Compte rendu projet

Matteo, Marius, Alexis

Description de notre projet:

Afin de modéliser un automate, nous avons décidé d'utiliser une structure "automate", qui se compose d'un nombre d'états (entier), d'un état initial (entier), d'un état final (tableau de 0 et de 1), d'une matrice d'adjacence, qui correspond à un tableau de tableaux modélisant les liens entre les états de l'automate. Par exemple, si il existe une transition entre l'état 1 et l'état 2, alors la valeur dans la matrice sera 1 (0 sinon). Enfin une matrice de transition sera prise en compte dans la structure qui est elle aussi un tableau de tableaux. Cette fois, la valeur dans la matrice sera la lettre associée à la transition.

Avec cette structure, nous allons pouvoir manipuler les automates et répondre à l'objectif principal qui est de savoir si un mot entré par l'utilisateur se termine sur un état final ou non, et si le mot est accepté par l'automate. Pour cela nous avons fait 4 fonctions qui sont les suivantes:

nouvel_automate
ajoute_transition
lire_mot
libere_automate

Description des fonctions:

La première fonction sert à créer un nouvel automate. Elle est du type de la structure automate et renvoie un automate. La fonction prend en paramètre un nombre de sommets n et l'état initial "init", réserve la place mémoire nécessaire globale, puis associe les valeurs à chaque paramètre de la structure (y compris les réserves de mémoire). Ensuite, à l'aide d'une boucle imbriquée dans une seconde boucle, on initialise la matrice d'adjacence, et la matrice de transition comme des matrices nulles. Enfin, on crée le tableau des états finaux et on met toutes ses valeurs à 0, car en effet il n'y a pas encore d'états finaux.

La seconde fonction sert à remplir un automate, soit ajouter les liens entre les différents états. Elle prend en paramètre 2 états et la lettre associée à la transition. Elle vérifie d'abord que les états donnés appartiennent bien à l'automate. Ensuite, elle change la valeur dans la matrice d'adjacence a 1 et change la valeur de la transition en celle passée en paramètre (lettre).
Enfin, cette fonction ne renvoie rien.

La troisième fonction est le cœur du sujet, c'est-à- dire la lecture du mot donné par l'utilisateur. La fonction prend en paramètre un automate et un mot (char) et effectue les actions suivantes.

Tout d'abord, il faut définir une variable etat_actuel qui correspond au début à l'état initial. Cette variable permettra de naviguer dans l'automate.
Dans une première boucle allant de "i"=0 a la longueur du mot, nous allons mettre dans une variable lettre la lettre a l'indice "i" du mot et initialiser une variable etat_prohain a -1.
Ensuite dans une autre boucle, imbriquée dans la première, si il y a une transition possible entre état_actuel et un autre état "j" , et que la lettre correspond bien à celle de la matrice de transition, alors on peut changer la valeur de état_prochain a "j". Dans le cas contraire, on affiche une erreur, soit que le mot n'est pas accepté par l'automate.
Enfin on regarde si état_actuel est un état final et on affiche une phrase en fonction du résultat de cette condition.

Enfin, la dernière fonction est faite pour libérer l'espace mémoire lors de la création de l'automate. Il faut libérer chaque matrice avec une boucle, puis le tableau des états finaux.

Le main()

Bien évidemment, la fonction main doit aussi être décrite. Celui- ci est dans un fichier différent et voici ce qu'il fait.

Il crée un automate, ajoute quelques transitions et états finaux, demande à l'aide d'un scanf un mot à tester, lit le mot et conclut sur ce mot, puis libère l'espace mémoire pris par l'automate.

Répartition du travail dans le groupe.

Matteo: fonction lire_mot + fonction ajoute_transition
Alexis: création de la structure + fonction main
Marius: fonction creer_automate + fonction libere_automate

le Makefile, Readme, la librairie, ainsi que ce rapport ont été rédigés par tout le groupe, en coopération.