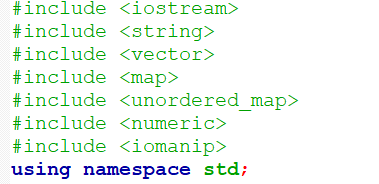
**RAPPORT TECHNIQUE : PROJET 12**

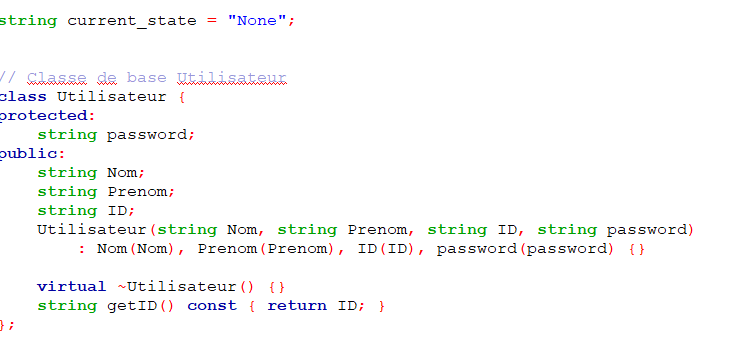
**Rédigé par Omar EL GHAZAL**

**Codage de l’ERP « Edutrack »**



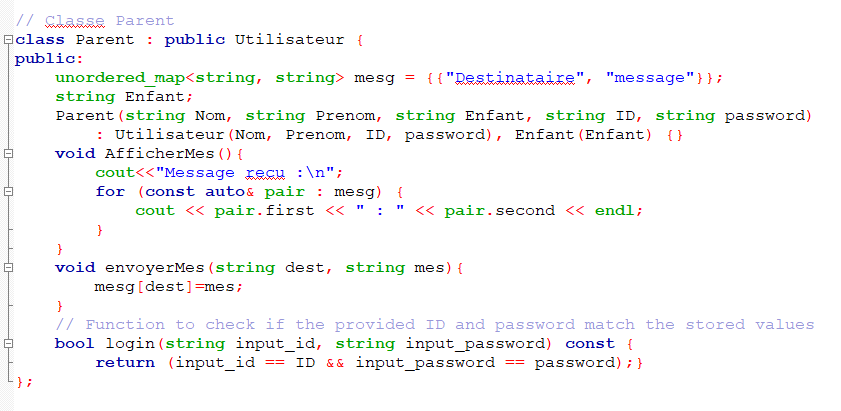
Tout d’abord, on initialise les biblioteque a l’aide de « include ». On ajoute :

Iostream (biblioteque de base), string (pour les chaines de caractere), vector (pour les vecteurs), map et unordered\_map (pour les dictionnaires), numeric et iomanip(pour les decalage de texte et formatage de texte). La variable « current\_state » servira pour changer letat (espace parent, eleve, page d’accueil).

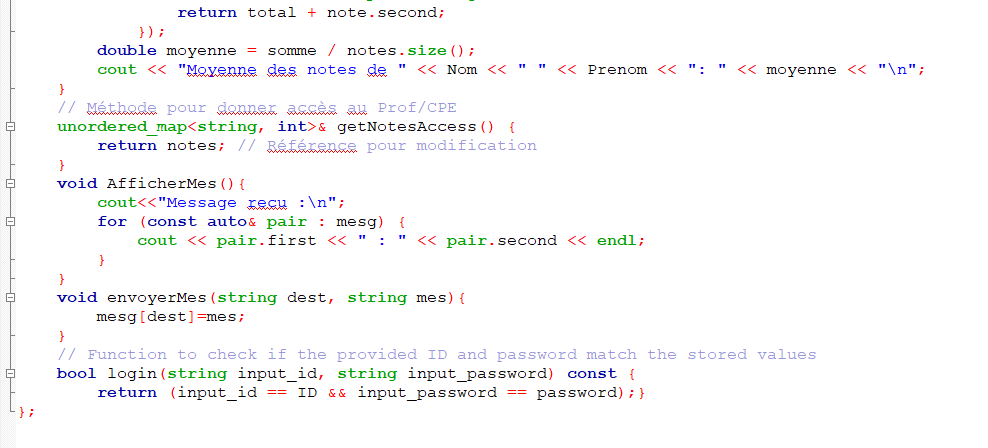
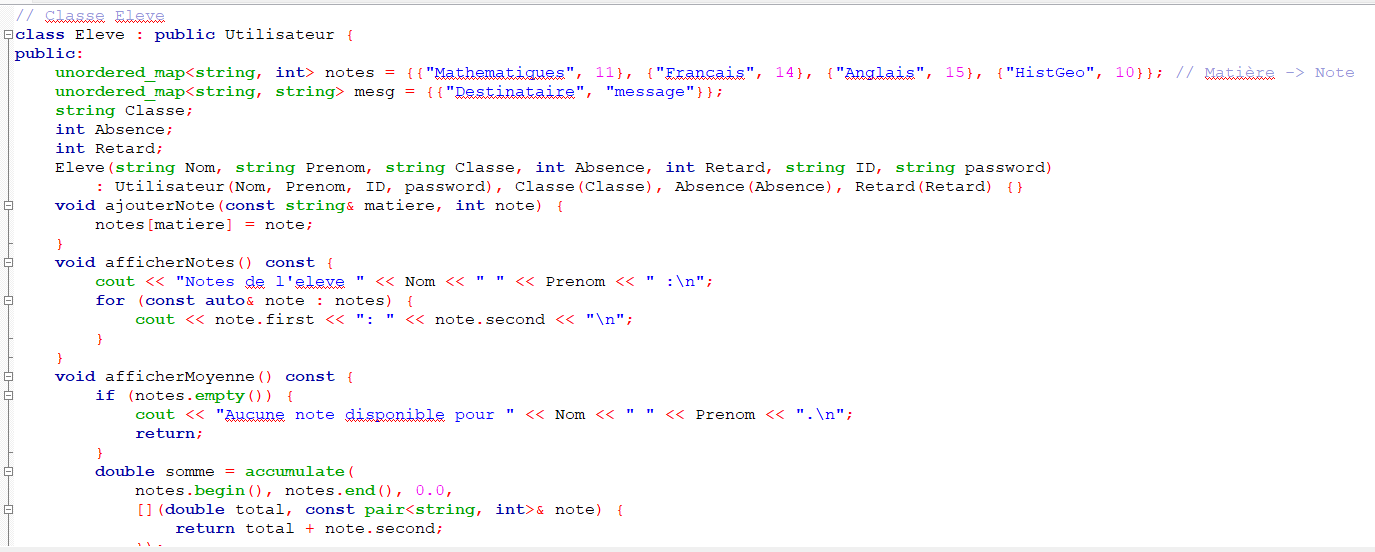


On cree une classe utilisateur avec les attributs nom, prenom et id (attribut public accessibles) et password (attribut proteger, accès limité).

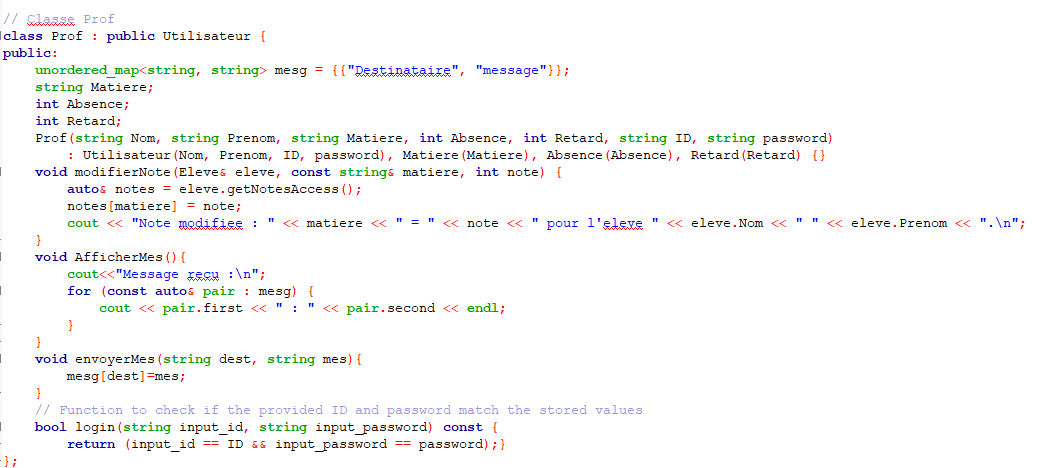
On initialise également un construteur Utilisateur.



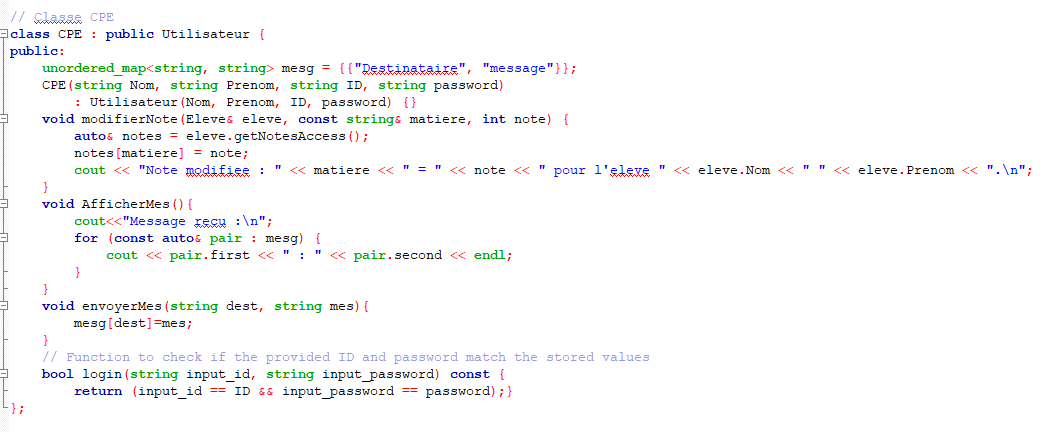
Ensuite on cree la classe parent avec comme attribut enfant (nom du fils ou fille du parent) . Cette classe herite de la classe utilisateur. Ensuite on initialise les methodes AfficherMes(pour afficher les messages reçus), envoyerMes (pour envoyer des messages) et la methode login qui permettra plus tard de verifier que l’utilisateur est connecté. On ajoute aussi un dictionnaire pour stocker les messages (clé : destinataire et valeur : message)



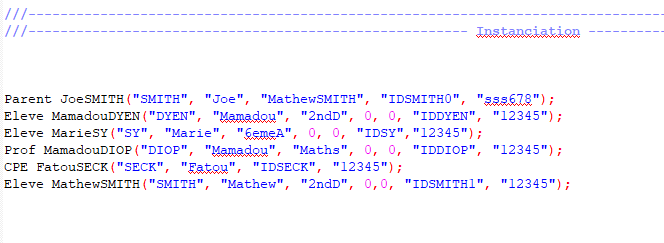
Enuite on cree la classe Eleve avec les attributs classe, absence, retard. On initialise aussi les methodes ajouterNote (pour ajouter des notes a leleve), afficherNote, afficherMoyenne (qui calcule la moyenne et l’affiche) afficherMes et envoyerMes et login. On ajout un dictionnaire pour les notes et un autres pour les messages.



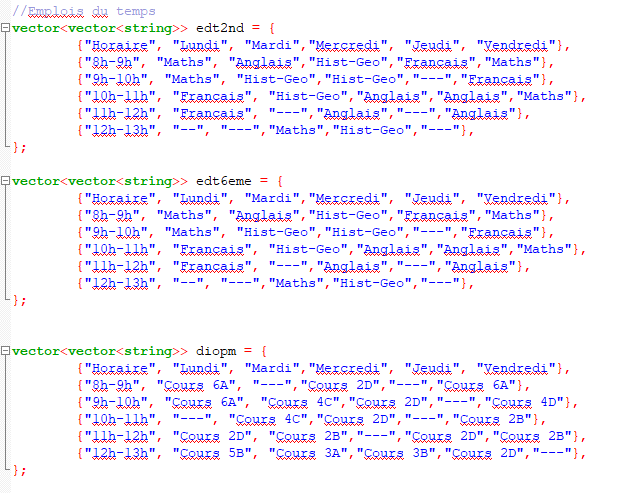
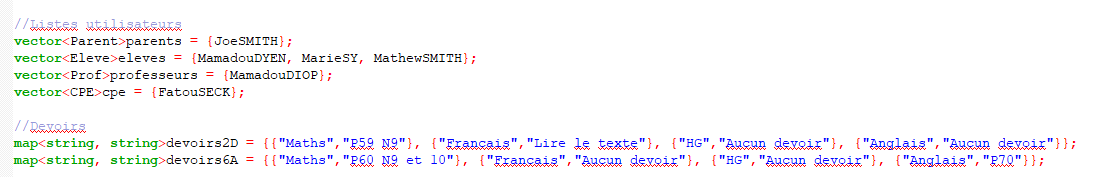
Ensuite on cree la classe Prof avec les attributs matiere, absence et retard. On ajoute aussi le dictionnaire de message. On ajoute les methode pour modifier les notes, afficher les messages et envoyer les messages ainsi quela methode login.



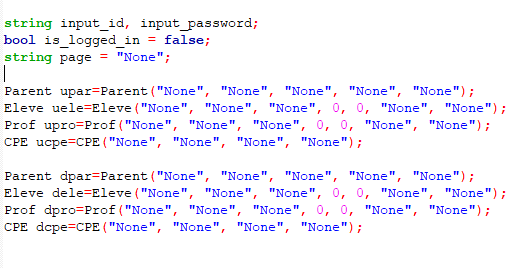
Enfin, on cree la classe cpe qui a pour attribut un dictionnaire pour les messages, les méthodes pour modifier les notes, afficher les messages, envoyer les messages et la methode login.



Ensuite, on instancie a l’aide des constructeurs definit dans les classes.

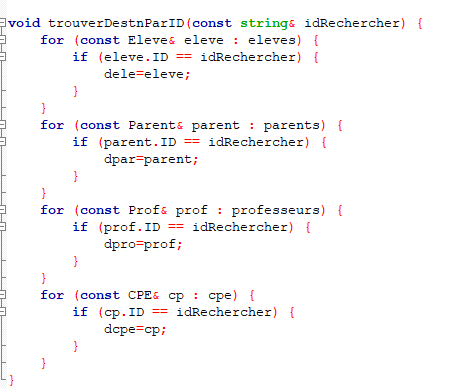


Ensuite, on stocke les instances (parents, eleves etc) dans des vecteurs. Puis on stocke les devoirs par classe dans des dictionnaires. Ils nous serviront plus loin.

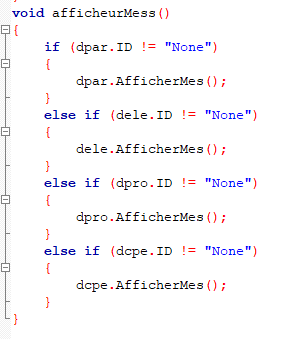


On cree des variable input\_id et input\_password (qui permettront de verifier si les identifiants sont corrects), un booléen is\_logged\_in pour savoir si l’utilisateur est connecté ou pas, et une variable page pour savoird dans quelle page se trouve l’utilisateur.

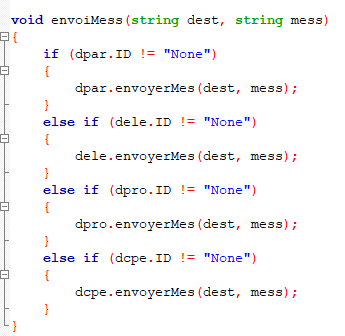
Ensuite, on cree des « variables » pour chaque categorie (parent, prof etc). Cela nous servira plus tard (pour les check sur les attributs ou bien pour selectionner un utilisateur).



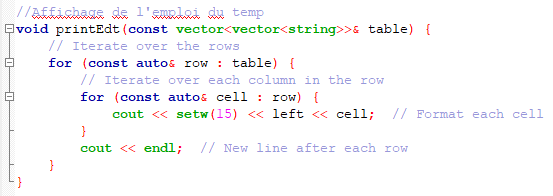
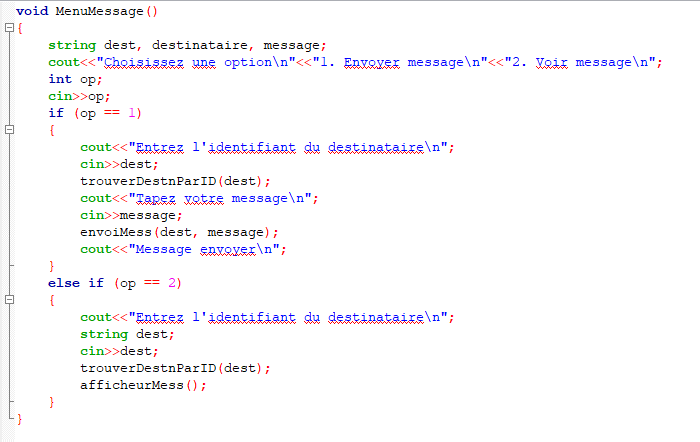
On cree une fonction qui permet de trouver le destinataire d’un message par son ID. Elle prend en parametre l’identifiant et check sa validité en realisant des egalité sur les utilisateurs de chaque liste.



La fonction ci-dessus permet d’afficher les messages d’un utilisateur (elle check si un utilisateur est sélectionné sachant qu’il est stocké dans une des variables qu’on a créer precedement).

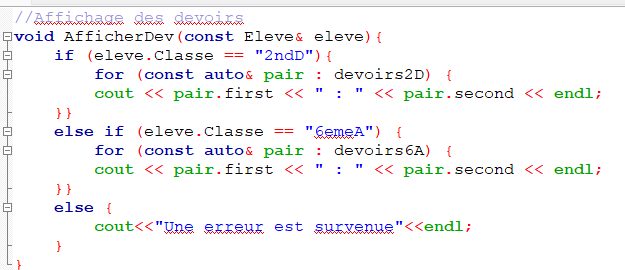


Ensuite, on crée une autre fonction pour envoyer des messages. Elle prend en parametre le nom du destinataire et le message. Ensuite, on realise des checks pour savoir qui est le destinataire (comment le reconnaitre ? En effectuant un check sur l’ID de la variable. Si l’ID nest pas « None », cela signifie que la personne en question est le destinataire du message).

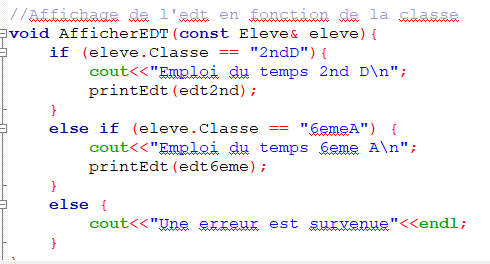


On cree une fonction qui nous permettra de choisir les options dans le menu « Messagerie ». Deux options seront disponible : l’envoi de message et l’affichage des messages reçus. Pour l’option 1, on entre l’identifiant du destinataire. Ensuite on fait appel a la fonction trouverDestParID qui va, en fonction de l’ID entré (a condition qu’il soit valide), stocker l’utilisateur en question dans une variable (la variable dele, dpro, dpar ou dcpe en fonction de la classe de l’utilisateur). Ensuite, on insere le message et on appel la fonction envoiMess qui enverra le message (en l’ajoutant au dictionnaire de message du destinataire). La deuxieme option suit le meme principe sauf qu’el affiche les messages à la place.

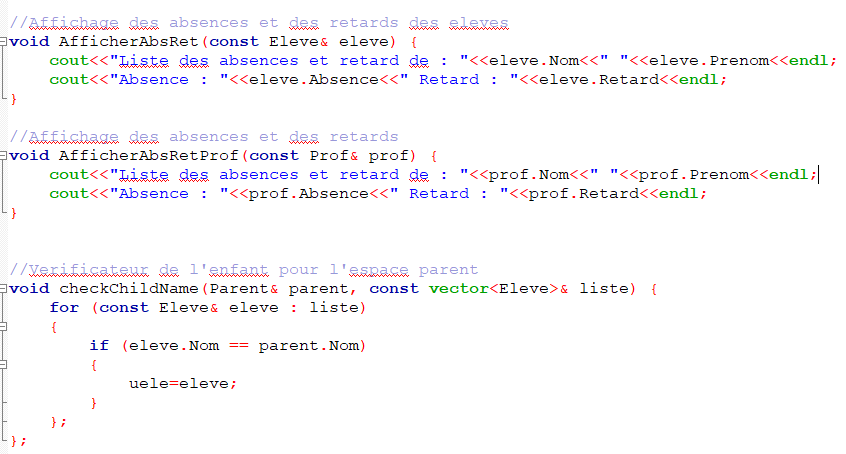
La fonction suivante printEdt affiche l’emploi du temps. On parours les cellules de la liste (emploi du temps sous forme de liste, défini precedement) et on affiche la cellule avec une marge de 15 pour afficher l’emploi du temps de manière organiser et lisible.



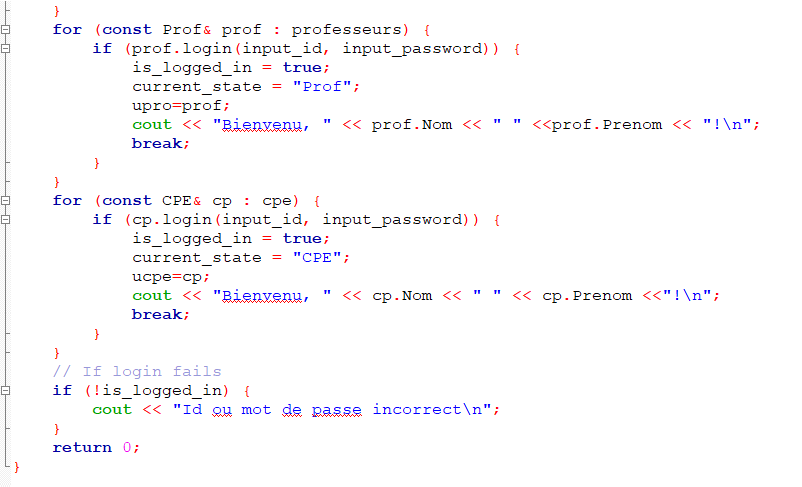
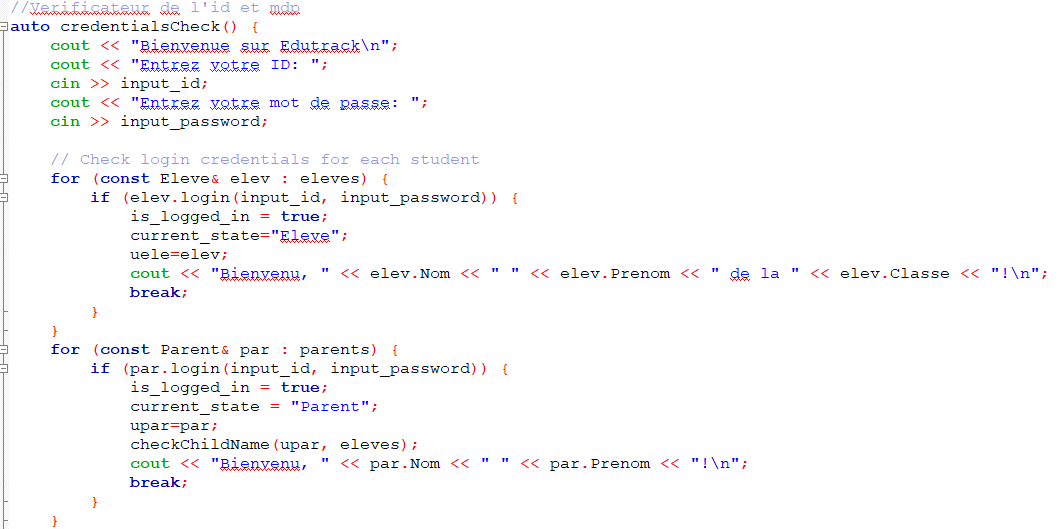
On cree une fonction pour afficher les devoirs. On affiche d’abord la clé du dictionnaire (sachant que, comme vu plus haut, les devoirs sont organisé sous forme de dictionnaire clé valeur avec comme clé la matiere et comme valeur le devoir a faire) ensuite la valeur. Les affichages se font apres un check pour connaitre la classe de l’eleve.



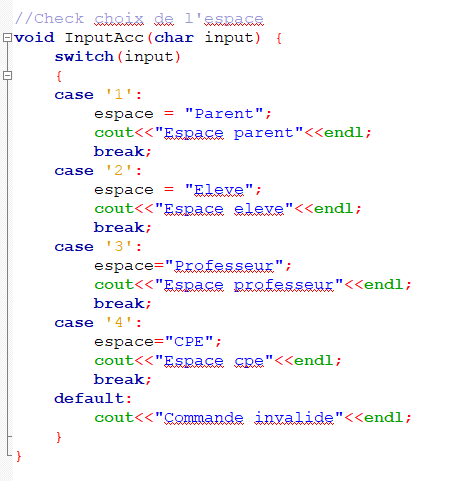
La fonction AfficherEDT affiche l’emploi du temps en fonction de la classe en appelant a fonction printEdt qui elle s’occupe du formattage de la liste sous forme de tableau(pour l’affichage dans la console).



Ici, nous avons 3 fonctions. Une pour afficher les absences et retard des eleves (affiche les attributs correspondant). Meme principe pour la fonction AfficherAbsRetProf. La fonction CheckChildName prend en parametre un objet de type Parent et une liste (liste d’eleves). Elle parcours les liste pour trouver un eleve ayant le meme nom de famille que le parent. S’il le trouve, se fils et stocker dans une variable. Cette fonction est d’une grande importance car elle permettra d’afficher l’emploi du temp du fils ou de la fille dans l’espace de son parent, ainsi que ses devoirs etc.



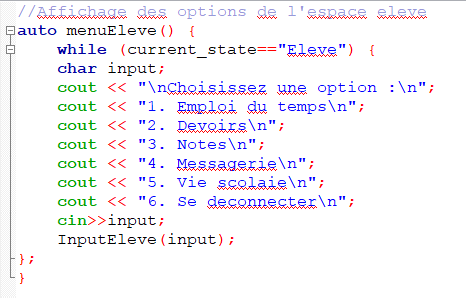
La fonction credentialsCheck verifie l’identifiant et le mot de passe saisi dans la page d’accueil. Elle verife si les données saisies existe en faisant appel a la methode login. Si les donnnées saisies correspondent, l’etat change en fonction de l’espace auquel on accede(Parent si on passe a l’espace parent, Eleve si on passe à l’espace élève etc). L’utilisateur est donc stocker dans une variable aussi (pour les operations a réaliser).



Maintenant on passe à la creation des espaces. Tout d’abord on crée une fonction pour vérifier l’option selectionnée sur la page d’accueil( espace parent, espace eleve etc). Pour cela on utilise un switch sur la valeur d’entré (input) avec un retour par defaut « Commande invalide ». Cette fonction sera fondamentale pour la suite.



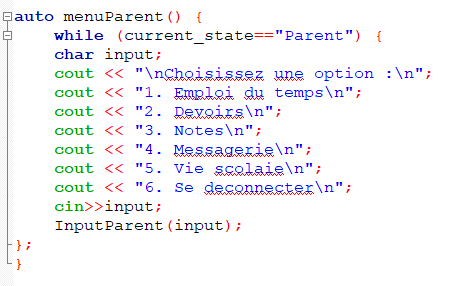
Même principe pour l’espace eleve. Pour chaque option, une action sera réalisé (en fonction de l’option) comme l’affichage de l’emploi du temps, des devoirs etc.



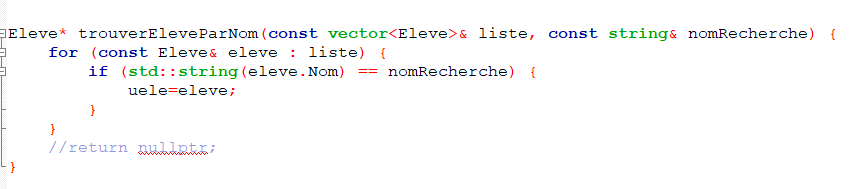
Cette fonction affiche les options disponibles dans l’espace élève. Lorsqu’une option est choisie, on appel la fonction précedente pour lire l’input et agir en fonction de ce dernier.



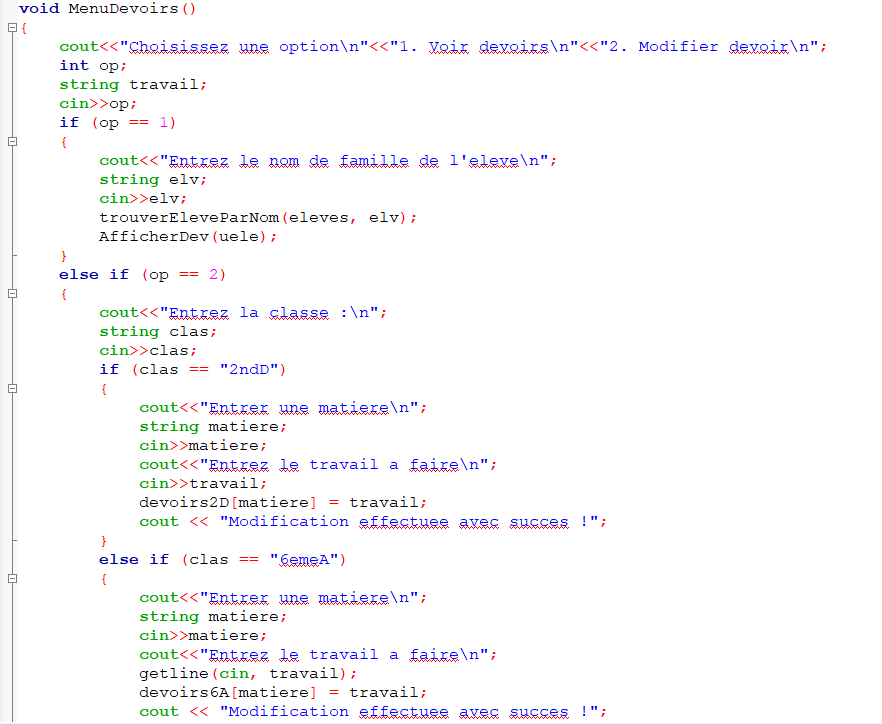
Cette fonction suit le meme principe que la fonction InputEleve, mais realise les options disponibles pour les parents.



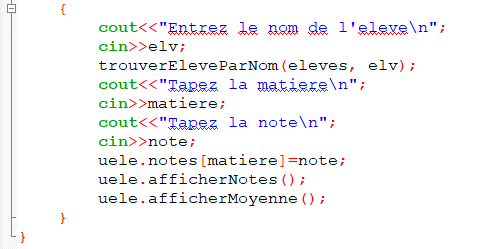
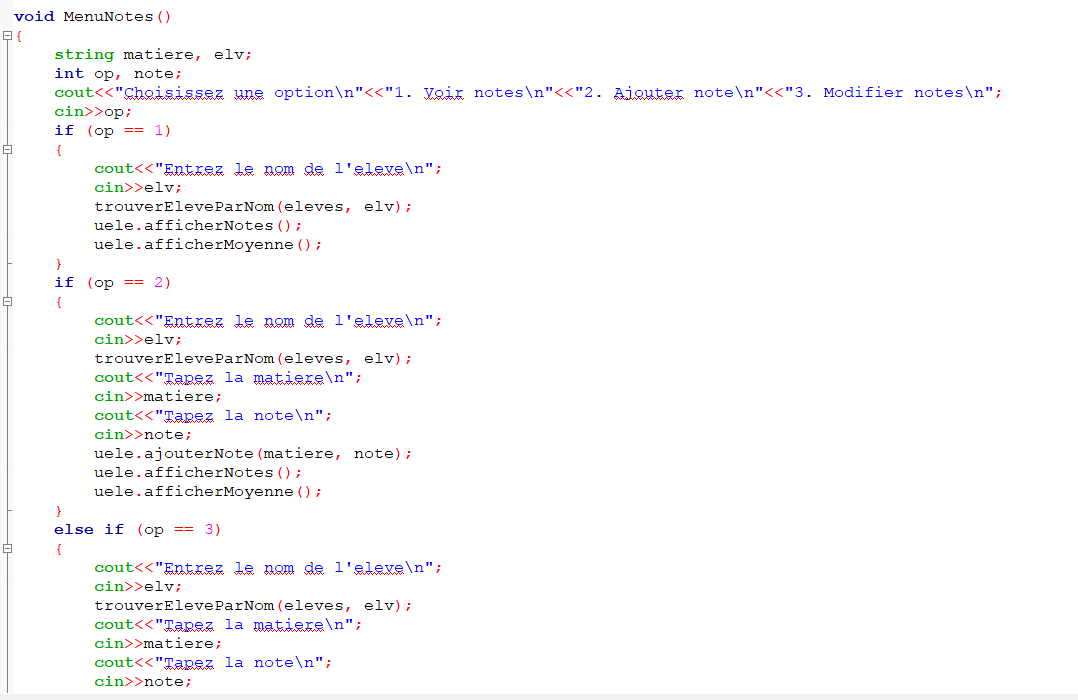
La fonction menuParent suit le même principe que la fonction menuEleve mais lis les inputs de la fonction InputParent.



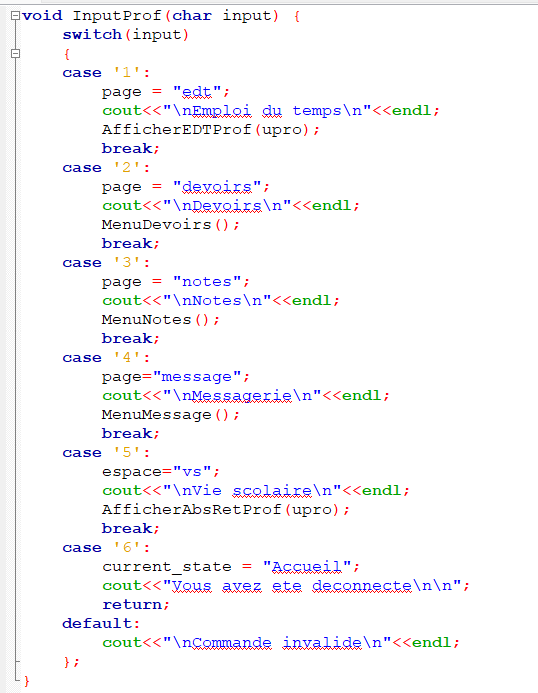
La fonction ci-dessus ressemble beaucoup à la fonction trouverDestParID (définie précedement) cependant, celle-ci trouve un éleve en fonction du nom de famille. Si le nom de l’éleve est reconnu, l’eleve est stocké dans une variable. Cette fonction nous sera utile plus loin.



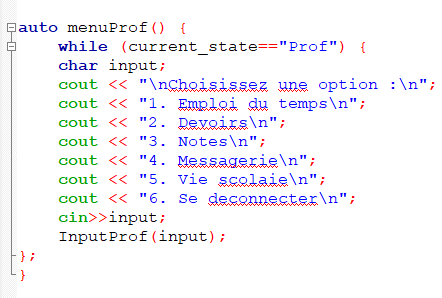
La fonction ci-dessus est principalement dédier aux professeurs et au cpe. Elle leur permet de modifier des devoirs, en ajouter ou bien juste les afficher. Le principe est simple : on saisie le nom de famille de l’élève puis à l’aide de la fonction trouverEleveParNom, on stock l’eleve concerné dans une variable. Si l’option 1 est sélectionné, on appel AfficherDev avec comme parametre la variable où l’élève a été stocké. Pour l’option 2, on tape la classe d’abord, ensuite la matiere et le travail a ajouter. Les données seront ajouter au dictionnaire de devoirs concernés.



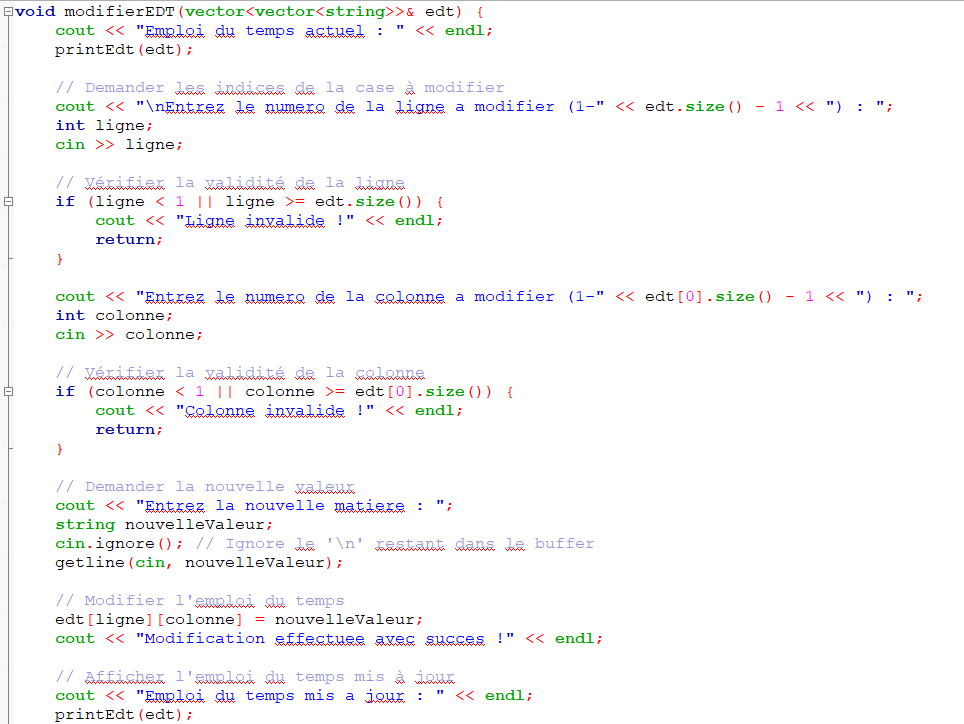
La fonction MenuNotes est exclusive aux professeurs et cpe. Elle affiche les différentes operations possibles sur les notes comme l’affichage simple des notes et de la moyenne des notes pour un élève (option 1) ou bien la modification/ l’ajout d’une note (option 2 et 3). Pour ajouter, afficher des notes ou bien pour afficher la moyenne, on appel la methode correspondante définit plus haut dans les classes Eleve et Prof. Pour la modification, on modifie juste la notes dont la clé est identique à la matière tapée en input.



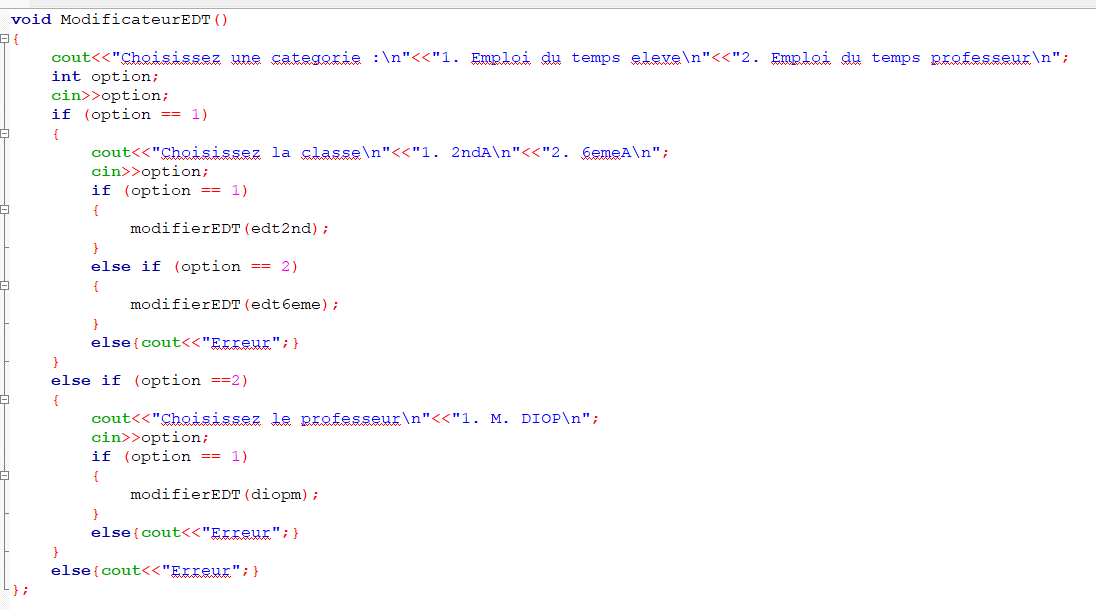
InputProf suit le même principe que InputEleve. Elle lis les input de l’utilisateur et réagis en fonction de l’option choisie



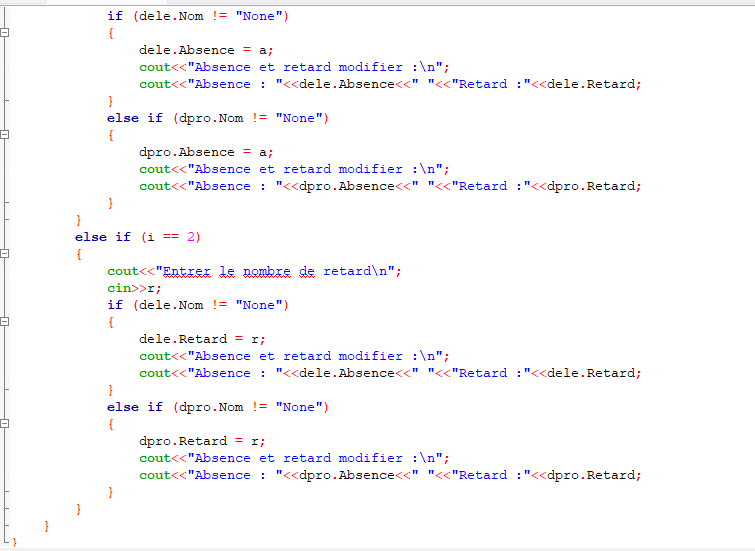
La fonction menuProf suit le même principe que menuEleve. Elle affiche les options disponibles pour les professeurs et appel InputProf pour lire l’input.



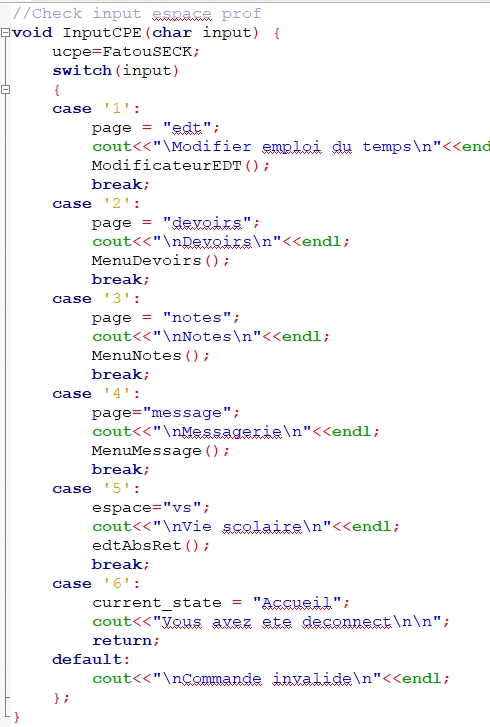
La fonction modifierEDT est indispensable pour la fonction qui suit. Tout d’abord, elle affiche l’emploi du temps actuel. Ensuite, elle prend en input la ligne et la colonne pour identifier l’élement a modifier (des checks sont inclus également afin de vérifier la validité des coordonnées données, et qu’elles n’excedent pas la taille du tableau). Ensuite, on tape la nouvelle valeur à insérer dans la cellule (le getline permet de taper un input avec des espaces). Ensuite, on modifie la valeur et on affiche l’emploi du temps modifié.



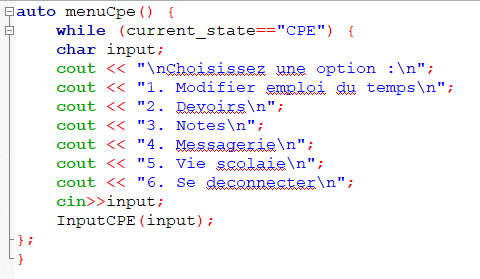
La fonction ModificateurEDT est l’une des seules fonctions exclusives a un seul utilisateur : le cpe. Elle permet de modifier l’emploi du temps des élèves ou du professeurs et ce en faisant appel a la fonction définie précedement(d’où son importance).



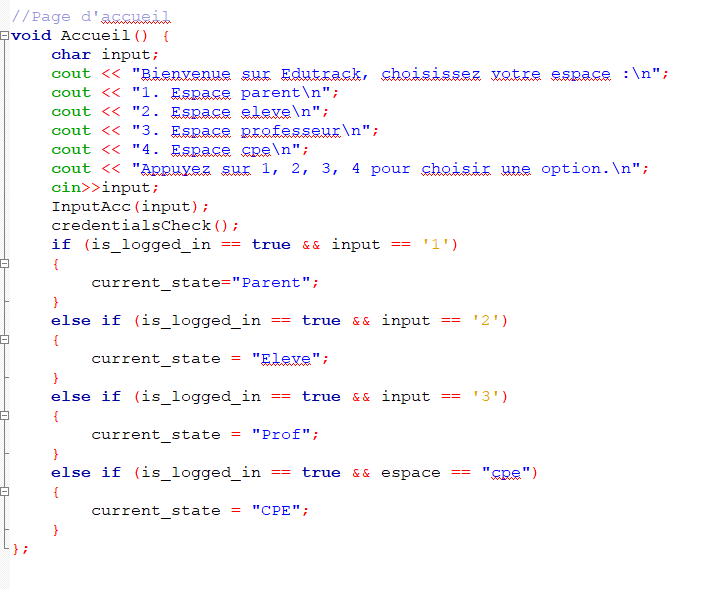
La fonction edtAbsRet (edit Absence Retard) est la deuxieme fonction exclusive à un utilisateur (cpe aussi). Elle permet de modifier les absences et retard des élèves et professeurs. Dans cette fonction, on commence par afficher les options (voir retards/absences ou modifier retard/absence. Cela se fait en trouvant l’utilisateur concerné par son ID (trouverDestParID) puis l’affichage des absences et retard en fonction de l’utilisateur (élève ou professeur). La deuxieme option modifie en accedant au valeurs par la clé du dictionnaires des absences et retard.



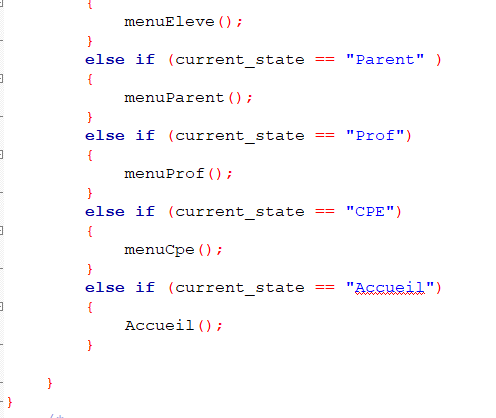
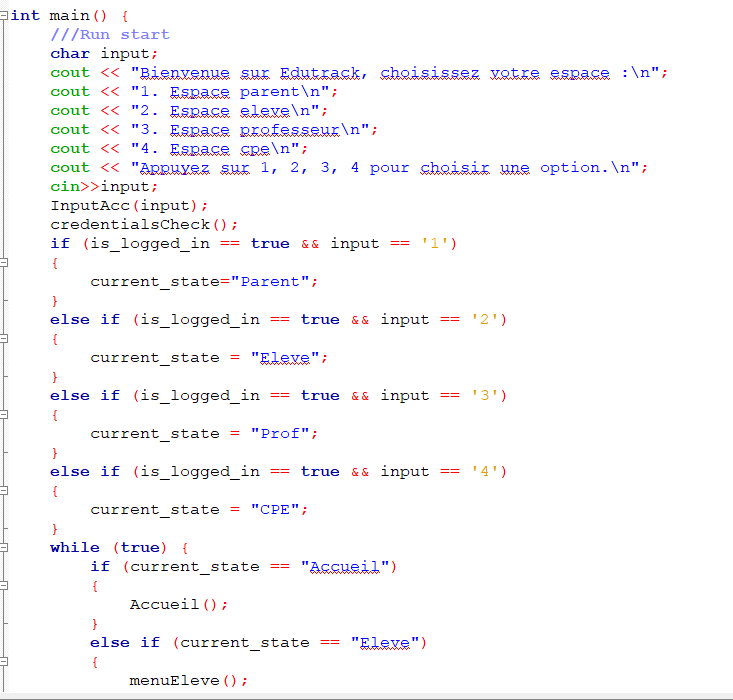
La fonction ci-dessus suit le même principe que les fonctions InputEleve ou InputParent.



menuCpe suit le même principe que menuEleve ou bien menuParent.



La fonction Accueil est la fonction qui centralise toutes les fonctions précedentes. Il s’agit de la page d’accueil avec les diverses options (espace parent, eleve etc). La fonction credentialsCheck est appelé ici pour vérifier les données entrées par l’utilisateur (id et mot de passe). Ensuite, si la personne est connectée avec succès, on change l’état (current\_state) vers l’espace en cours.



Pour terminer, la fonction main (« main loop ») qui sera excécutée après le debug du code. Elle est similaire à la fonction Accueil, mais rajoute le « gestionnaire des espaces ». De ce fait, elle permettra d’appeler les fonctions des menus (celle qui contiennent toutes les options des epsaces parents, élèves, professeurs et cpe) et donc d’utiliser l’ERP.