|  |
| --- |
| Guillaume de La Grandiere, Inès Kettani |
| Initiation à l’Algorithmique et à la Programmation |
| Dossier de développement |

Table des matières

[Présentation 2](#_Toc55244026)

[Introduction : 2](#_Toc55244027)

[Objectif du projet : 2](#_Toc55244028)

[Entrées et sorties : 3](#_Toc55244029)

[Organisation du développement 4](#_Toc55244030)

[Sprints : 4](#_Toc55244031)

[Tests des sprints : 5](#_Toc55244032)

[Bilan du projet 6](#_Toc55244033)

[Difficultés rencontrées : 6](#_Toc55244034)

[Conclusion : 7](#_Toc55244035)

[Annexe 1 : validation entrée/sortie 7](#_Toc55244036)

[Sprint 1 : 7](#_Toc55244037)

[Sprint 2 : 8](#_Toc55244038)

[Sprint 3 : 8](#_Toc55244039)

[Sprint 4 : 9](#_Toc55244040)

[Sprint 5 : 9](#_Toc55244041)

[Sprint 6 : 9](#_Toc55244042)

[Annexe 2 : code source sprints 11](#_Toc55244043)

[Sprint 1 : 11](#_Toc55244044)

[Sprint 2 : 15](#_Toc55244045)

# Présentation

## Introduction :

Les progrès technologiques ont permis un essor du traitement de l’information sous forme numérique. Cette méthode permet de réduire les erreurs potentielles dû à l’imprécision d’un traitement physique sur papier. Quand une entreprise atteint une taille conséquente, elle rencontre la nécessité de crée un système de gestion informatisé. Le but de ce projet sera donc de créer un système de gestion pour cette entreprise.

## Objectif du projet :

Le but de ce projet est de programmer un interpréteur de commandes capables de gérer une base de données sous forme de tableaux. Il doit être capable de gérer toutes les étapes de la complétion d’une commande : enregistrer un client et ses commandes, gérer les travailleurs en fonction de leurs compétences et répartir de manières optimisée les taches à effectuer ainsi que facturer les commandes. En finissant le sprint 6 release nous avons un programme complet qui peut enregistrer des clients, des spécialités, des commandes ainsi que des travailleurs. Il peut assigner des taches au commandes dans différentes spécialités et les assigner aux travailleurs en fonctions de leur quantité de travail à effectuer. Ces tâches peuvent ensuite être avancées, réassignées et terminées. Enfin, les commandes sont facturées une fois terminées, et, quand toutes les commandes sont facturées, le programme doit afficher la liste de tous les clients ainsi que la facture totale qu’ils ont à payer.

## Entrées et sorties :

Le projet comporte 12 entrées, chacune correspondant à une fonction du programme : traite\_demarche, traite\_embauche, traite\_commande, traite\_supervision, traite\_client, traite\_travailleurs, traite\_specialites, traite\_tache, traite\_progression, traite\_charge, traite\_passe et traite\_interruption.

Toutes ces fonctions sont appelées dans le main via une commande de l’opérateur grâce a la fonction get\_id qui permet de récupérer une chaine de caractère. Celle-ci est comparée aux différentes commandes possibles grâce a la fonction strcmp() qui permet de comparer deux chaines de caractères, importée de la librairie <string.h>.

Les sorties sont effectuées via des printf qui se trouvent dans différentes fonctions :

* traite\_supervision qui affiche l’état des tâches pour chaque commande.
* affiche\_client qui affiche un client et ses différentes commandes.
* affiche\_travailleur qui affiche les travailleur compétents pour une spécialité.
* affiche\_specialite qui affiche une spécialité et son coût horaire.
* traite\_charge qui affiche les taches assignées à un travailleur ainsi que le nombres d’heures de travail restantes pour celles-ci.
* traite\_facturation qui affiche la facture pour une commande avec la facturation de chaque tâche.
* traite\_fin qui affiche la facture totale de chaque client et clos le programme.
* traite\_interruption qui affiche le message d’interruption et clos le programme.

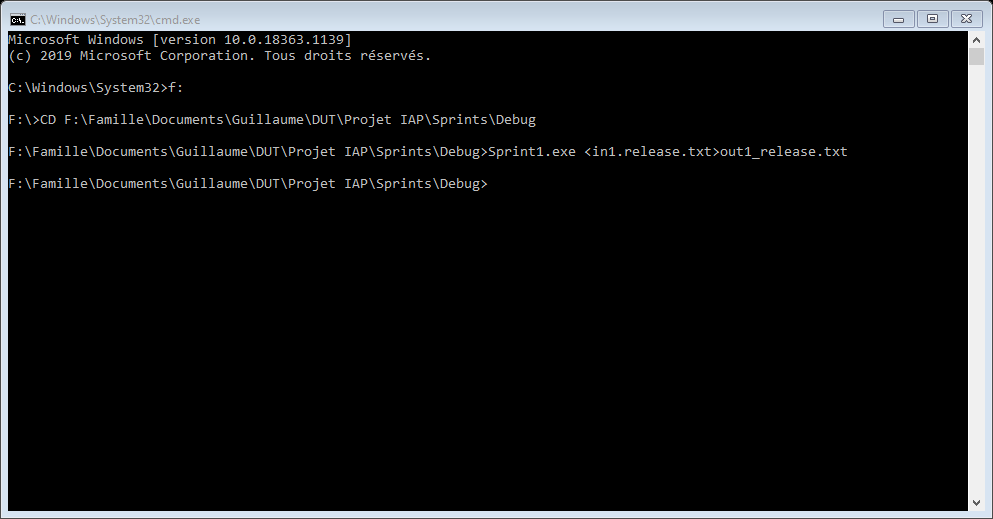
# Organisation du développement

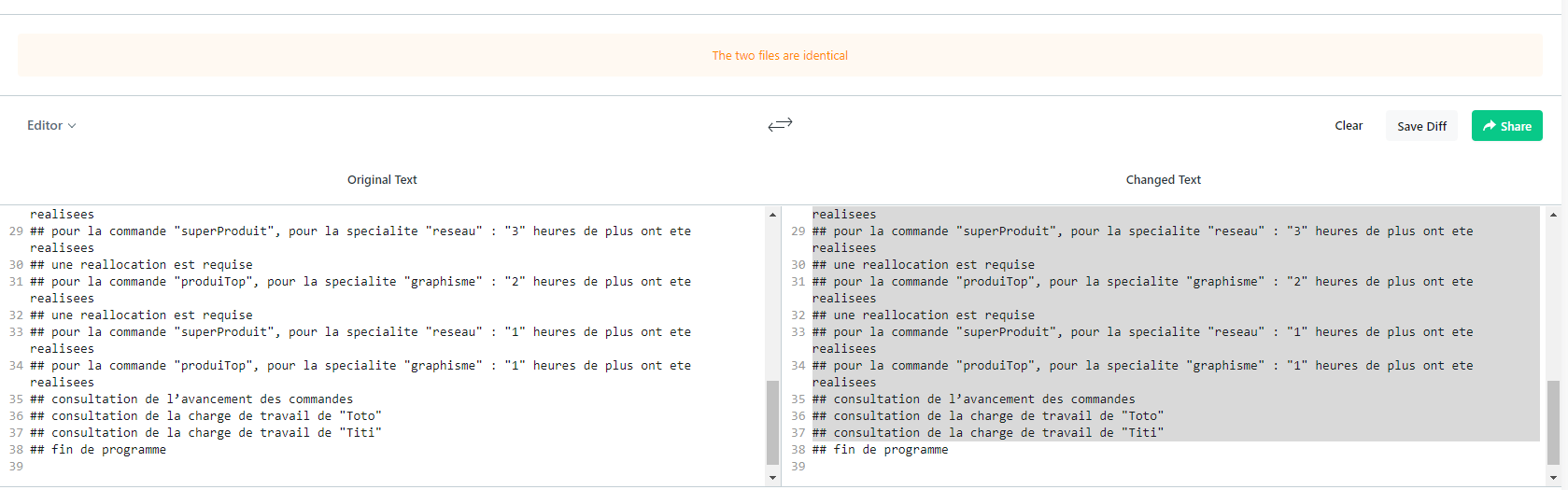
## Sprints :

Le projet est organisé en sprints, permettant une évolution du programme par palliés. Ils sont catégorisés comme suit :

* Sprint 1 : toutes fonctions provoquent uniquement un affichage en fonctions des paramètres d’entrée
* Sprint 2 : le programme doit pouvoir enregistrer les travailleurs, les spécialités ainsi que les clients. Il doit pouvoir les afficher sur demande de l’opérateur
* Sprint 3 : Le programme doit pouvoir enregistrer des commandes et créer des tâches associées a celles-ci. Il doit aussi pouvoir afficher les taches et leur avancement. Il doit aussi pouvoir avancer et compléter les tâches
* Sprint 4 : Le programme doit pouvoir assigner automatiquement les tâches au travailleur ayant le moins de charge de travail.
* Sprint 5 : Le programme doit pouvoir facturer une commande terminée, et, une fois toutes les commandes facturées, afficher la facturation finale pour chaque client et clore le programme
* Sprint 6 : Le programme doit pouvoir réassigner une tache non complétée à la demande de l’utilisateur.

## Tests des sprints :

Pour valider les tests, un fichier contenant des entrées doit être lu et exécuté par le programme et donner en sortie un fichier identique à celui fourni. Le fichier « out » peut être obtenu en passant le fichier « in » en entrée lors d’une exécution du programme dans l’invite de commande comme ceci :

Le fichier « out » ainsi obtenue peut être alors comparé au fichier « out » fourni pour vérifier la validité du code. Nous avons utilisé le site [diffchecker.com](https://www.diffchecker.com/) pour effectuer la vérification.

# Bilan du projet

## Difficultés rencontrées :

Les difficultés rencontrées ont été progressives : sur le premier sprint l’écriture était uniquement répétitive. Sur le second sprint aucune difficulté particulière n’a été rencontrée. A partir du 3-ème sprint des erreurs sont apparus dû à la complexité des structures du programme. Pour le quatrième sprint les structures ne sont plus fournies et le plus compliqué a été de savoir comment modifier de manière optimisée les structure pour prendre en compte les affectations de tâches et ne pas surcharger la mémoire.

Par la suite le seul réel problème a été l’optimisation du programme une fois le sprint 6 release terminé.

Dans sa version actuelle, le programme n’est pas complètement optimisé et certaines commandes peuvent paraitre brouillon. Nous avons fait notre maximum mais le programme n’est pas le plus réduis possible.

## Conclusion :

Ce projet présentait une certaine difficulté de part le fait qu’il s’agissait du premier projet que nous avions à réaliser en C. Cependant la division en sprints permet un travail échelonné et une difficulté progressive.

En supplément les sprints ne donnent pas de consigne précises ce qui nécessite de chercher à travers les « in » et les « out » fournis pour connaitre une partie des commandes et des instructions à implémenter.

Ce projet nous a aussi permis d’apprendre à coder en équipe sur un code et à relire le travail des autres pour corriger les erreurs d’inattention ainsi que celle de codage. Cela nous a donné un aperçu du travail dans le monde professionnel.

# Annexe 1 : validation entrée/sortie

## Sprint 1 :

## Sprint 2 :

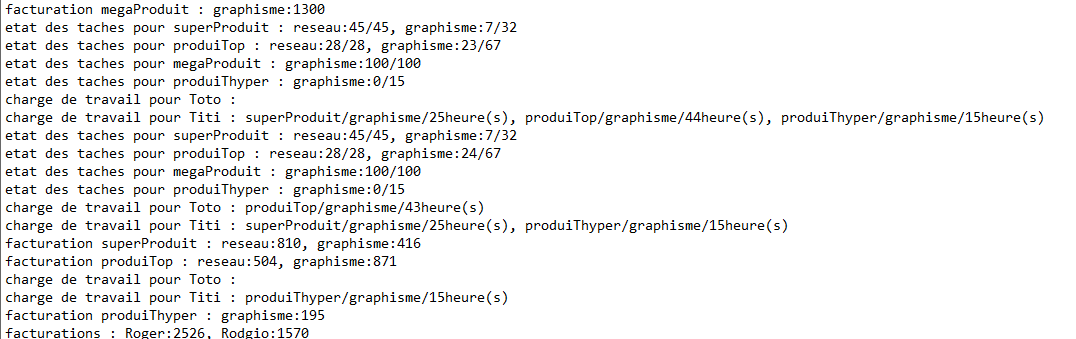
## Sprint 3 :

## Sprint 4 :

## Sprint 5 :

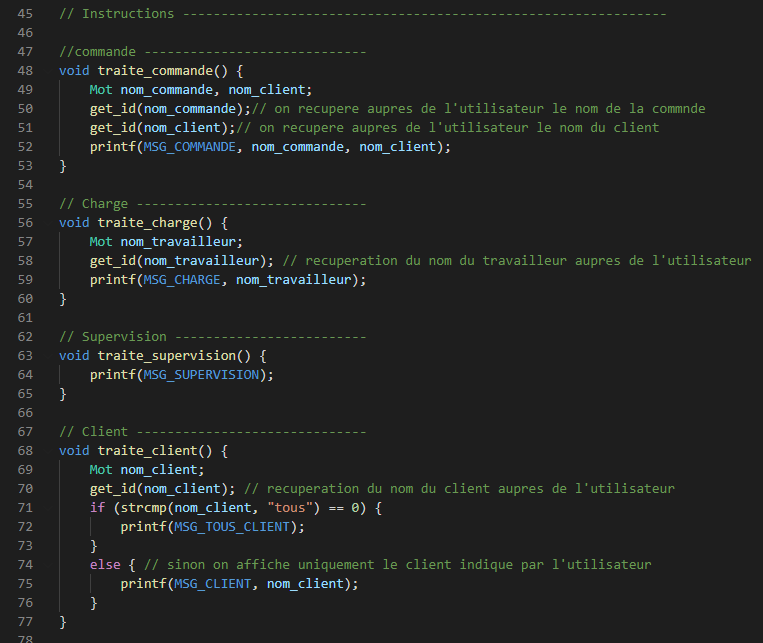
## Sprint 6 :

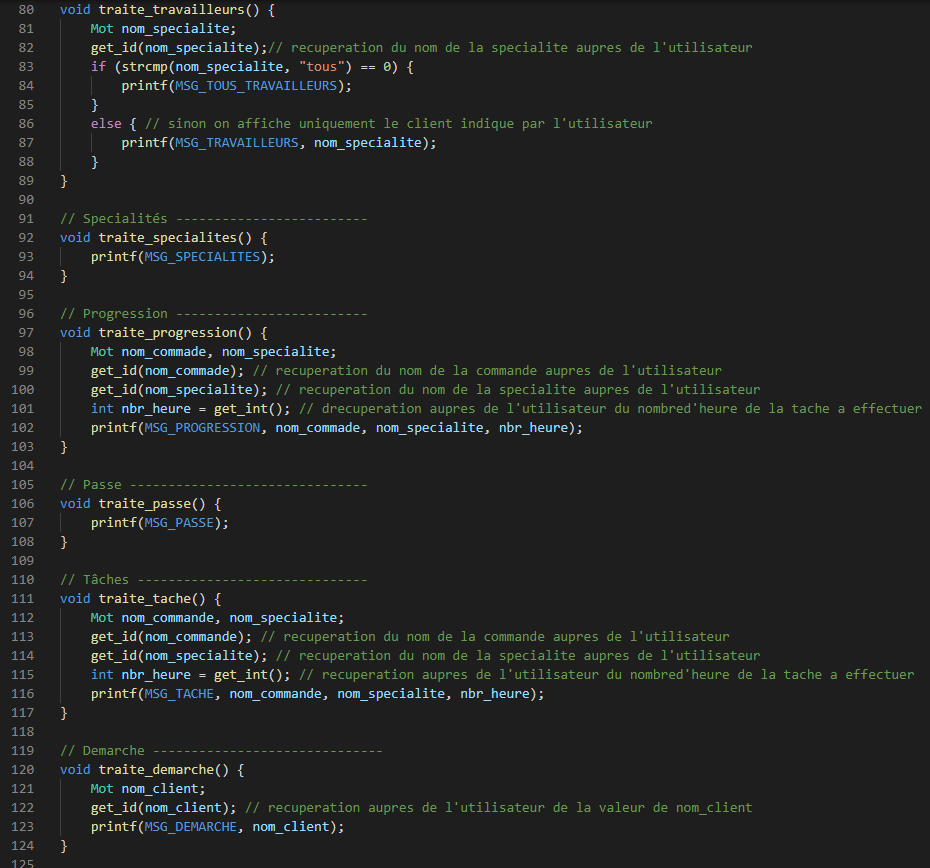
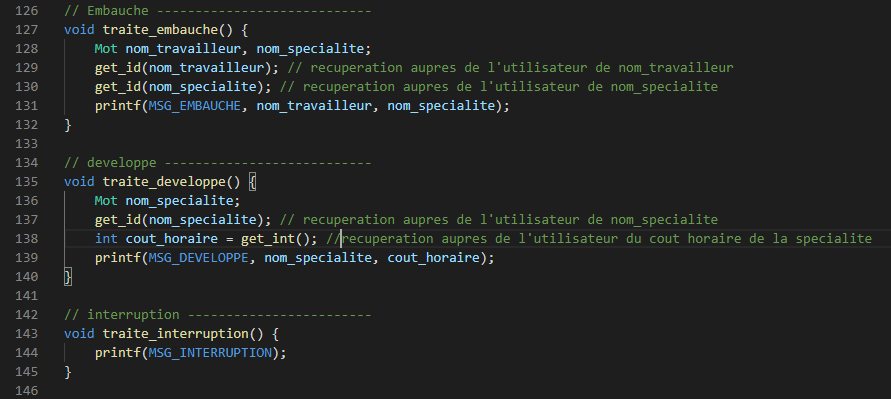
« out » sprint 6 release :

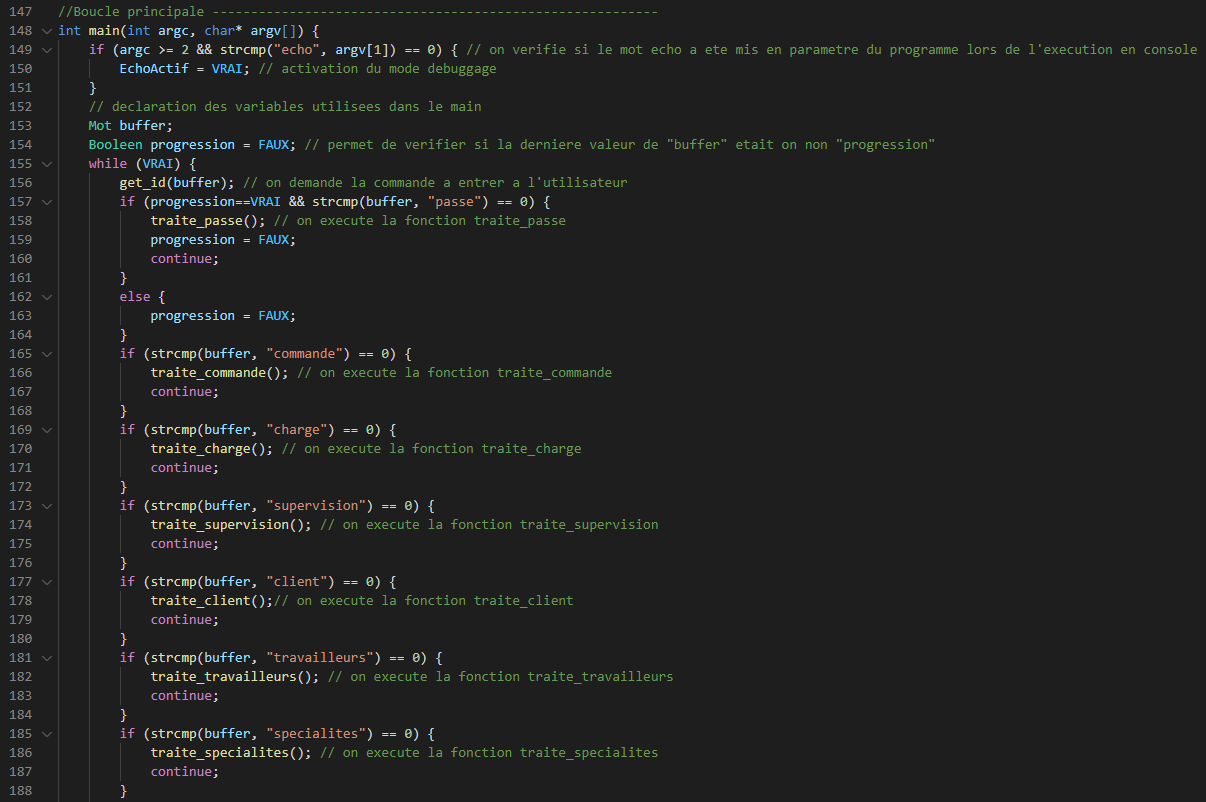
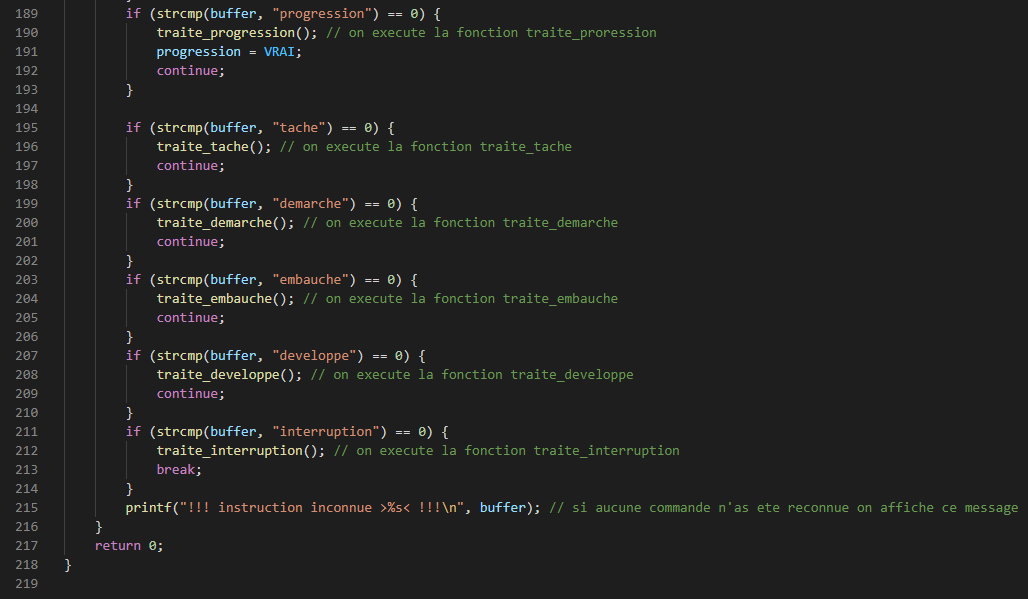


# Annexe 2 : code source sprints

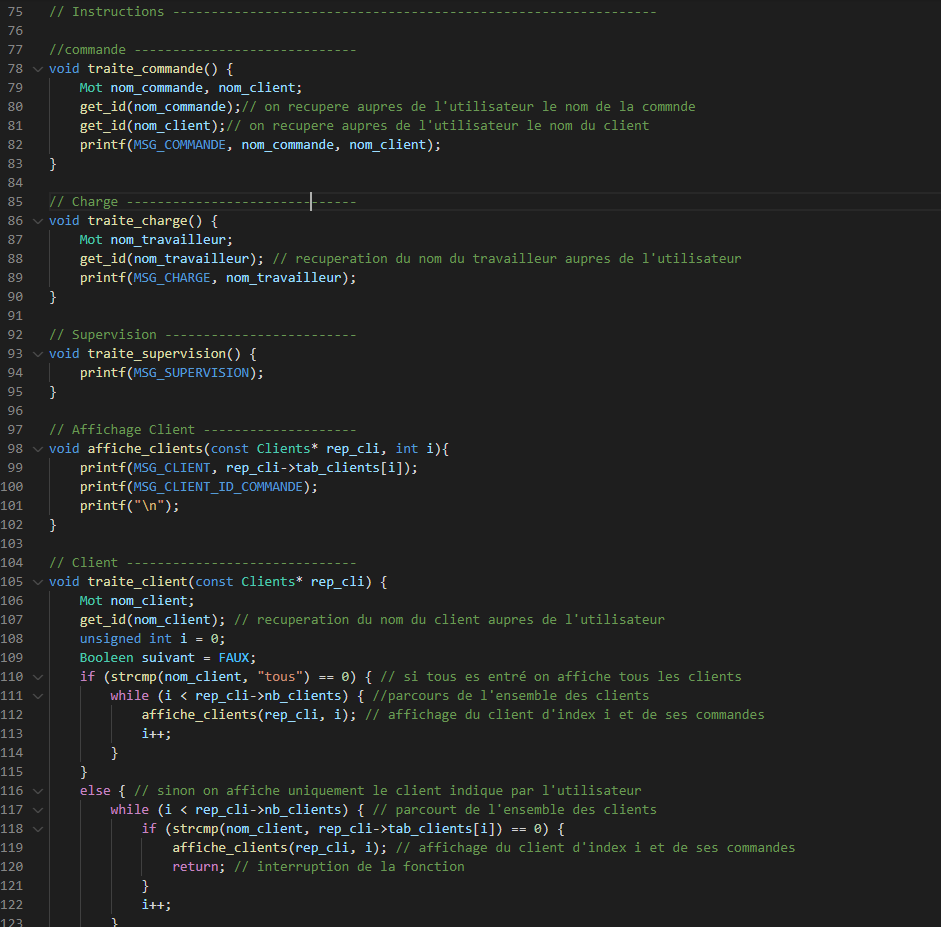
## Sprint 1 :

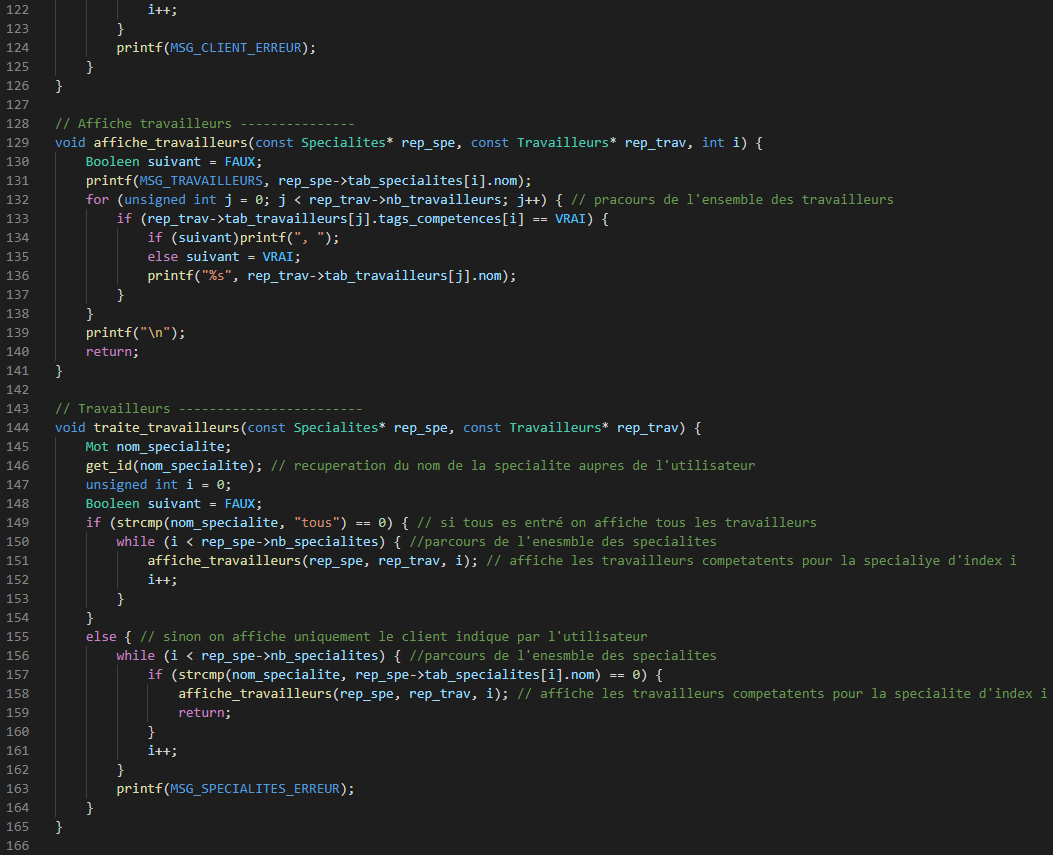


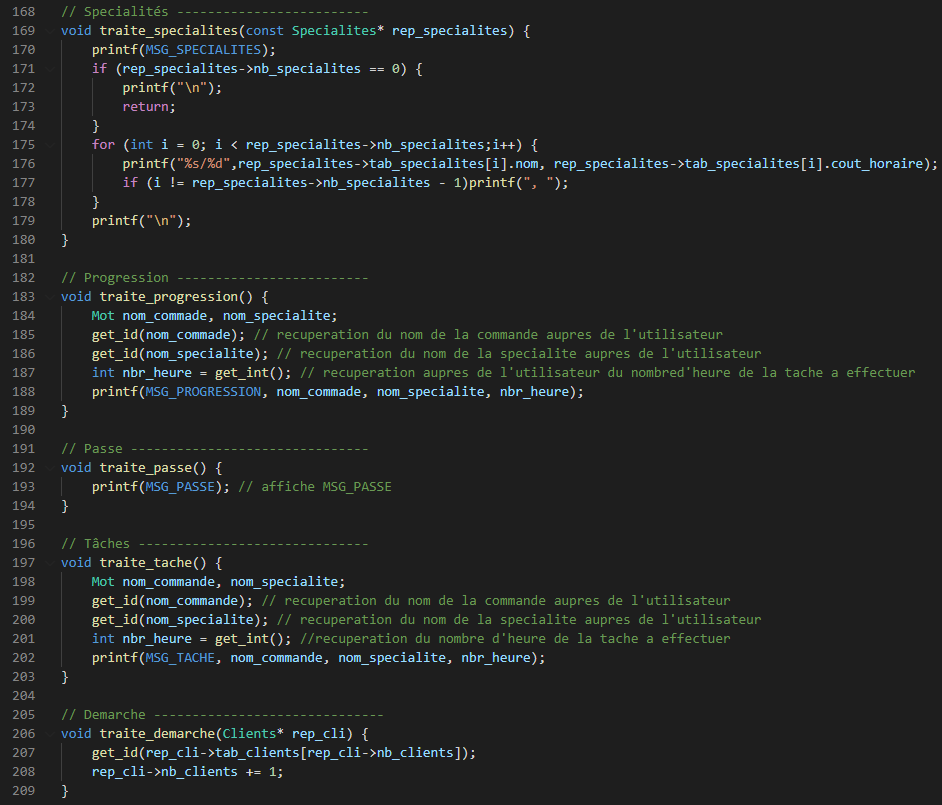


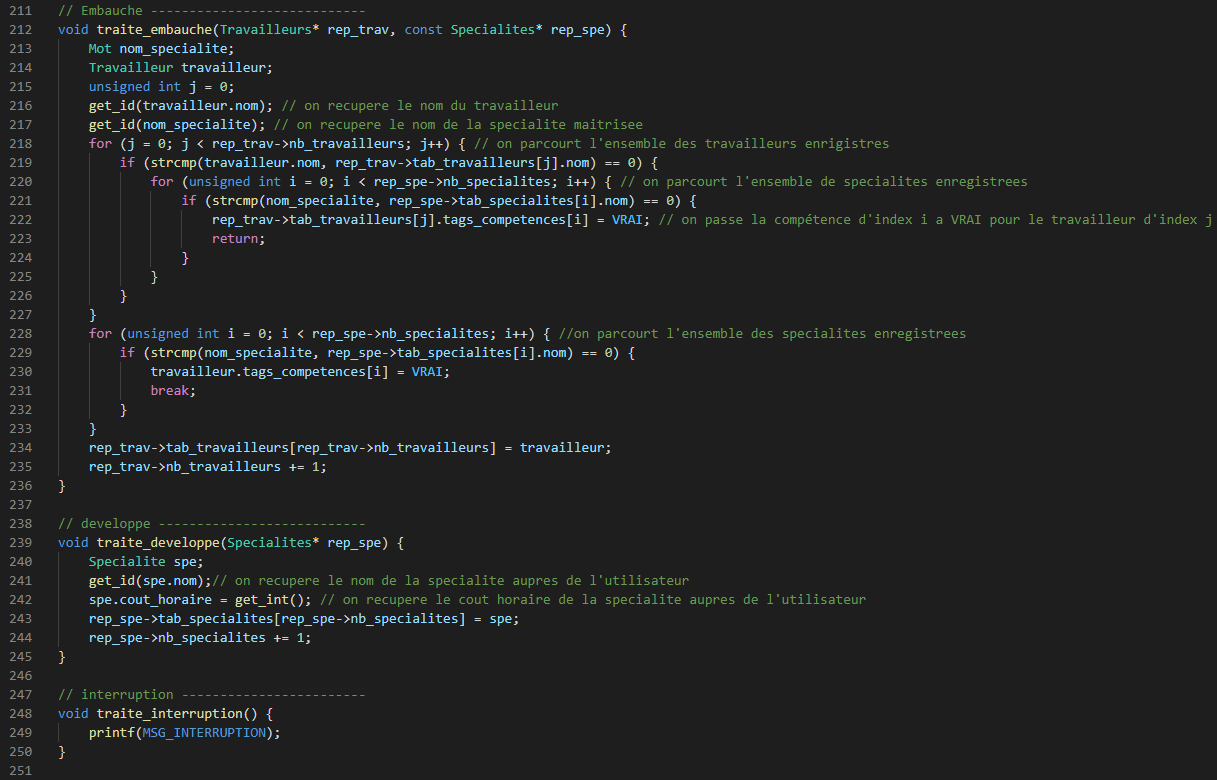


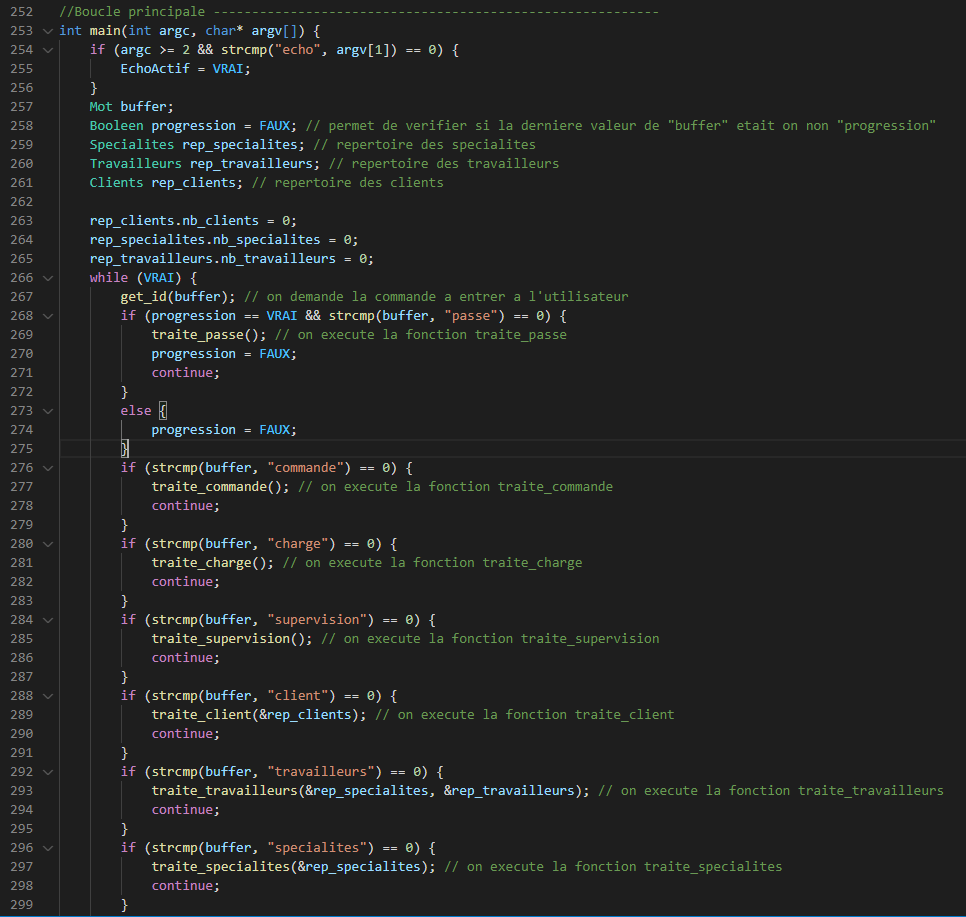
## Sprint 2 :

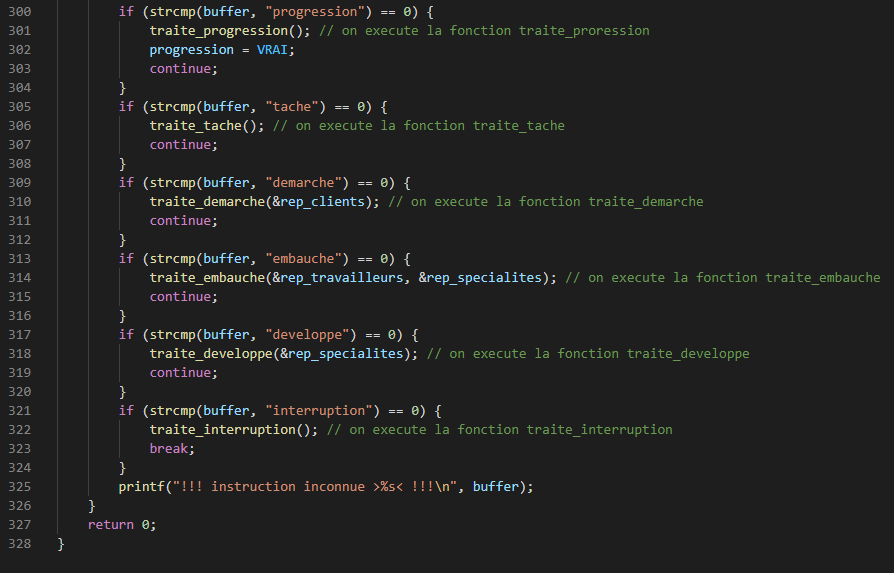




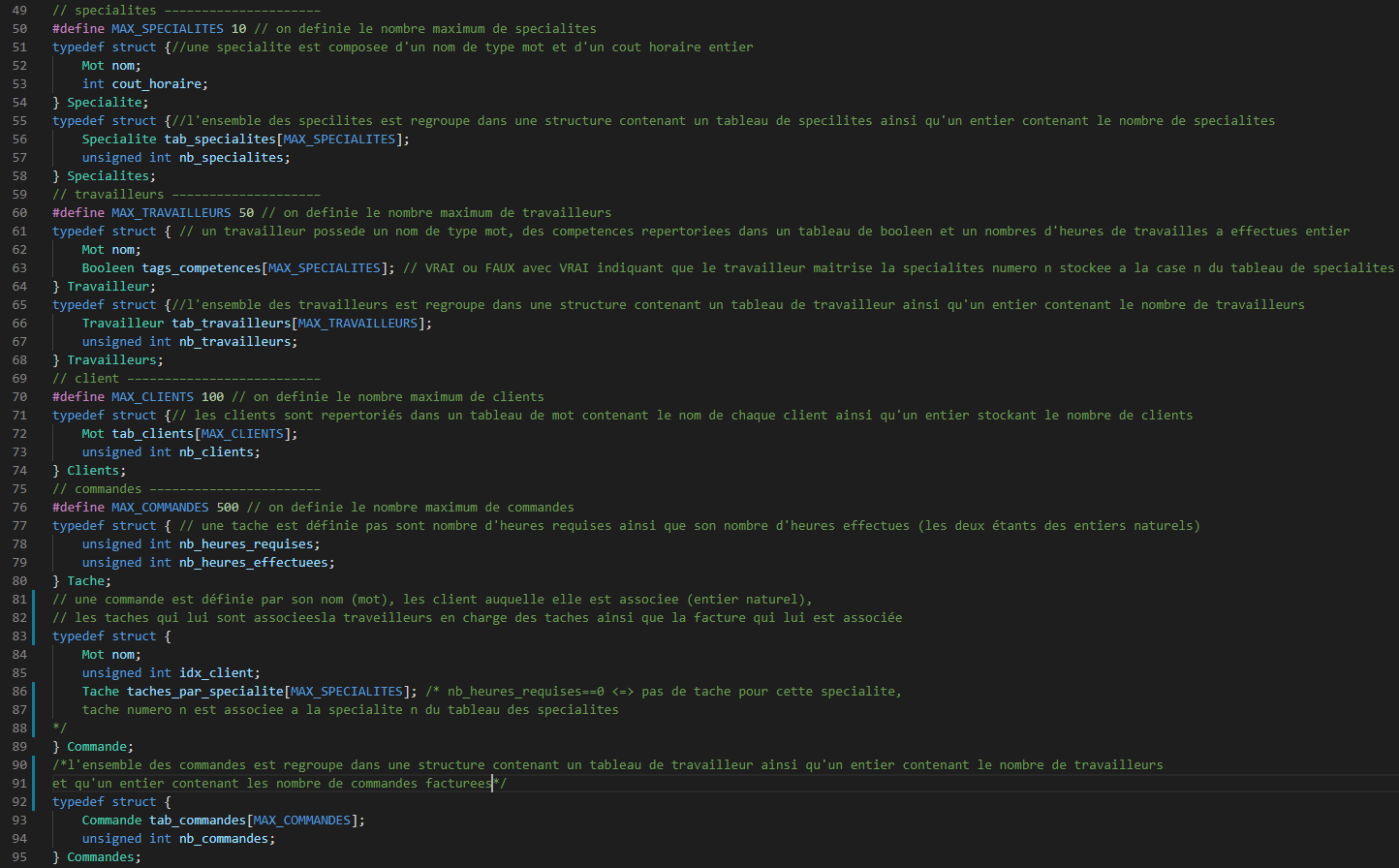
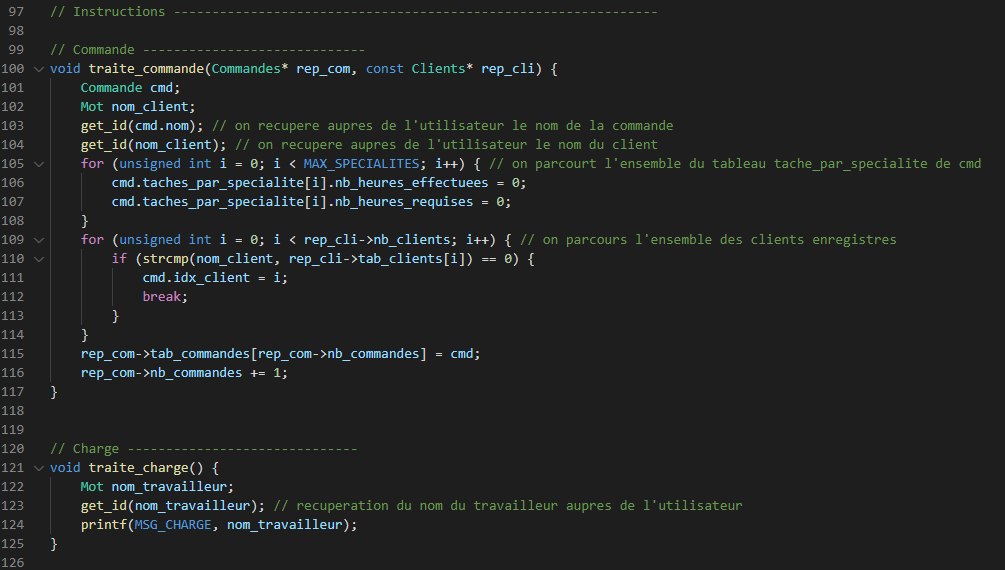


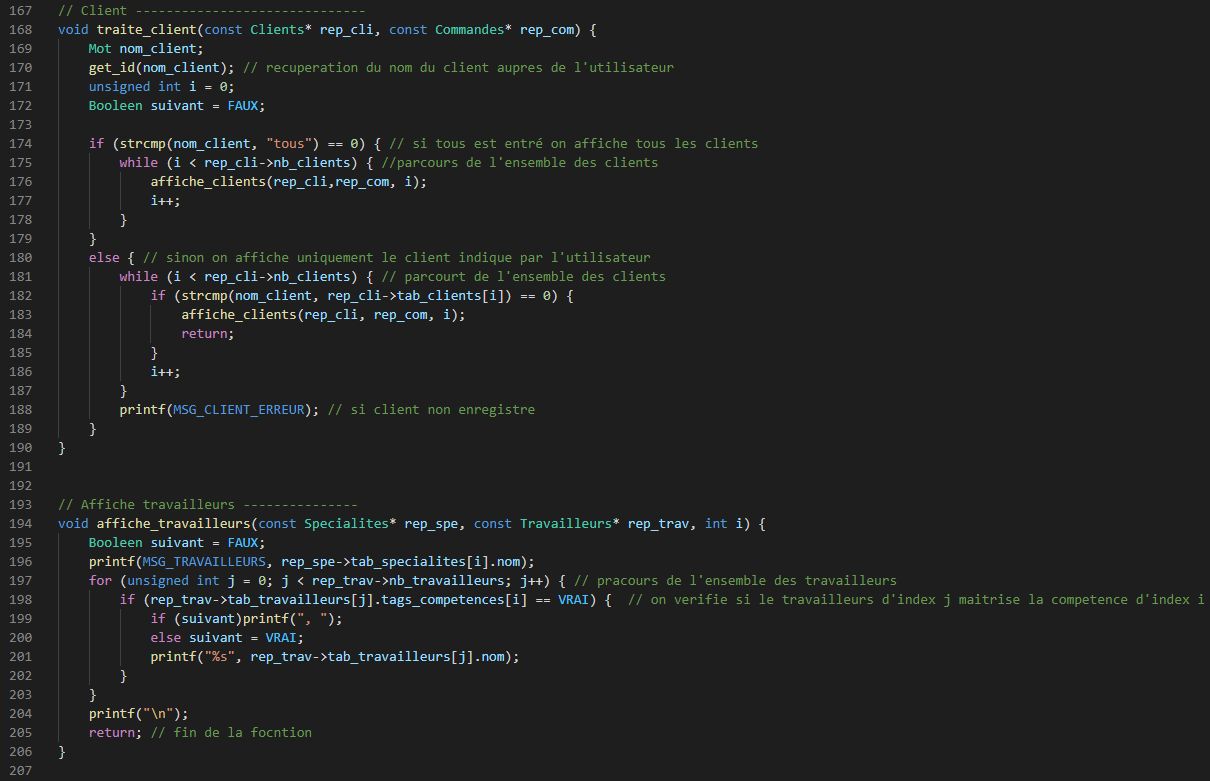


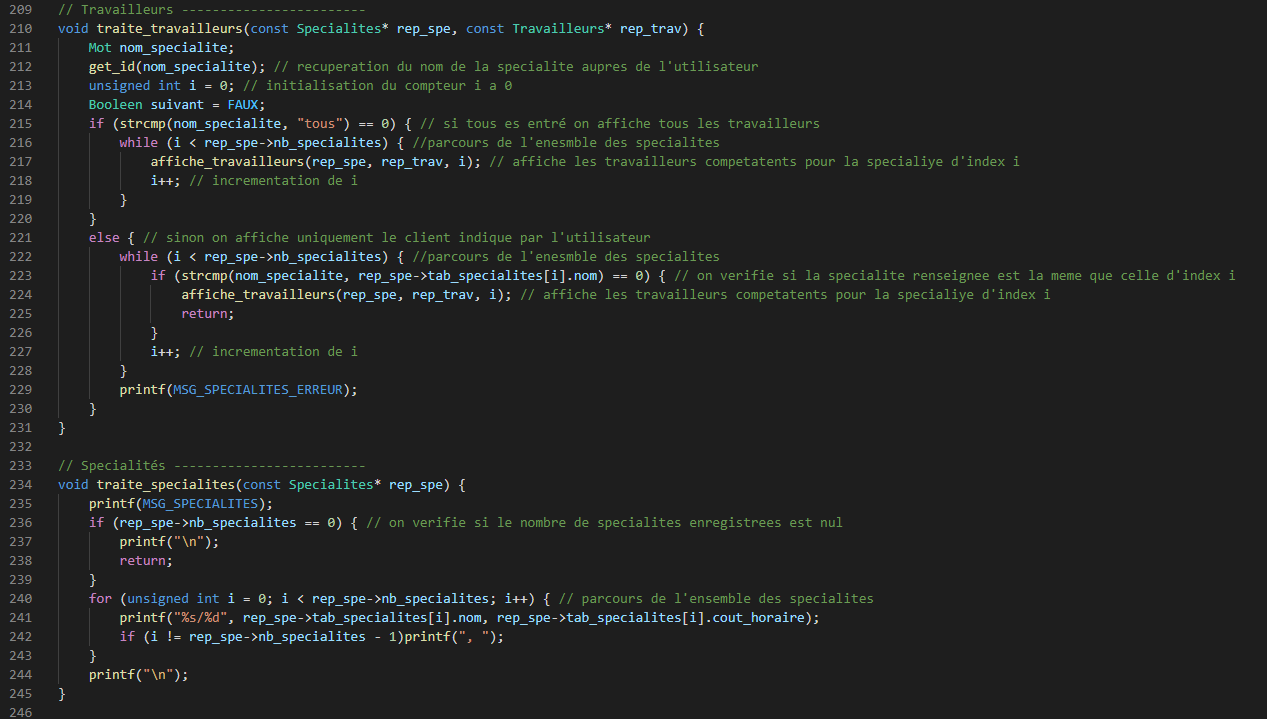
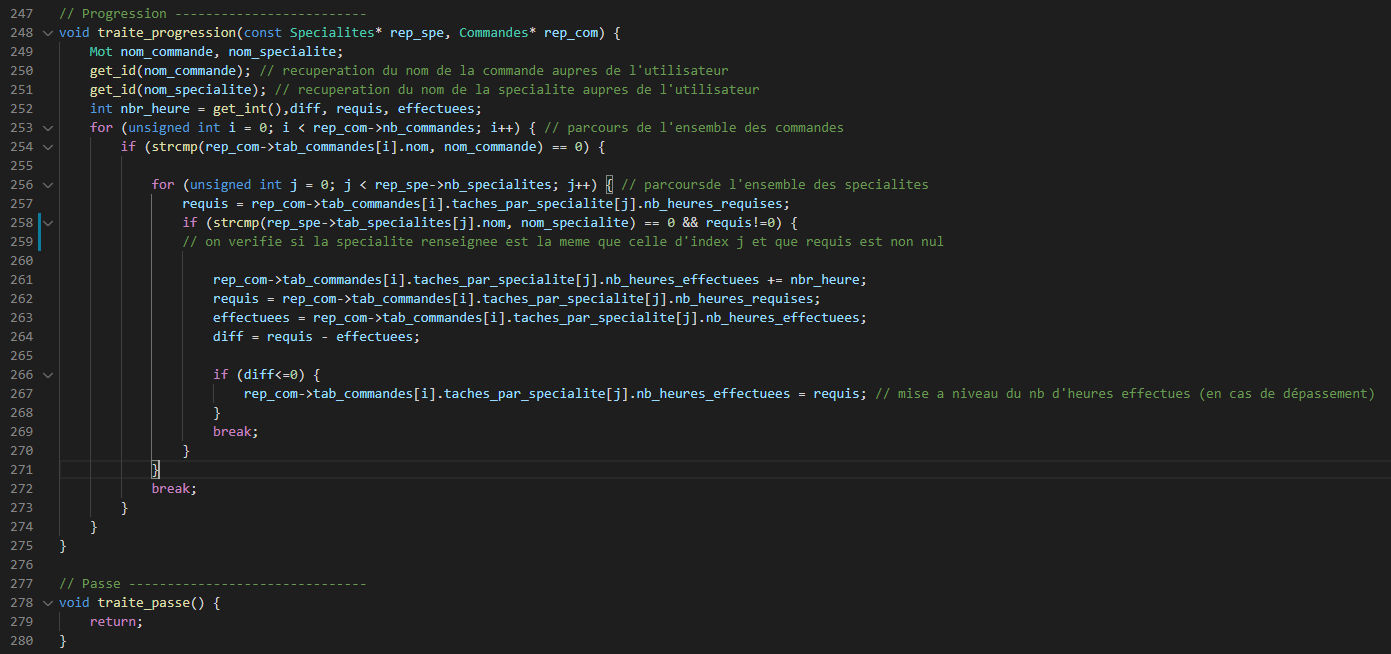


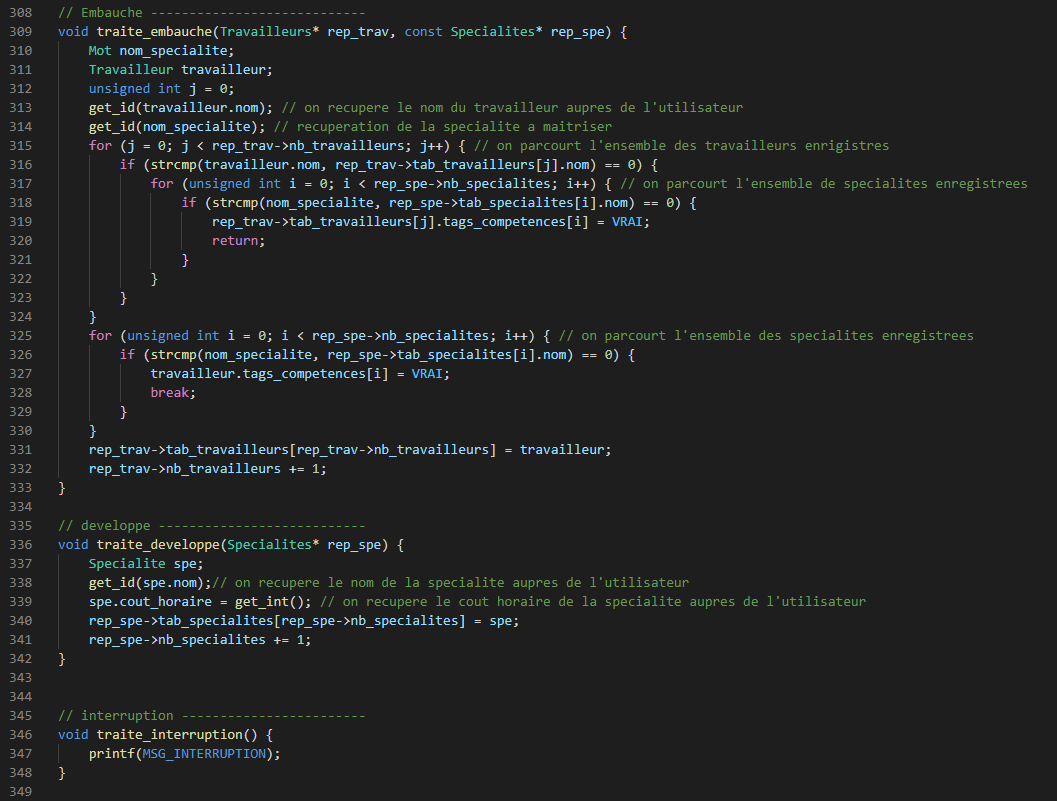


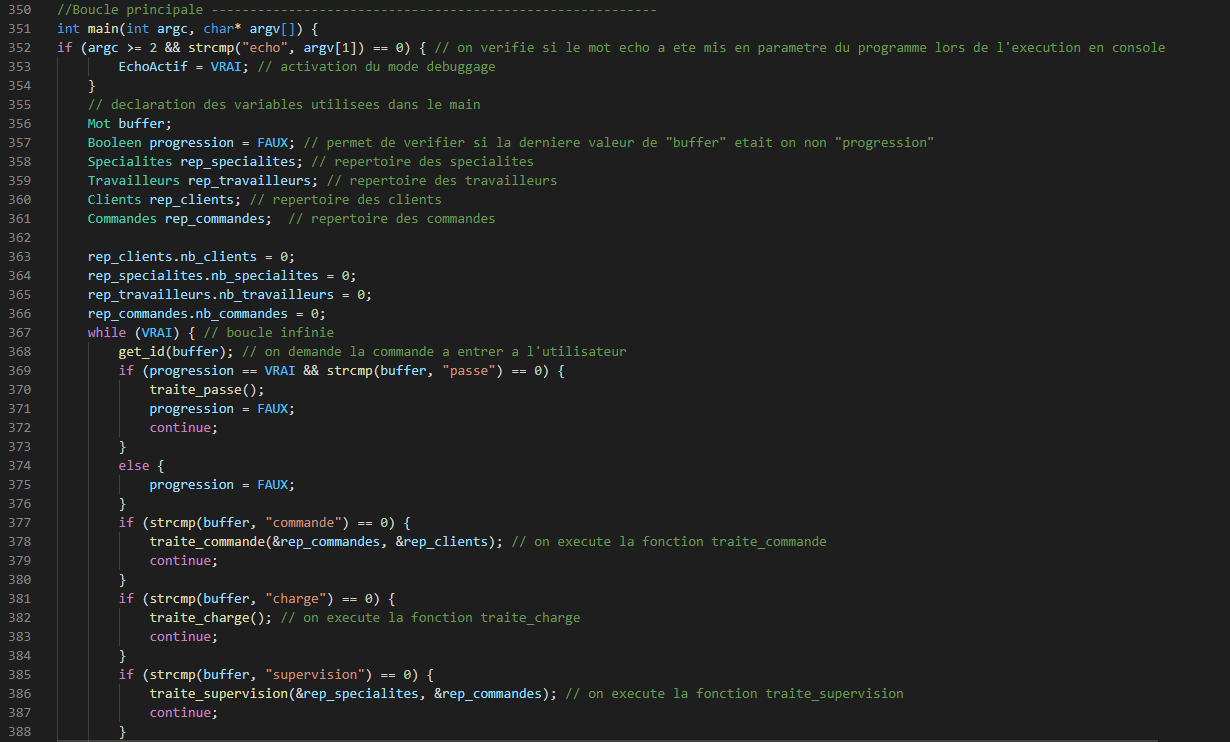
## Sprint 3 :





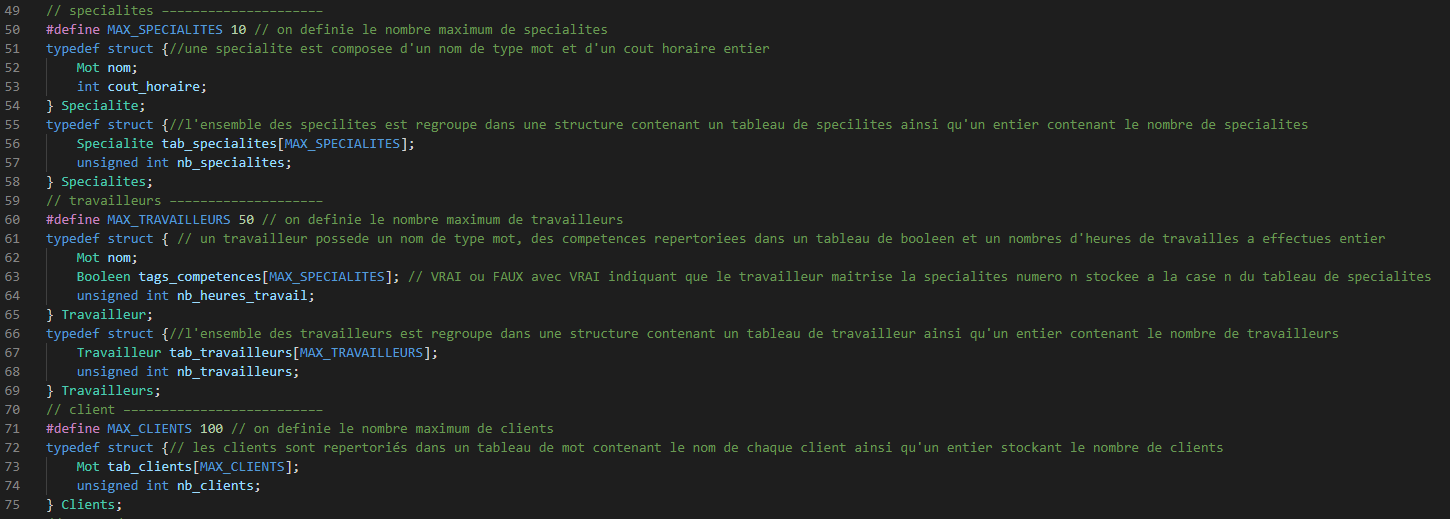
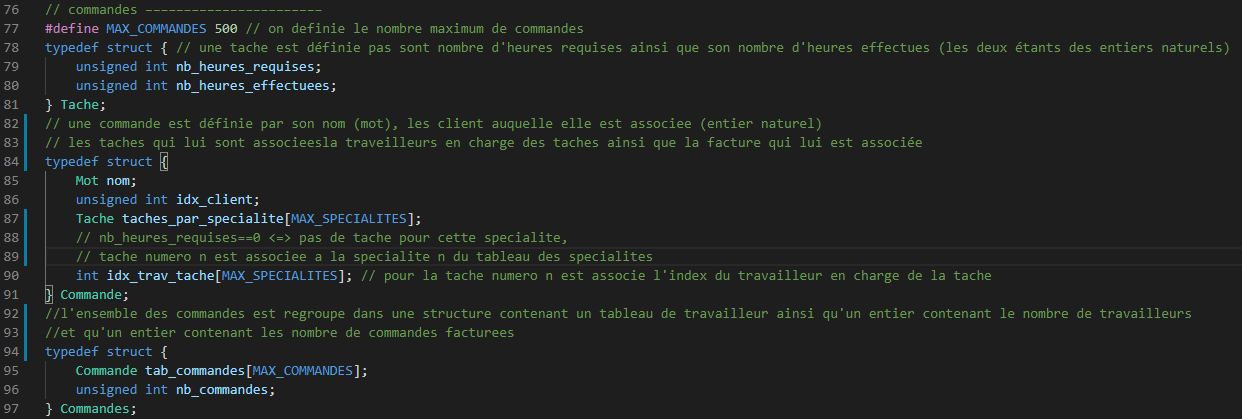


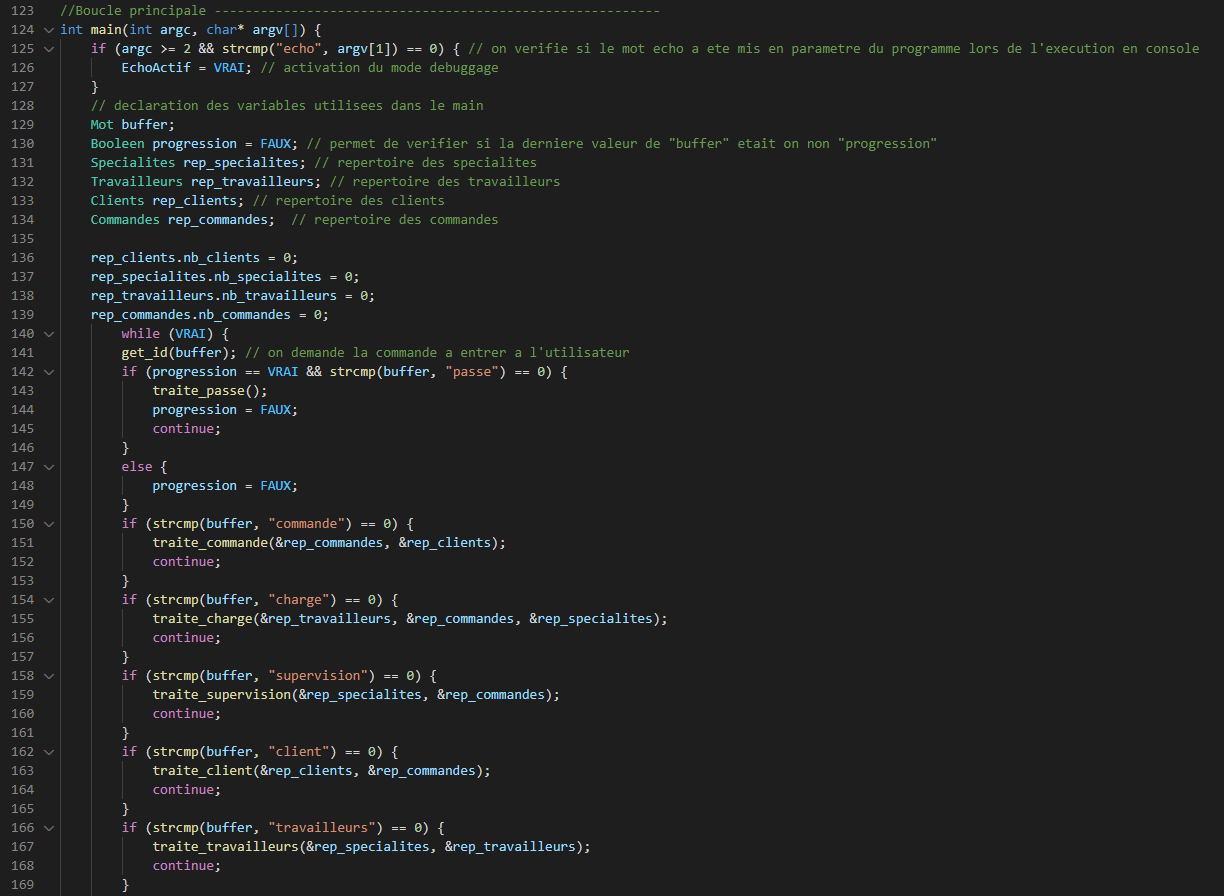
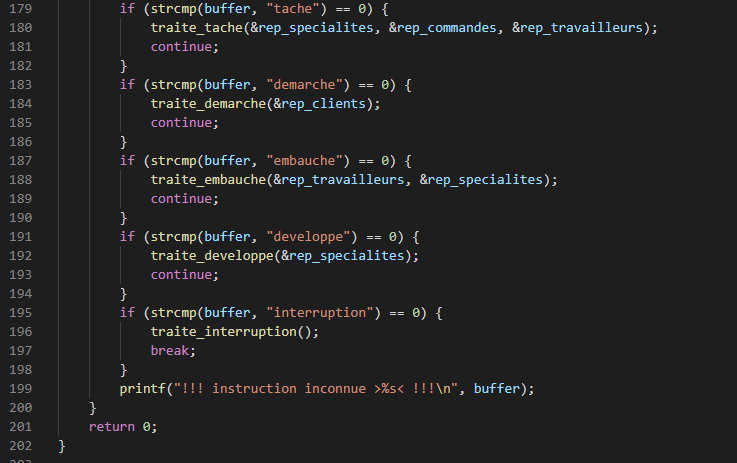




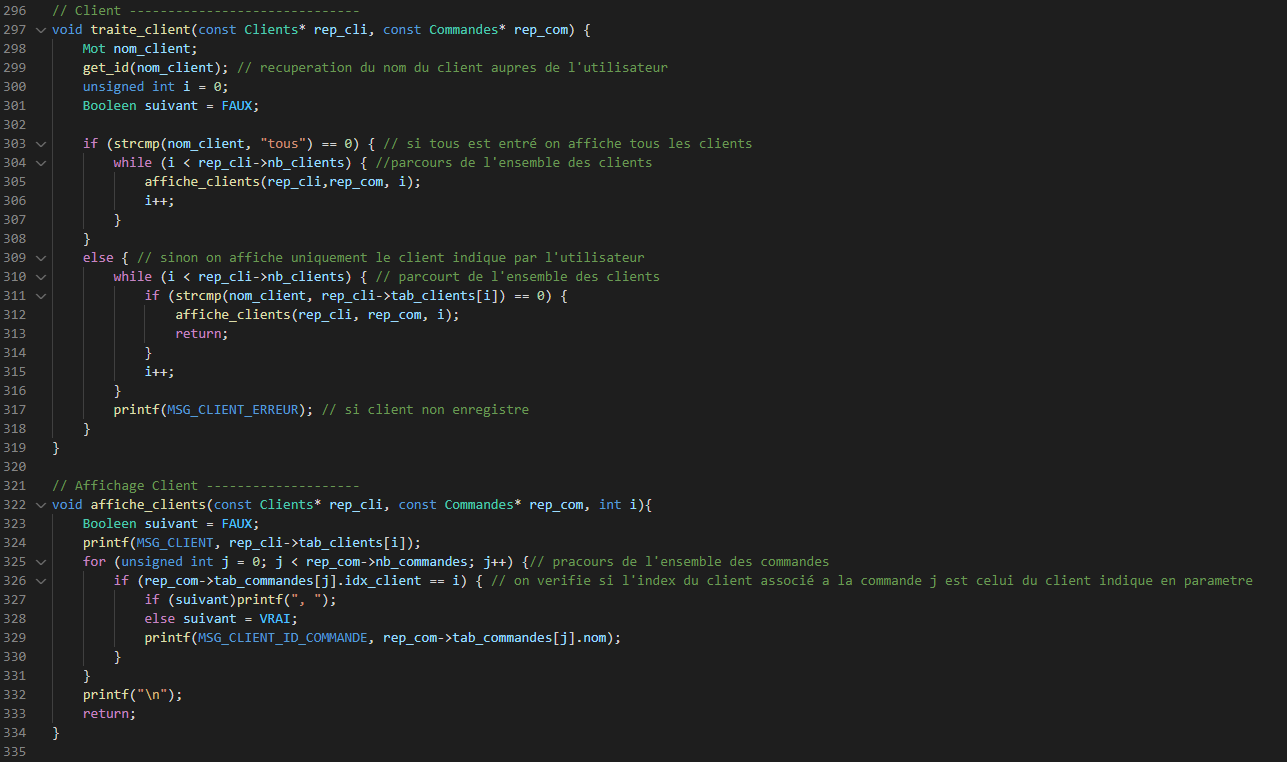
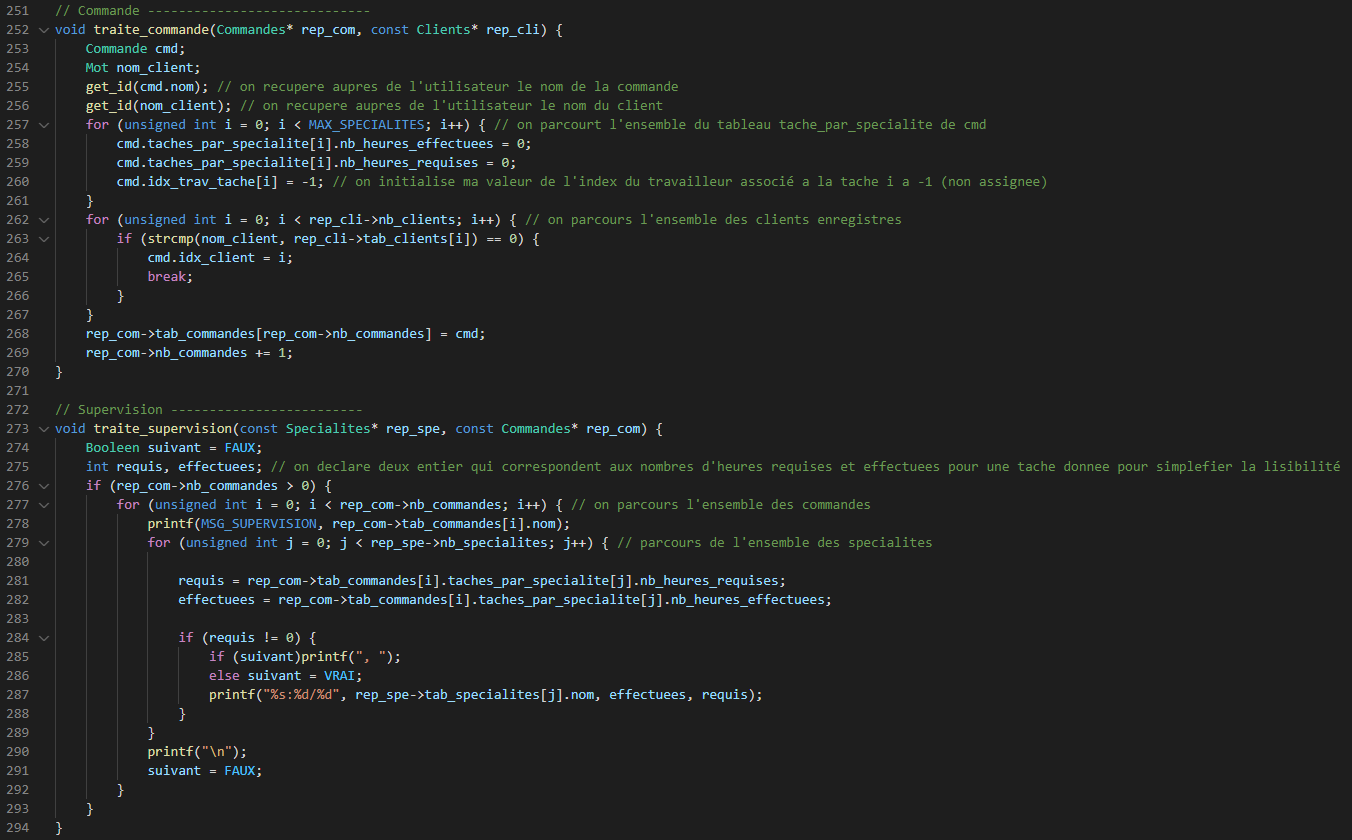


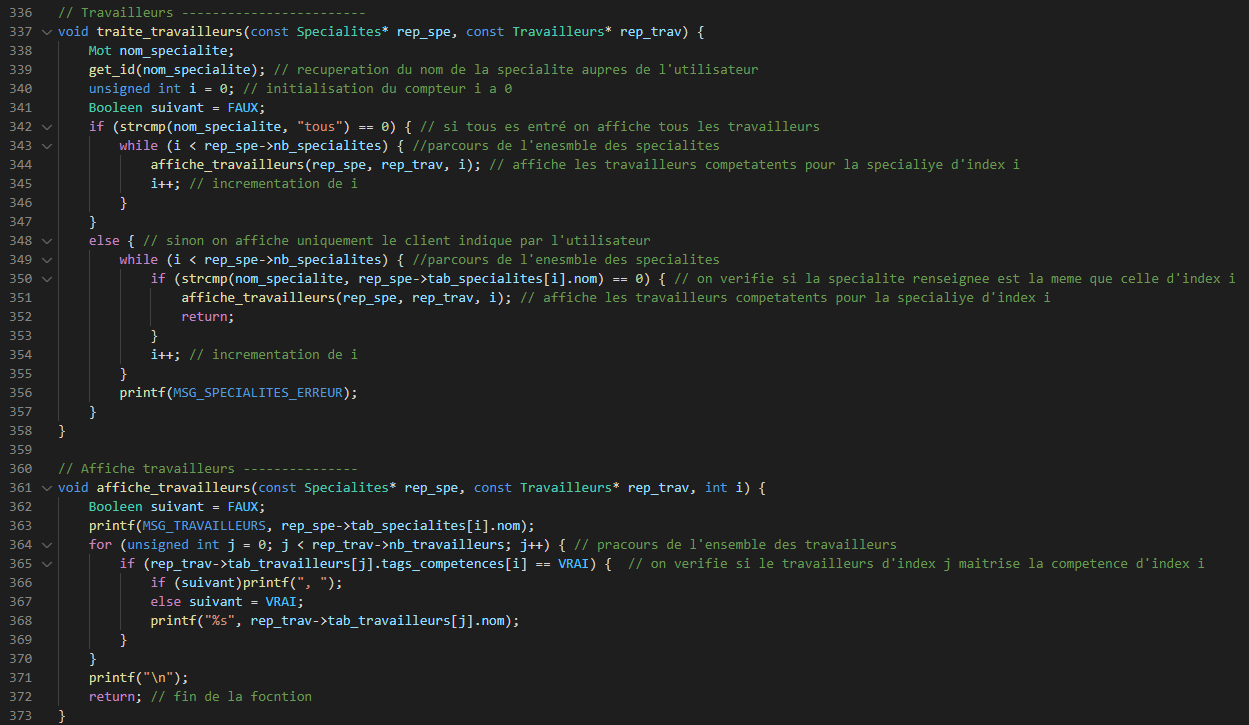
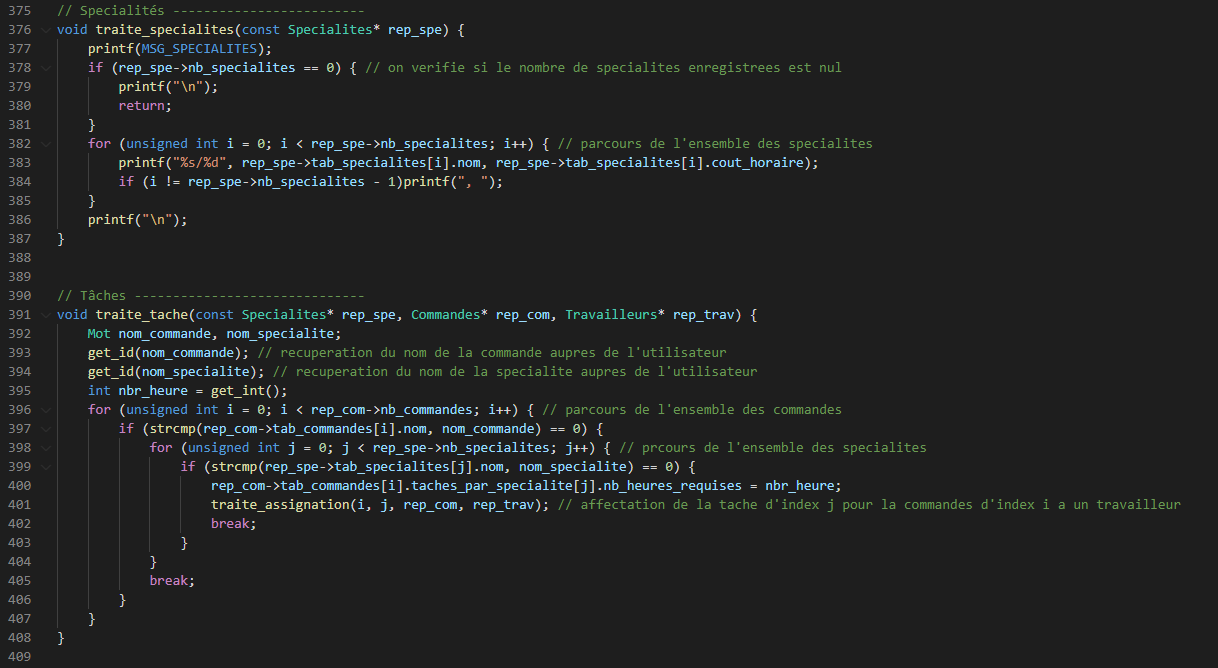
## Sprint 4 :

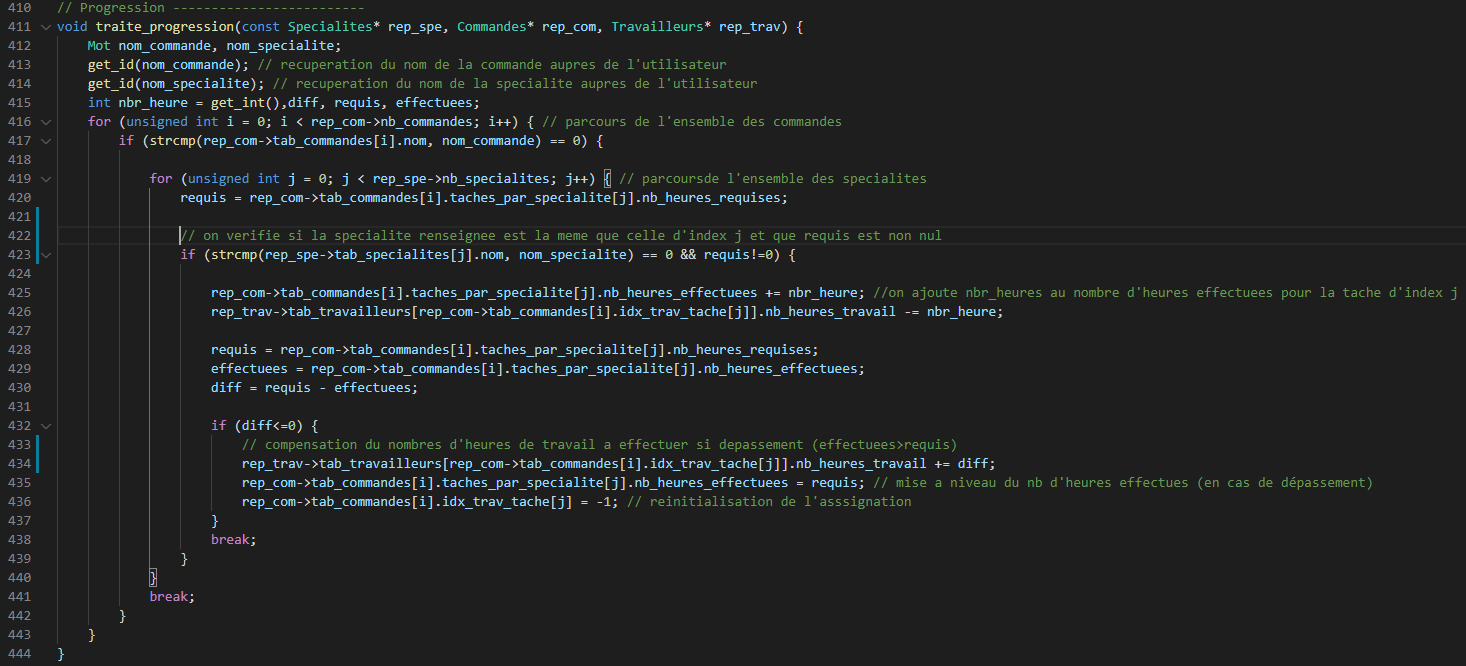
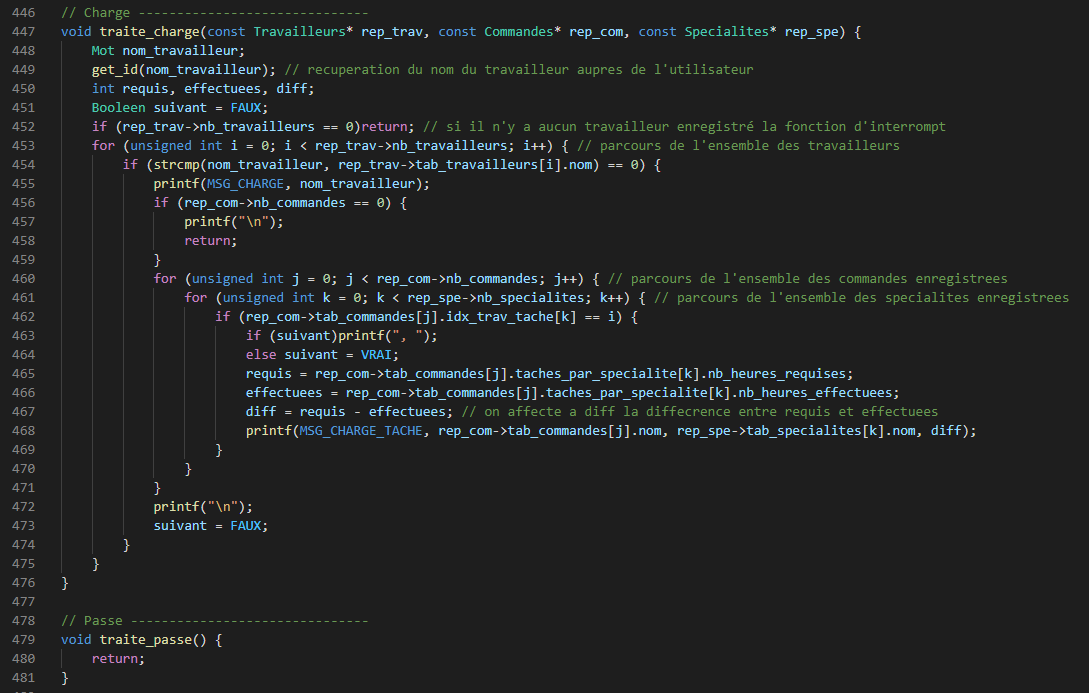


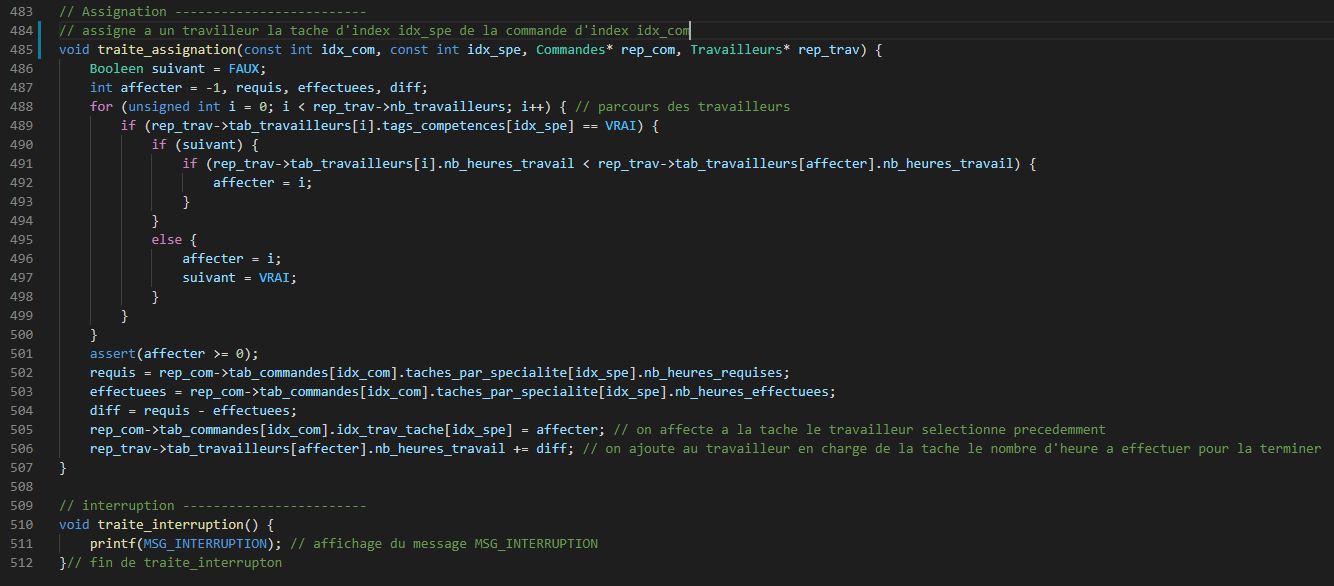




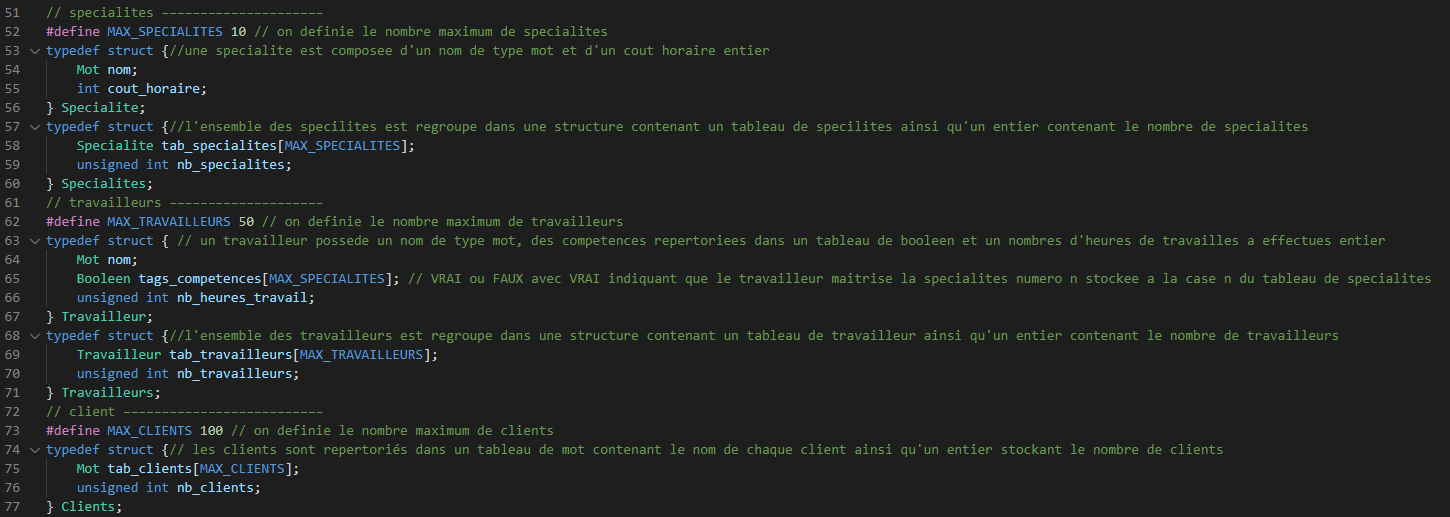
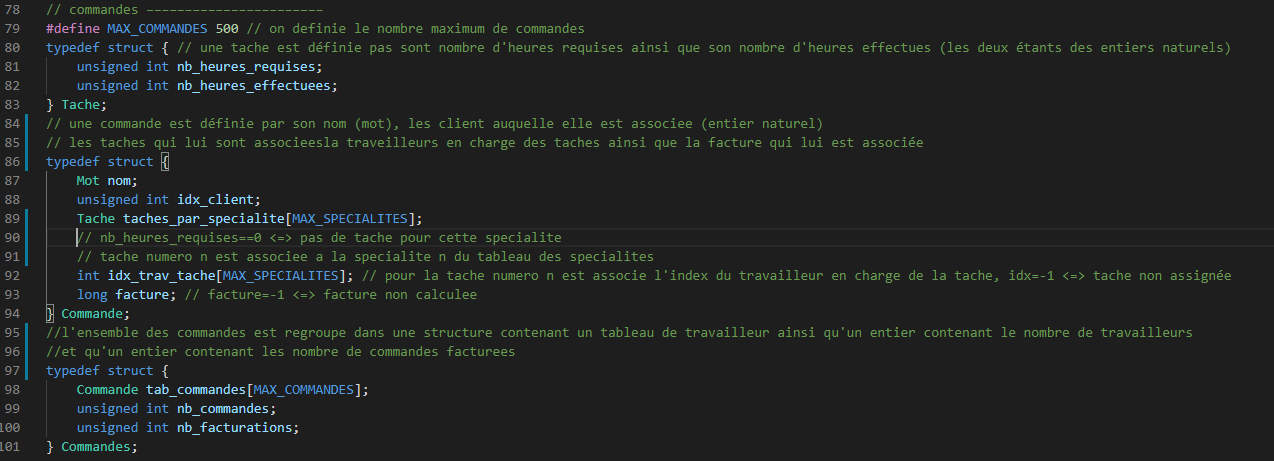


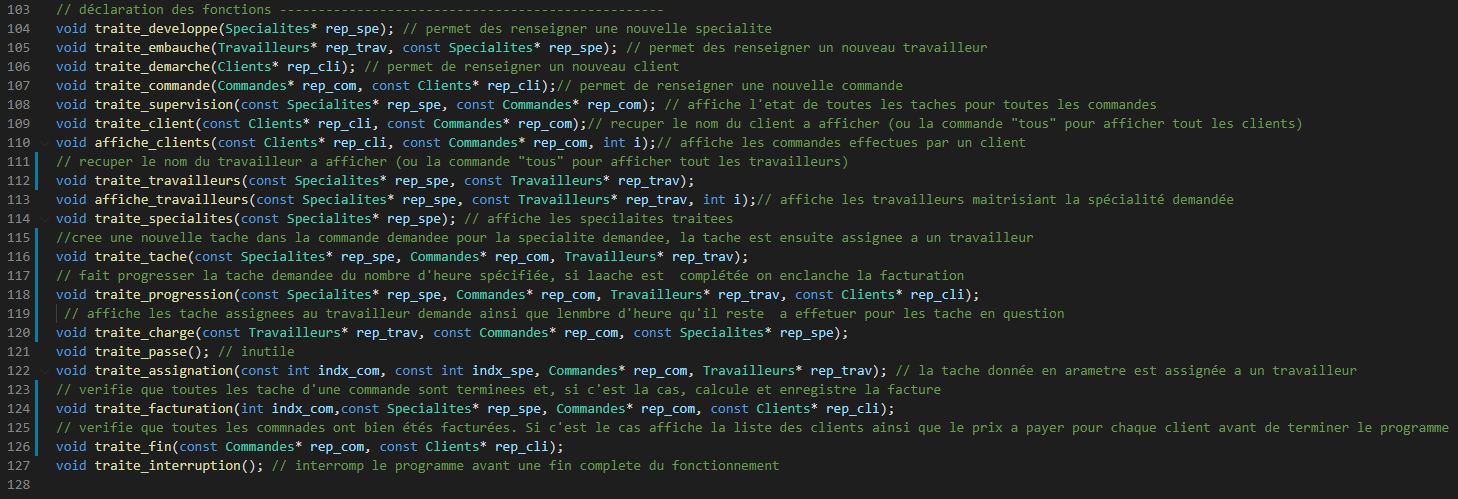
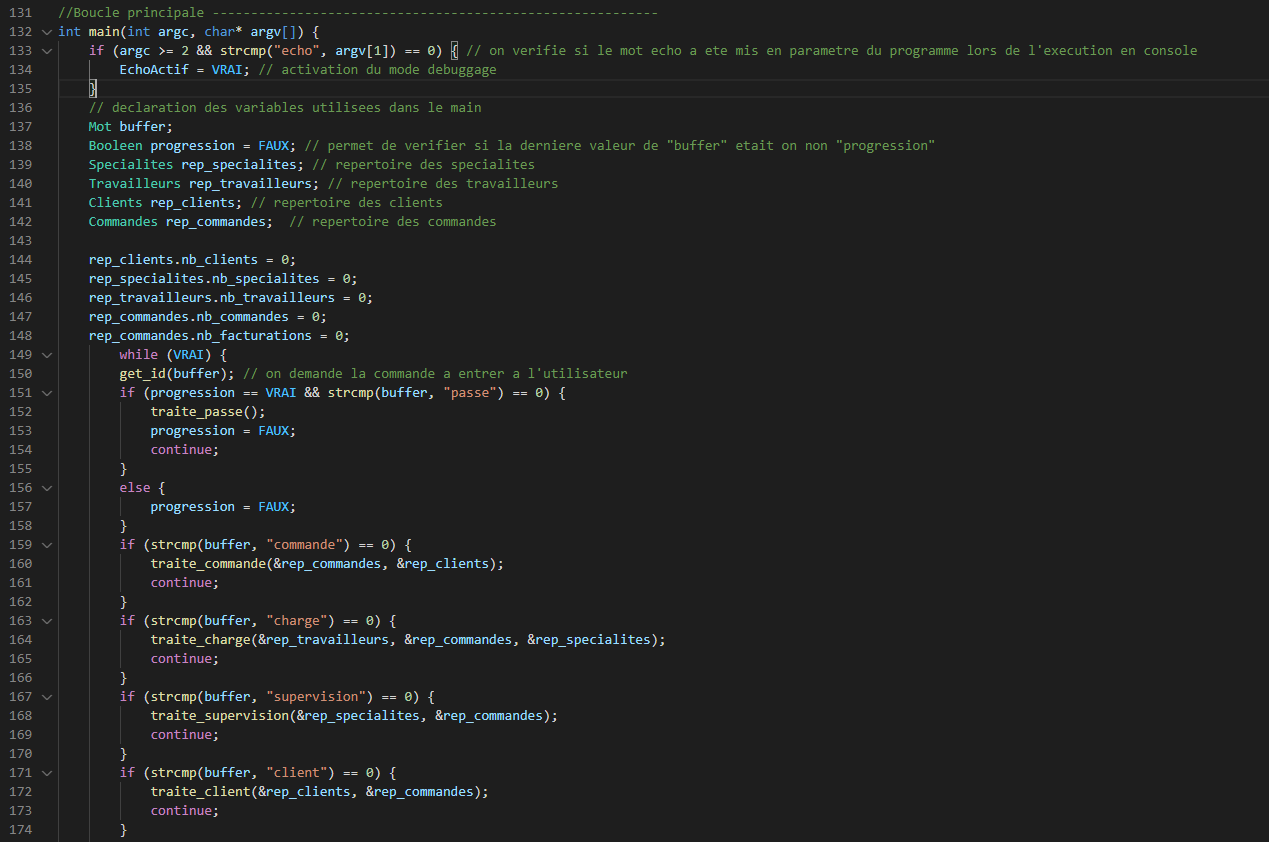






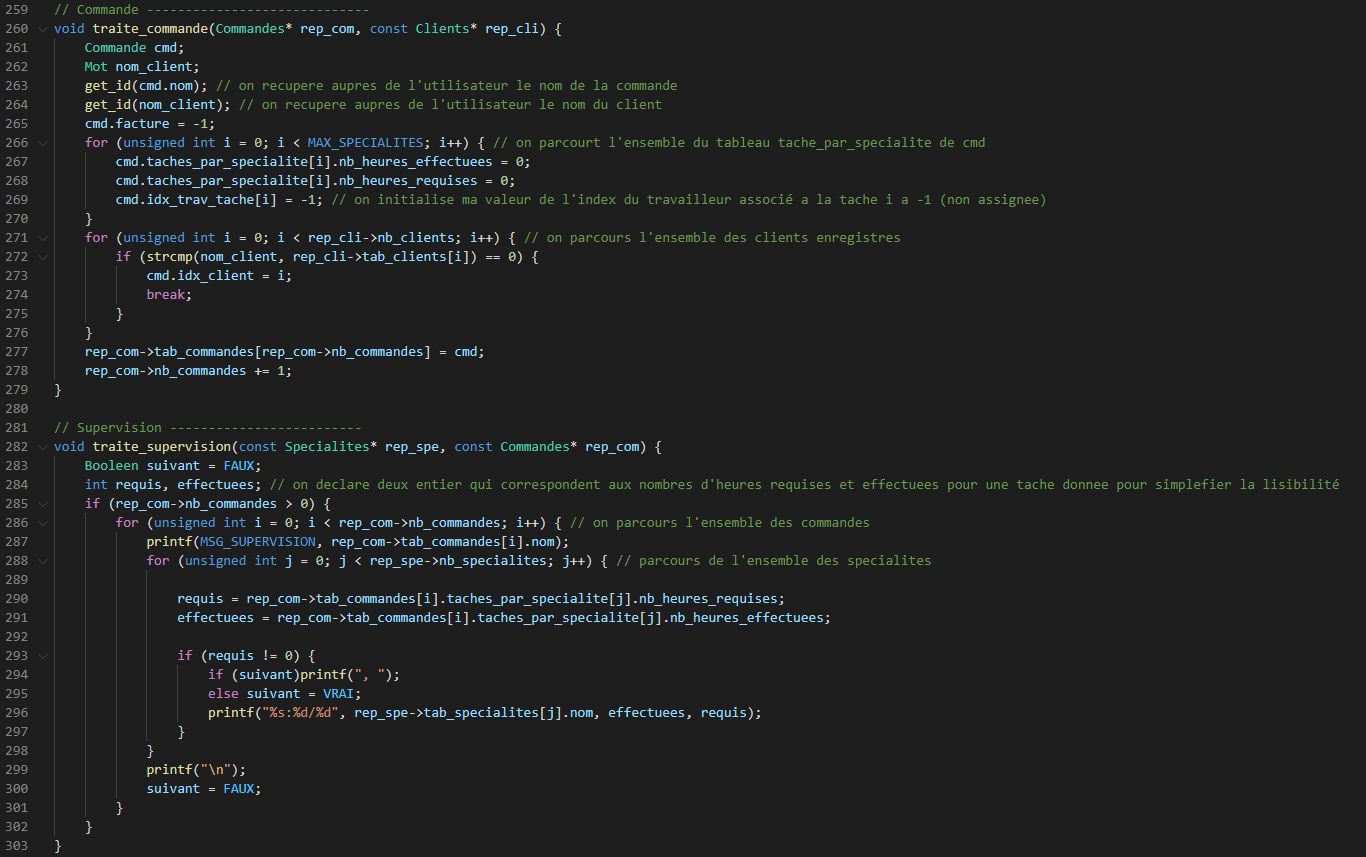
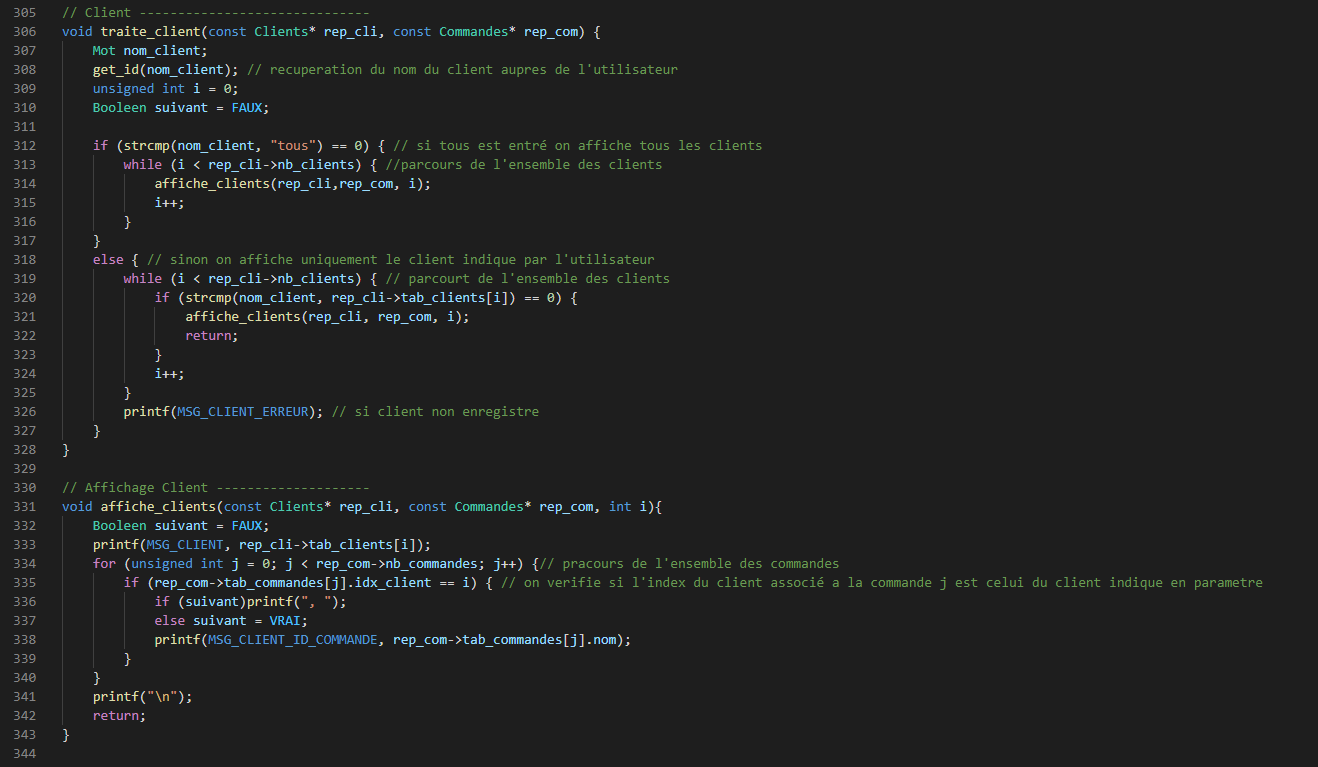
## Sprint 5 :

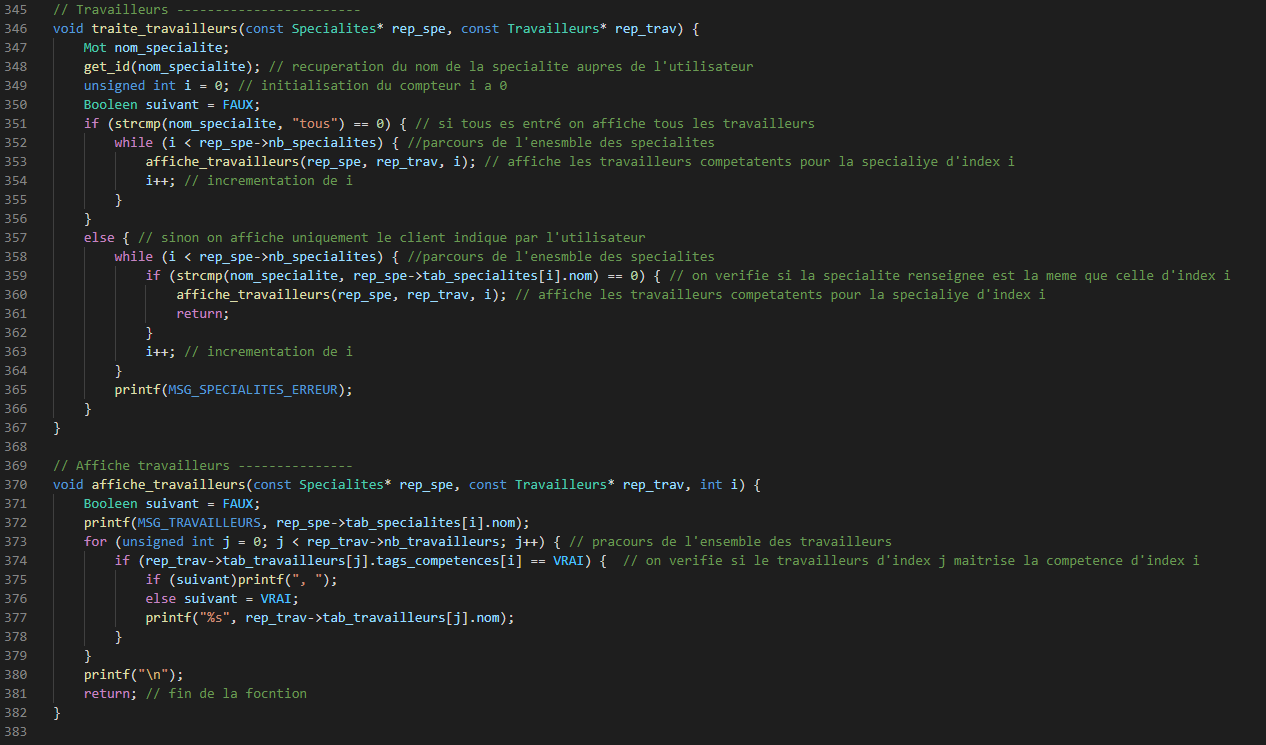
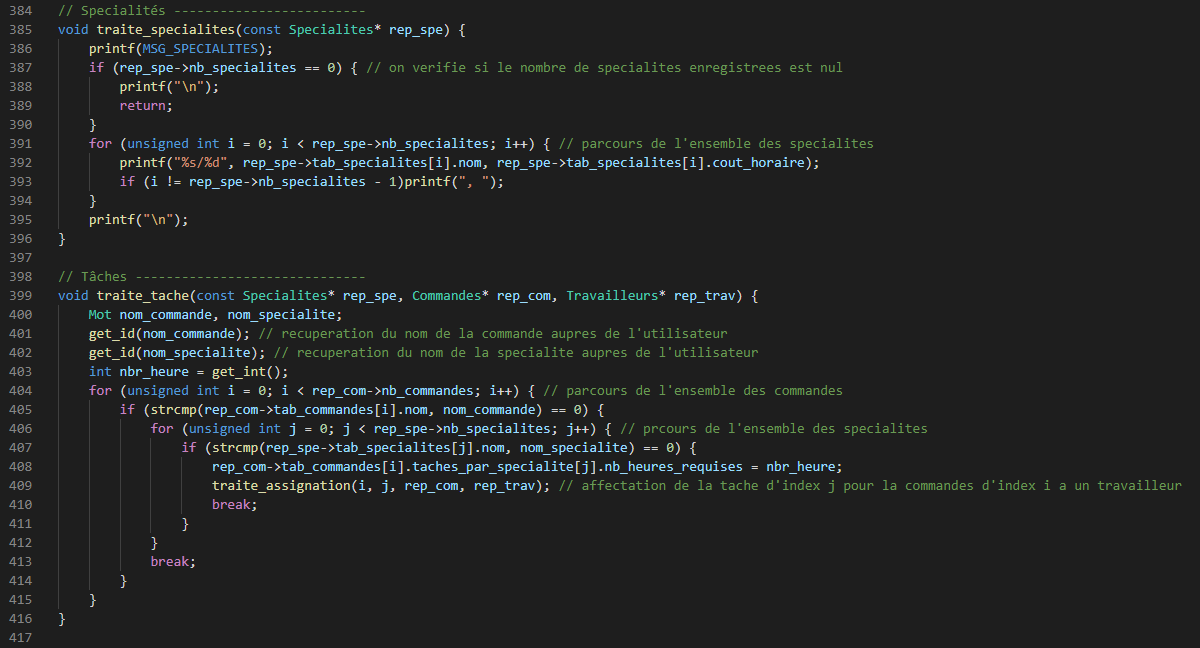


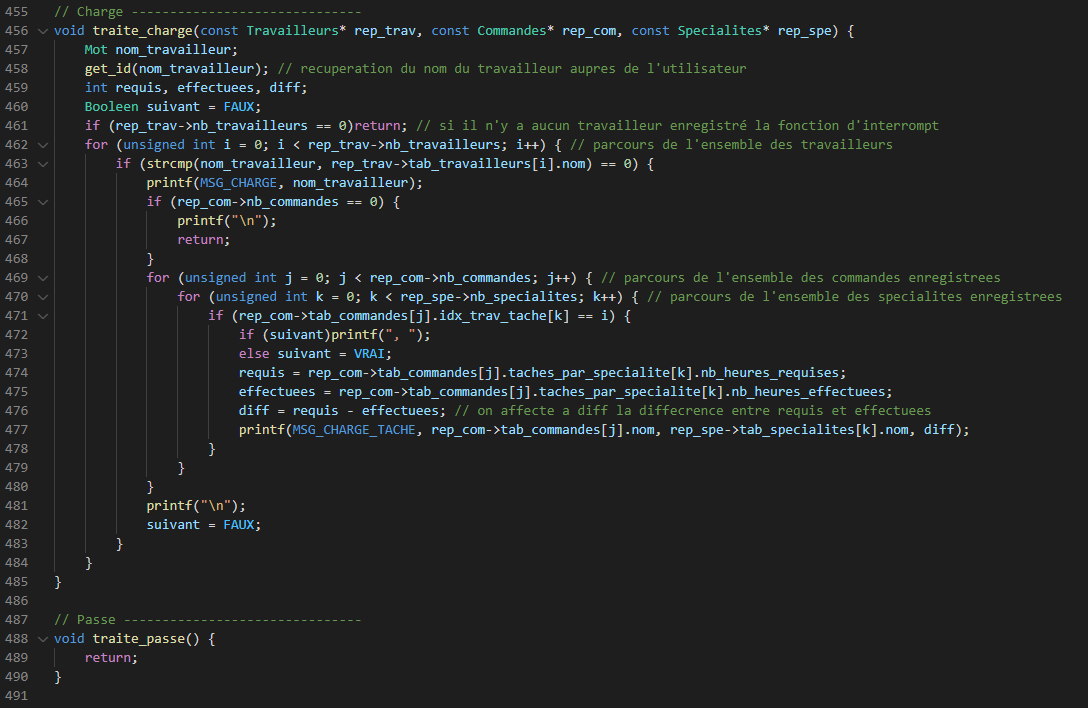


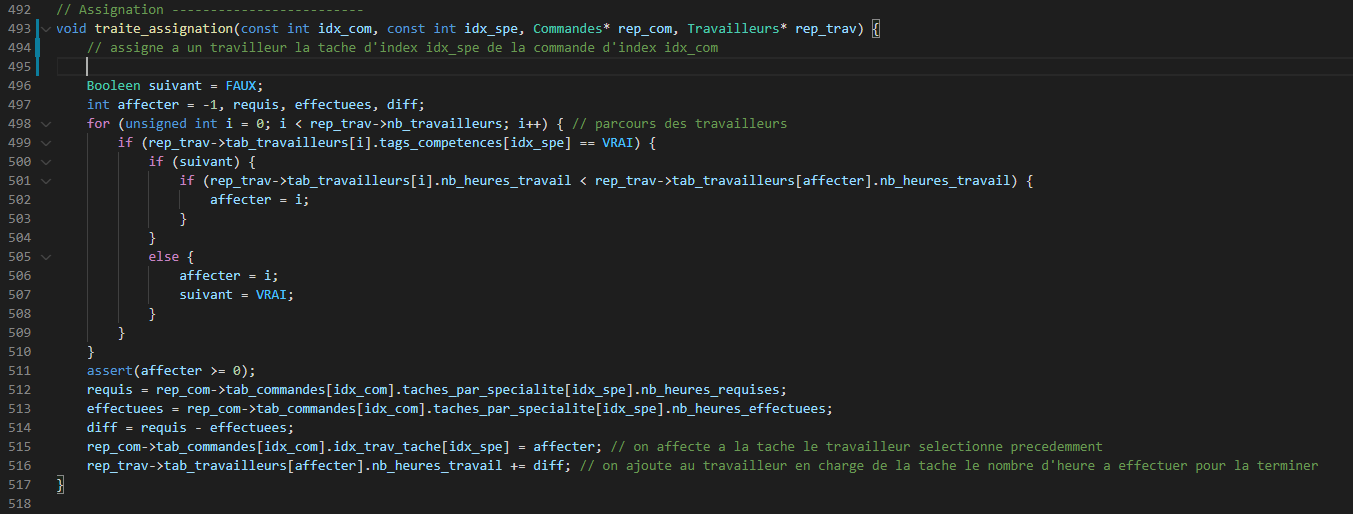
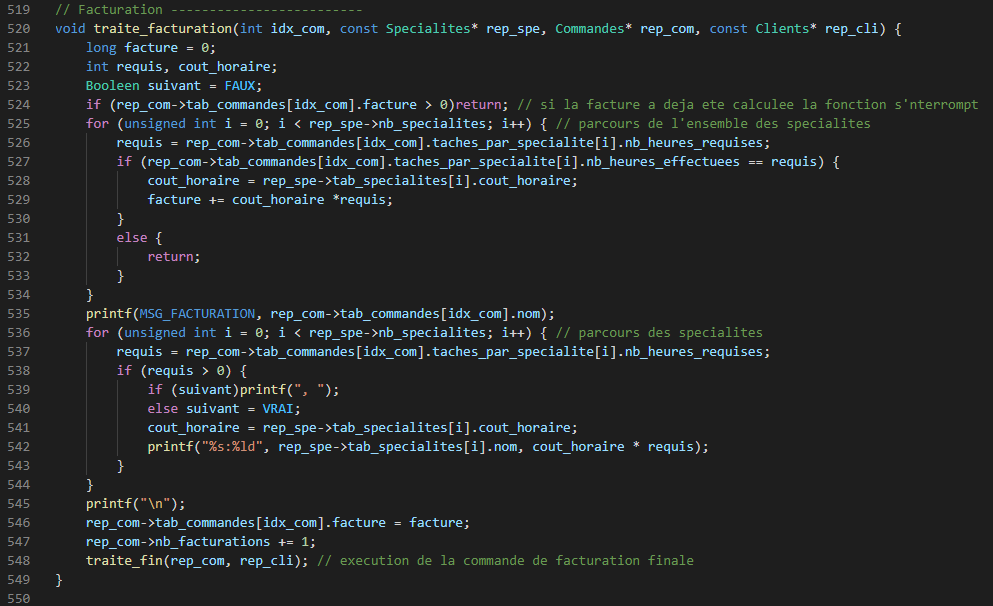


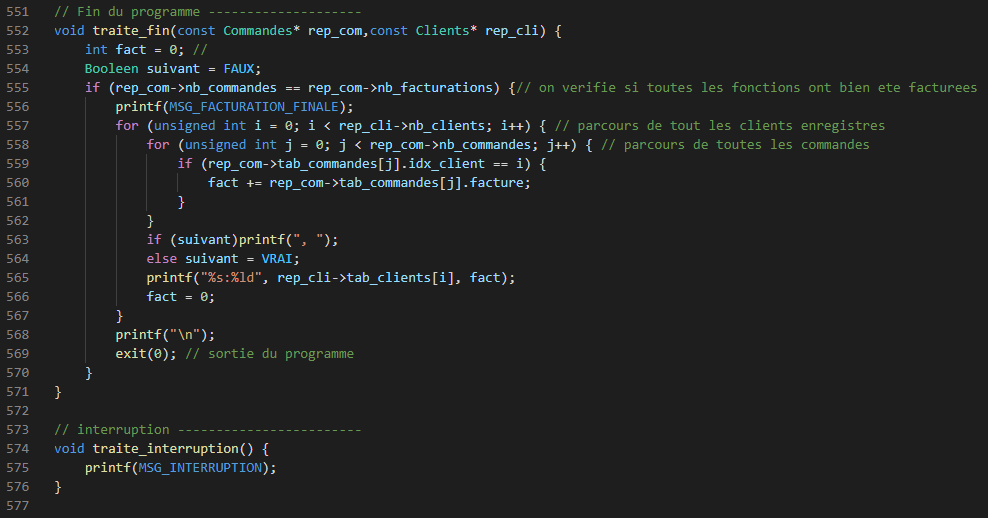












## Sprint 6 :

