécutable.

La première partie de ce rapport se penche donc sur l'organisation et la répartition des différentes tâches ainsi que sur les moyens qui ont été mis en œuvre afin de communiquer du mieux possible. Vous trouverez aussi dans cette partie les retours des différents membres du groupe sur cette expérience et éventuellement sur les points à améliorer pour de futurs projets de ce type.

La seconde partie quand à elle comporte un aspect un peu plus technique puisqu'elle vise à expliquer le code que chacun a écrit. Chaque membre vous expliquera donc les choix qu'il a fait en fonction des différents problèmes rencontrés. Les méthodes et la gestion des fichiers sera aussi expliqués si elles ont eues un intérêt particulier au sein du codage.

Enfin la dernière partie se basera sur le fonctionnement du programme pas-à-pas et sur les différentes options implémentées au cours de son développement.

I/ Organisation globale du projet et répartition des tâches

a) Organisation globale ;

Lors de l'annonce du projet, nous avons tout de suite voulu nous mettre ensemble tous les trois puisque ayant déjà l'habitude de travailler et de nous retrouver tous ensemble, il nous serait plus facile de communiquer et ainsi optimiser l'avancée du projet. Pour cela nous avons donc utilisé les outils à notre disposition et que nous avions tous en commun ; c'est-à-dire un serveur Discord afin de pouvoir communiquer le plus facilement (il nous arrivait cependant de communiquer aussi via téléphone et les réseaux sociaux pour des questions plus précises sur le travail de tel ou tel membre du groupe) et le GitHub de Baptiste. Nous avons choisi GitHub car c'est une des expériences qui nous avait le plus plu dans le projet du S3. En effet la

possibilité de pouvoir travailler tous dans notre coin avant de mettre en commun, tout cela en contrôlant les différentes modifications nous semblait essentiel dans le processus de développement. Nous avons même au début tenté de « lier » le répertoire du projet à notre terminal (via une clé SSH) avant d'abandonner l'idée car trop compliquée alors que l'on pouvait directement uploader nous fichiers grâce à l'interface de GitHub. A partir du moment où nos « outils » de projet étaient en place, nous avons commencé à nous répartir les tâches.

b) La répartition des tâches ;

Pour ce qui est de la répartition des tâches, nous avons commencé par distinguer deux grands domaines : il fallait que quelqu'un gère tout ce qui touchait à l'interface et à l'interaction utilisateur/programme, et que les autres se concentrent sur le dur théorique du programme à savoir la mise-en-place et le codage du réseau ferroviaire. Ayant commencé naturellement chacun à coder une ébauche de programme, Baptiste a décidé de s'occuper de la partie interface, Titouan s'est chargé d'écrire les différentes structures et les premières fonctions du réseau, et Pierre, dans la lignée de notre programme d'informatique théorique, s'est attelé à coder l'algorithme de plus court chemin de Dijkstra dont nous avons besoin pour calculer les trajets de notre réseau.

c) Problèmes rencontrés liés à l'organisation

Comme vous le verrez par la suite, l'organisation prévue n'a pas forcément eu le résultat attendu et nous avons expérimenté les problèmes récurrents liés à un projet qui s'étale sur plusieurs mois. En effet, lors de l'annonce du projet, nous étions tous les trois très enthousiastes à l'idée de faire ce projet et nous nous motivions déjà à faire plus que ce qui était demandé (je pense notamment à la réalisation de l'interface du programme dans une fenêtre graphique ce qui aurait demandé de la coder en Java). Nous avons donc tous commencé à coder dans notre coin des petits programmes en fonçant un peu tête baissée sans avoir réellement pris conscience de l'ampleur de ce qui nous attendait et sans avoir vraiment évalué toutes les options, erreurs, et choix possibles dans le code.

Le problème fut que avec l'arrivée des CC, l'accélération du rythme en cours et la place que commençait à prendre certaines matières dans notre travail quotidien, la date échéante du projet se trouvant encore loin, on a commencé à avoir une baisse de motivation et d'implication dans le projet. Et cela nous a valu l'arrêt du code pendant plusieurs semaines.

Les conséquences de cette pause dans le projet, c'est que lorsque les programmes des cours ont commencé à s'alléger et que nous avions de plus en plus de temps libre à consacrer au projet, nous avons pleinement pris

conscience du travail restant à fournir et de la complexité de celui-ci. De plus, nos pseudos-codes du début n'étaient globalement plus valables ou plus adaptés à la vision que nous avions du projet et la direction que l'on voulait prendre. Nous avons donc quasiment été forcés de tout reprendre depuis le début ce qui nous a fait perdre un temps considérable et ne nous a pas permis, comme vous le verrez par la suite, de finaliser ce projet ; de nombreuses options et fonctionnalités du programme n'ayant pas encore été implémentées. En ce qui concerne l'organisation, nous avons aussi pu observer une détérioration, que nous avons encore du mal à expliquer, au sein de la communication entre nous ce qui a fini par avoir des conséquences lorsque un membre attendait le travail d'un autre pour pouvoir continuer le sien. Malgré tout nous avons réussi à obtenir un résultat qui fonctionne et qui est compilable et exécutable. Passons maintenant à l'explication du code par chacun des membres du groupe.

II/ Organisation du code

a) L'interface et la gestion des logins (Baptiste)

Commençons donc par la partie interface du programme. Pour ma part, j'ai ressenti que la partie « mise en place de l'interface » n'était pas la plus complexe théoriquement mais en pratique j'ai tout de même eu pas mal de problèmes à résoudre concernant ce sujet.

Le problème principal étant que l'interface est composée d'un menu principal, puis de sous-menus, de sous-sous-menus et même parfois de sous-sous-sous-menus. L'une des solutions alors qui m'a paru la plus évidente pour faire face à cette organisation à été de tout organiser sous forme de « switch case » imbriqués les uns dans les autres puisque c'était à l'époque la seule structure de choix en c que l'on connaissait. Un autre problème est alors apparu puisqu'en remplissant des switch par d'autre switch je me retrouvais alors avec plusieurs variables de choix qui posaient de plus en plus de problèmes, notamment avec la gestion de la mémoire du stdin. De plus la question du retour en arrière se posait puisque à chaque fois qu'un sous-menu avait fini de s'exécuter, le programme sortait de la boucle, puis de la boucle supérieure et ainsi de suite jusqu'au return du main ce qui conduisait à l'arrêt pur et simple du programme. J'ai alors opté pour des « fonctions menus », ce qui a marqué le début de l'organisation de mon code en fichiers. Désormais j'avais deux fichiers menu.h et menu.c qui géraient mes fonctions menus auxquelles j'avais directement intégré des switch. Ainsi cela allégeait la fonction main en plus de la rendre plus lisible. Pour la gestion du retour en arrière, j'ai été dans l'obligation d'aller chercher une réponse à mon problème sur internet (les forums quel monde merveilleux…). J'ai alors découvert l'existence des labels, qui malgré leur mauvaise réputations dans le monde des développeurs me permettaient de parvenir à mes fins, j'ai donc

décidé de les garder (le fait de placer un goto menuPrincipal juste avant le return du main empêchait le programme de se fermer à chaque fermeture d'un sous-menu).

Une autre chose que j'ai codé au sein de ce projet et que j'ai oublié de mentionner auparavant, c'est le Makefile. Malgré le fait que celui-ci n'a pu être totalement bouclé qu'à la fin du projet (puisqu'il nécessitait la totalité des fichiers) il m'a permis de gagner beaucoup de temps sur la compilation et la recherche d'erreurs.

La deuxième grosse partie de mon travail s'est basé sur de la gestion de fichiers dans le but de rendre accessible ou non un menu à l'aide d'identifiants. Pour cela, j'ai crée différents fichiers dans lesquels seraient stockés le login et le password du contrôleur et de l'administrateur. La difficulté étant de conserver ces identifiants à la fermeture du programme, il était nécessaire de passer par de la gestion de fichiers. J'ai réussi a résoudre ce problème grâce à une fonction **read** qui me permettait à chaque redémarrage du programme d'initialiser les identifiants. Pour ce qui est de la modification de ces identifiants depuis les différents menus, l'ouverture des fichiers en mode 'w+' me permettait de réinitialiser le fichier avant d'y transférer les nouveaux identifiants saisis par l'utilisateur. Grâce à cela, les menus étaient désormais protégés et les modifications d'identifiants perduraient dans le temps grâce à un stockage externe au programme des identifiants. Les fonctions de gestion de fichiers ont d'ailleurs été réutilisées par la suite par Titouan pour stocker les billets des voyageurs.

Pour finir, je dirais que ce qui m'a pris le plus de temps (hormis la résolution des problèmes et des situations évoquées ci-dessus) c'est la 'mise en page', et la volonté que j'avais que le programme réagisse comme une véritable application aboutie et non pas juste comme une succession de ligne de codes. Pour cela, j'ai massivement utilisé la commande **system('clear'),** ce qui permettait de changer d'affichage en effaçant complètement l'affichage précédent. Cette méthode m'a permis d'approcher le modèle d'interface graphique souhaitée au départ pour nous éloigner du terminal basique et son affichage d'étapes successives.

b) L'algorithme de Dijkstra (Pierre)

Afin de calculer le plus court chemin, j'ai décidé d'utiliser l'algorithme de Dijkstra d'entre toutes les options qui m'étaient disponibles principalement car c'est celui avec lequel j'étais le plus familier. L'algorithme en lui même permet de renvoyer en mémoire les villes par lesquelles on est passé ainsi qu'évidemment la distance la plus courte.
Le cahier des charges de la fonction a évolué au fur et à mesure du projet, ce qui a amené à de nombreuses modifications par rapport à la fonction de départ à tel point qu'elle n'a plus rien à voir et est passée par une refonte complète avant sa version finale.

Dans sa version finale la fonction prend en argument une matrice d'adjacence pondérée : chaque « case » de cette matrice correspond à la durée du chemin direct entre deux villes, et lorsque celui ci n'existe pas la valeur est mise, par défaut, sur « inf » correspondant virtuellement à l'infini mais dans les faits étant juste une valeur bien trop haute pour que l'algorithme ait une quelconque chance de la sélectionner. L'algorithme prend également en argument évidemment la ville d'arrivée et la ville de départ, mais plus surprenamment le nombre de ville. En réalité le nombre de ville en tant qu'argument à été rajouté vers la fin du projet car un problème d'include subsistait, et ce fut le moyen le plus efficace de le résoudre, quitte à sacrifier la modularité.

Le projet quoique positionné dans une période plus que surchargée niveau travail a été intéressant et instructif, il n'a pas été perçu comme un perte de temps ou d'énergie. Au contraire c'est plutôt l'impression d'un investissement sur le long terme qui reste, car même s'il n'a pas aboutit comme nous l'aurions voulu et souhaité, ce projet nous a apporté beaucoup au niveau de la gestion de projet, de la planification et de l'approche d'un travail à long terme.

Lorsque le projet a débuté, j'étais trop confiant vis-à-vis de ma capacité à construire un algorithme de plus court chemin,de le moduler et le modeler à ma guise. De ce fait la première version que j'ai fournie n'était pas satisfaisante et j'ai travaillé beaucoup plus que je l'imaginais sur cette fonction. La difficulté se trouve donc dans le temps dépensé dans une fonction que je jugeais simple, temps que je n'ai donc pas pu dépenser pour faire les fonctions auxiliaires que je voulais .

c) Le fonctionnement du réseau (Titouan)

Le réseau de train est une structure composée de Lignes, de Trajets et de Villes.
Un Trajet contient :

- Une ville de départ **depart**
- Une ville d'arrivée **arrivée**

Les Trajets sont composés de deux Villes, une de départ (départ) et une d'arrivée (arrivée). Un vice du réseau est que les trajets sont seulement implémentés dans un seul sens

- Un nombre de place restantes **nbPlacesRestantes**

Cette variable modélise le nombre de places encore disponible pour le trajet entre deux villes directement connectées. Pour palier au vice du trajet a sens unique, il suffit de créer une variable nbPlacesRestantesInverse pour représenter le trajet dans le sens inverse a celui implémenté. En attendant, cette sous-variable représente les places disponibles pour un trajet entre deux villes dans les *deux* sens. La

capacité maximale d'un train est codée sous la variable constante NBPLACES dans reseau.c.

- Un **temps**

C'est le temps que prend le train pour aller d'une ville à celle d'après.

Les Lignes contiennent :

- Un tableau contenant les étapes (villes) de la ligne **tEtapes[]**
- Un tableau contenant les trajets partiels entre toutes les villes de la ligne **tTrajetsPartiels[]**

Ces tableaux contiennent respectivement toutes les villes desservies et les trajets entre elles. A partir des ces tableaux il est facile d'accéder au temps qu'il prend pour aller d'une ville a l'autre même si elles ne sont pas reliées directement entre elles.

- Un nombre d'étapes **nbEtapes**
- Un nombre de trajets **nbTrajets**

Le réseau est composé de :

- 4 lignes, celles données par le sujet **tLigne[]**

Le réseau est ainsi initialisé avec Lignes Paris-Marseille, Paris-Toulouse, Bordeaux-Nice et Paris-Lyon,

- Un tableau de villes desservies **tVilles[]**
- Un nombre de ville **nbVilles**
- Un nombre de trajet **nbTrajet**

J'ai eu quelques problèmes d'allocation de mémoire avec des variables à plusieurs niveau. Par exemple, si « r » est le réseau complet, « r->tLigne[0]→tTrajetsPartiels[0]→depart→nomVille » serait le int 0, correspondant à la ville de Paris. La ligne
printf(« %s », returnNomDeVille(r->tLigne[0]→tTrajetsPartiels[0]→depart→nomVille) ;
afficherait « PARIS ». Les fonctions returnNomDeVille(Int i), afficheNomDeVille(Ville v) ou afficheNomDeVilleInt(Int v) sont toutes plus ou moins les mêmes switch case, et servent à l'interface.
La fonction InitialiseTrajet(Ville, Ville) était un peu un challenge car pour initialiser un trajet je ne pouvais pas le faire directement dans le réseau, sinon ça m'aurait pris des plombes. Pour gagner du temps j'ai créé une fonction qui allouait la mémoire nécessaire à un trajet, en initialisant les nombre de places restantes à NBPLACES et en associant les villes aux variables **départ** et arrivée. Pour initialiser le temps j'ai dû le faire à la main car chaque trajet prend un temps différent que celui d'à côté.
La fonction creerReseau() mets toutes ces fonctions ensemble pour créer la structure la plus complexe du projet.

La fonction matAdj(r) crée la matrice d'adjacence d'un réseau. Pierre en avait besoin pour créer l'algorithme de plus court chemin Dijkstra.
J'ai écrit un README sur le Github résumant comment le réseau était codé pour que Pierre et Baptiste puissent comprendre comment la structure fonctionnait sans avoir eu à comprendre le code.

III/ Fonctionnement du programme

Le programme s'articule donc sous la forme de différents menus et sous-menus permettant à chaque profil d'effectuer des choix en fonction de sa situation.

Après un message de bienvenue, le menu principal invite l'utilisateur à saisir son profil : soit voyageur, soit contrôleur, soit administrateur. On note aussi ici la présence d'une option pour quitter le programme, comme expliqué ci-dessus c'est la seule façon de quitter le programme. Plaçons nous à présent dans les différents cas de figure :

- Si vous êtes voyageur, vous avez le choix entre trois options : rechercher un trajet, réserver un billet et revenir au menu principal. Dans la première option, la liste de tous les arrêts et gares s'affiche et en saisissant les numéros correspondants vous êtes invités à sélectionner votre gare de départ et votre gare d'arrivée. Le programme est ensuite censé vous afficher le trajet correspondant entre les gares de votre choix tout en affichant les arrêts intermédiaires. L'option « réserver un billet » débute par le même affichage, au détail près qu'après avoir saisi votre départ et votre destination, celui-ci vous demande de saisir votre nom pour vous réserver un billet sur ce train, vos coordonnées sont ensuite envoyées dans un fichier consultable et exportable par l'administrateur réseau. La dernière option vous permet de revenir au menu principal pour changer de profil.

- Si vous êtes contrôleur, avant même d'afficher le menu, le programme vous demande un login et un mot de passe puisque vous tentez d'accéder à un menu privé et sécurisé. Après avoir correctement entré vos identifiants vous pouvez accéder au menu contrôleur. Celui-ci offre le choix entre trois options. Dans la première option, la liste de tous les arrêts et gares s'affiche et en saisissant les numéros correspondants vous êtes invités à sélectionner votre gare de départ et votre gare d'arrivée. Le programme est ensuite censé vous afficher le trajet correspondant entre les gares de votre choix tout en affichant les arrêts intermédiaires. Dans la deuxième option, le programme vous demande de saisir votre nouveau mot de passe et votre nouveau login. Une fois que c'est fait, si vous quittez le programme ou juste le menu contrôleur, il vous sera demandé vos nouveaux identifiants. La dernière option vous permet de revenir au menu principal pour changer de profil.

- Si vous êtes administrateur, le programme vous demande un login et un mot de passe puisque vous tentez d'accéder à un menu privé et sécurisé. Après avoir correctement entré vos identifiants vous pouvez accéder au menu administrateur. Celui-ci offre quatre choix d'options. Dans la première option vous accédez au menu de gestion de trajet. Ce menu était censé permettre de ajouter/supprimer une gare, ajouter/supprimer/modifier un train, ajouter/supprimer un trajet, mais les problèmes liés à la fusion entre l'interface et le réseau n'ont pas permis de réaliser cette partie.
La deuxième option a souffert du manque de temps pour finir le projet elle aussi mais elle était censé récupérer les fichiers contenant la liste des passagers ayant réservé leur billet, ou bien la liste des trains sur une ligne, pour pouvoir l'exporter au format JSON. La troisième option permet de modifier les identifiants d'un contrôleur ou bien de l'administrateur. Dans les deux cas la démarche est la même : le programme vous demande de saisir votre nouveau mot de passe et votre nouveau login. Une fois que c'est fait, si vous quittez le programme ou juste le menu contrôleur, il vous sera demandé vos nouveaux identifiants. La dernière option vous permet de revenir au menu principal pour changer de profil.

Lorsque vous revenez au menu principal et que vous choisissez l'option pour quitter le programme, le programme vous affiche un message de fin et le programme se ferme.

Conclusion

Nous avons commencé le projet avec énormément d'enthousiasme et de motivation, et chacun d'entre nous a débuté le projet dès les premiers jours ce qui nous a permis de garder une base solide de travail durant la totalité du projet, et ce même lorsque nous n'avions que très peu de temps pour travailler dessus. En effet il y a eu des moments où gérer la quantité de travail que nous avions en parallèle ne nous a pas permis de travailler comme nous le voulions et ça a donc résulté avec une période creuse, période suite à laquelle nous étions content d'avoir travaillé autant au début.
Dès que le temps nous l'a permis nous avons repris le projets avec une efficacité qui nous a permis de rattraper beaucoup du temps que nous n'avons pas pu consacrer au projet. Néanmoins, nous n'avons pas réussi, à peu de choses près, à fournir un travail complet et fini. Nous avons certes pu avoir toutes les parties qui marchaient séparément, mais le temps nous a finalement manqué pour tout raccorder et ce même si les parties marchaient toutes deux à deux. Il y a un certain nombre de choses que nous aurions aimé rajouter au cahier des charges du projet de départ que nous n'avons pas pu implémenter tels que de la modularité (dans les voyageurs ou les trajets, les lignes…). Toutefois ce projet aura été une expérience enrichissante que ce soit au niveau de la gestion de projet, de la planification,

de la communication et évidemment de la manière de gérer un code (non pas pour soi même mais pour le partager afin que d'autres puissent à leur tour travailler dessus). C'est d'ailleurs un aspect positif majeur des problèmes qu'on a pu rencontrer, ils sont enrichissants : on a dû chercher les outils pour palier à ces problèmes, et ce sont des outils que nous avons maintenant à notre disposition pour d'autres projets.