Rapport mini-projet
Interpréteur machine de Turing

Groupe:

- Lucas Colomines
- Benboudiaf Linda

Plan

1. Introduction
2. Algorithme et Diagramme des classes.
3. Conclusion

1.Introduction:

Le projet ci-joint est un interpréteur de machine de Turing. Comme spécifié dans le sujet, les transitions, les états et le vocabulaire de la machine sont à définir dans un fichier texte.

Lors de l'exécution du programme, il faut lui indiquer le fichier définissant la machine de Turing, le mot d'entrée ainsi qu'un, deux ou trois arguments permettant de choisir ce que le programme doit afficher comme output.

Le projet à été réalisé en c++11, il considère que l'entrée est suivi d'une infinité de symboles Blank (boucle infinie si l'exécution n'aboutis pas sur un état final de la machine). Le projet est globalement très commenté de manière à fournir une lecture et une compréhension facile du code source.

Les transitions sont stockées comme une liste d'adjacences, pour cette raison, l'exécution d'une machine contenant une très grand nombre de transitions peut être lente. Cependant, si un résultat existe pour une machine et un input, le programme renverra toujours un output.

Une gestion bas niveau des erreurs est fourni dans l'output.

2- Algorithme & Diagramme des classes:

L'idée est de lire ligne par ligne le fichier texte qui définit les états, les transitions ainsi que le vocabulaire puis lire mot à mot la ligne de donnée et l'insérer dans un dans un tableau.

Nous avons créé des structures de données pour stocker intelligemment le contenu du fichier texte. Nous travaillons donc avec des vecteurs de structures. (Voir commentaires Elements.h et Elements.cc pour plus de précisions).

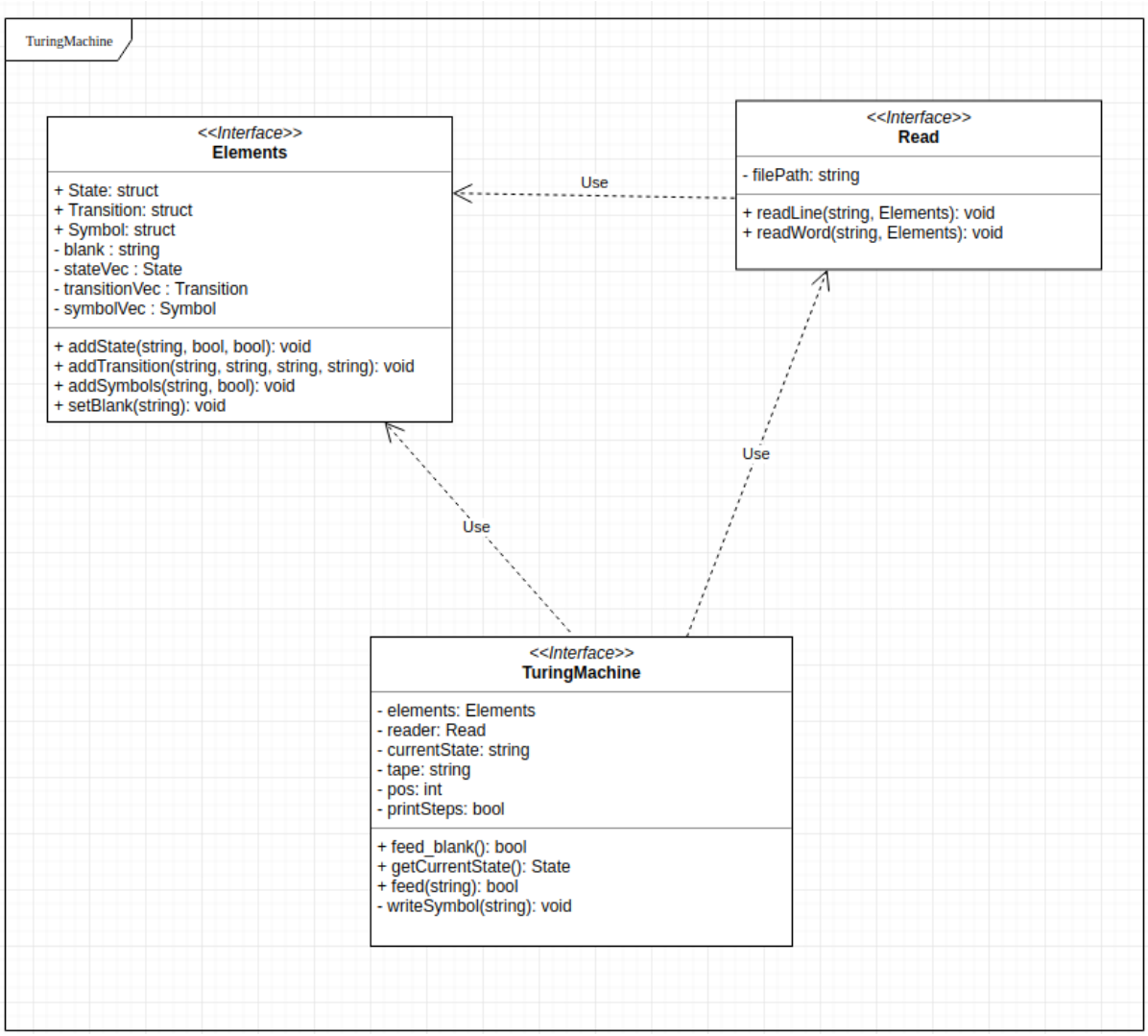
Concernant la lecture du fichier, la classe Read (voir les commentaires dans le header Read.h pour plus de précisions) est dédiée uniquement pour le processus de lecture de fichier, extraction des paramètres et d'insertion dans les tableaux.

La classe TuringMachine est la classe centrale qui assemble la classe Read et Elements, elle est définie par :

- **elements** : représente les transaction, le symbole, les états.
- **reader** : initialise **elements** lors de la lecture d'un fichier donnée.
- **currentState**: contient le nom de l'état courant de la machine. Est initialisé à l'état initial contenu dans le fichier texte.

- **tape** : définit le ruban qui va contenir le mot résultant de l'exécution de la machine sur le mot entrant.
- **pos** : représente la position du curseur sur le ruban (**tape**).

le diagramme des classes ci-dessous définit la relation entre les classes ainsi que leur composition:



3- Conclusion

Nous avons trouvé le projet très intéressant, nous avons eu quelque interrogations sur le choix de structures de données ainsi que l'optimisation de l'algorithme, nous avons fait le choix de ne pas trop nous concentrer sur l'optimisation de manière à proposer un projet suffisamment proche, dans sa structure, de la définition théorique des machines de Turing pour qu'il soit compréhensible par une personne ne maîtrisant que peu le langage de programmation. Le projet a été réalisé dans un cadre d'échange, on a utilisé l'outil Git (gitlab) afin d'organiser le développement du projet.