On présente ici les principaux points de l'implémentation et les choix techniques réalisés pour le projet de traduction de code d'un fichier C vers un fichier OCaml analogue. On a conçu ce projet de manière modulaire, pour nous faciliter la gestion et l'extension du code.

Nous avons choisi de structurer notre projet en utilisant des fichiers "manager" pour chaque étape clé de la traduction. Cette approche nous permet de créer une sorte de "chemin à suivre" où chaque nouveau lexème trouvé dans le code source C déclenche l'entrée dans le fichier de gestion correspondant. Cette modularité rend le fichier principal de traduction très court et facile à maintenir, car il ne fait que lancer les processus définis dans les différents gestionnaires.

Pour illustrer, à la première étape qui concerne les variables et leur définition, nous avons créé deux composants :

- Variable Manager : Lit et récupère le nom et la valeur des variables.
- Write Variable : Écrit la variable en OCaml, en utilisant des références appropriées.

Pour les commentaires, nous avons modifié le lexer de manière à :

- Détecter les commentaires sur une seule ligne (//) et les lire jusqu'à la fin de la ligne.
- Pour les commentaires multilignes (/* ... */), nous les détectons naïvement jusqu'à la borne de fin.
- Dans le lexer, tout le commentaire est réuni avec le même lexème 'C'

Les fonctions *printf* en C et en OCaml étant assez similaires, nous procédons à une lecture linéaire sans modifier la chaîne à afficher. Nous convertissons toutes les variables suivant le formatage OCaml et ignorons les parenthèses initiales et finales ainsi que les virgules.

Les boucles *while* en C et en OCaml sont très semblables. La principale difficulté réside dans la traduction de la condition. La méthode employée pour les boucles *while* est aussi appliquée pour les structures conditionnelles *if* et *else*. Nous parcourons la condition entre parenthèses, convertissons les opérateurs logiques de C en OCaml et les variables de manière assez identique que lors de l'étape des variables. Le reste du corps de la boucle est recopié sans modification.

La traduction des boucles *for* de C en OCaml est plus complexe en raison des différences de syntaxe. Nous avons donc choisi de ne pas les traduire directement, mais plutôt de les transformer une boucle *for* en boucle *while* en gardant la même.

condition d’arrêt, mais en déclarant la variable temporelle (souvent appelée i) avant la boucle while et en modifiant cette valeur à la fin de chaque implémentation de boucle.

Pour les fonctions, nous avons ajusté la syntaxe du début en ajoutant le mot-clé *let* et en retirant la déclaration *return*, laissant simplement la variable à renvoyer.

Certaines simplifications, comme la transformation des boucles *for*, peuvent ne pas couvrir tous les cas complexes et nécessitent une amélioration pour gérer les cas particuliers. La conversion des opérateurs logiques et arithmétiques est faite de manière basique. Il pourrait y avoir des cas non gérés correctement, nécessitant une approche plus approfondie.

En conclusion, notre approche modulaire facilite la traduction de C vers OCaml. Chaque gestionnaire de fichier traite un aspect spécifique de la syntaxe, rendant le processus global plus clair et organisé. Les choix techniques, bien que parfois simplifiés, permettent une première version fonctionnelle, avec des pistes d'amélioration identifiées pour une couverture plus exhaustive des cas complexes.