Projet d'algorithmes et programmation en C

Groupe 1

Thomas Lavaur - Fadwa Gardani
Jeremy Fournie - Jonathan Briand

M1 Cryptis 2019/20

Sommaire

1 Organisation général du projet

L'ensemble du projet est dans le dossier `Projet Turing`
Ce dossier est composé des fichiers `.c` et `.h` composant la machine de Turing et du `makefile` permettant la compilation du projet dans le terminal. Il y a également un sous-dossier contenant le rapport et un autre sous dossier contenant les fichiers `.txt` c'est à dire les tables de transitions des différents programmes de calcul.

2 Lancement de la machine de Turing

1. Ouvrir le terminal et acceder au dossier `Projet Turing`
2. Lancer `make` pour compiler le projet si ce n'est pas déjà fait.
   (`make` récupère le fichier `makefile` où sont écrites les différentes lignes de compilation.)
3. Lancer `./Main` dans le terminal pour démarrer la machine de Turing.
4. Suivre les instructions jusqu'à l'exécution finale du calcul.

3 Fonctionnement de la machine de Turing

1. La machine de Turing reçoit en entrée un programme représenté par sa table de transition stockée dans un fichier `.txt`.
2. La machine reçoit un mot et nous demande de confirmer la saisie, sinon on écrit un nouveau mot.
3. affichage de la bande.
4. Choisir une vitesse d'éxécution :
   — 0 pour instatannée.
   — 1 pour très rapide.
   — 2 pour moyen.
   — 3 pour lente.
5. La tête de lecture se place sur le premier caractère à gauche.
6. Le programme boucle tant que l'état est différent de `refuser` ou `accepter`.
7. La machine de Turing s'arrête en affichant le temps de calcul et le résultat.

4 Choix dans la construction de la machine de Turing

4.1 La machine

La machine de Turing est une structure :

```
struct turing{
    char* alphabet;
    transition* transitions;
    int cardinal_de_l_alphabet;
    int nombre_de_transition;
};
```

qui est construite grâce à la fonction `construction_programme()` qui permet de charger la table de transition choisie.
Ce chargement s'effectue en allant prendre la liste des programmes de calculs disponibles dans le dossier `programmes`. En particulier l'exploration des dossiers et fichiers est réalisée par l'inclusion au début du fichier `.c` et `.h` de `#include<dirent.h>` pour la gestion des dossiers et `#include<sys/types.h>` pour les ordinateurs n'utilisant pas un système d'exploitation Windows.

Puis cette même fonction `construction_programme()` initialise les différents paramètres de la structure `turing` nouvellement crée en lisant le fichier `.txt` de la table de transition choisie par l'utilisateur.

Enfin la fonction `destruction_programme(turing t)` libère l'espace mémoire grâce à la fonction `free()`.

4.2 Structure des tables de transitions

Chaque programme de calcul est un fichier `.txt` organisé comme un tableau dont les séparateurs sont représentés par des virgules ou des sauts de lignes avec :

1. En 1ère ligne, l'alphabet utilisé par la table de transition
2. Chaque ligne suivante est organisée telle que :

   `état_présent,caractère_lu,caractère_à_écrire,déplacement,état_suivant`

   où les déplacements appartiennent à l'alphabet {<,>,-} et les états sont des entiers avec en particulier, l'état d'acceptation est signifié par l'entier `-1` et l'état de refus par `-2`

4.3 La bande de lecture/écriture

Les données de travail sont inscrites sur une bande composée de cellules. Il s'agit d'une liste doublement chainée. Chaque cellule est une structure :

```
struct cellule{
    char caractere;
    struct cellule* suiv;
    struct cellule* preced;
};
```

On commence par initialiser la bande grâce à la fonction `initialisation_bande(turing t)` qui prend en argument la structure turing et retourne la bande complétée par le mot choisi. Cette fonction crée d'abord une bande vide (fonction `creation(bande b)`) puis demande à l'utilisateur de saisir une valeur. La bande reçoit les caractères en vérifiant à fur et à mesure qu'ils appartiennent à l'alphabet du programme choisi et les écrit de gauche à droite grâce aux fonctions `ecriture(bande b, char c)` et `deplacement_droite(bande b)`. On déplace ensuite la tête de lecture pour qu'elle pointe sur le premier caractère à gauche au moyen de la fonction `deplacement_gauche(bande b)`.

4.4 Calcul de la machine

Pour le temps de calcul, nous avons pris le calcul d'exécution du calcul sur la bande. Cette valeur varie en fonction de :

1. La table de transition séléctionnée.
2. La longeur du mot et sa complexité en fonction du calcul demandé.
3. La vitesse d'exécution renseignée par l'utilisateur.

Le temps de calcul est effectué par l'insertion judiscieuse du code source suivant :

```
#include <time.h>
    ...
    int temps_de_calcul = clock();
    ...
    printf("le temps de calcul sur la bande est de = %d ms", temps_de_calcul);
```

Le temps de calcul est affiché pour une meilleure lisibilité en millisecondes.

5 Exemple : Ajouter 1

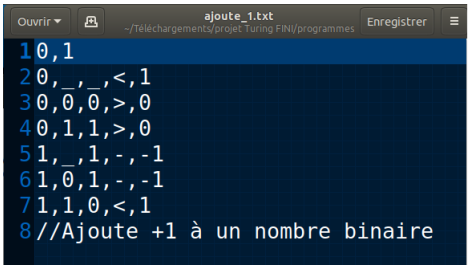
5.0.1 Table de transition de l'incrémentation

La table de transition permettant d'incrémenter de 1 un nombre binaire est donnée par le fichier ajoute_1.txt divisée en trois parties :
1) On rappelle que la première ligne correspond à l'alphabet du calcul.

Puis chaque ligne est un quintuplet composé de :

- état_présent,
- caractère_lu,
- caractère_à_écrire,
- déplacement,
- état_suivant

Enfin un commenaire explicite ce que calcul cette table.



5.0.2 Choix du programme d'incrémentation

```
camille@camille:~/Téléchargements/projet Turing FINI$ ./Main

Choix du programme à charger :
                              1 : multiplier_par_2.txt
                              2 : verifier_mot.txt
                              3 : longueur_binaire.txt
                              4 : divisibilite_par_4.txt
                              5 : ajoute_1.txt
                              6 : palindrome.txt
Entrer le numéro du programme : 5

Veuillez appuyez sur entré pour continuer...
```

Ainsi l'ouverture d'un terminal dans le dossier Projet Turing et l'exécution de ./Main permet le démarrage du programme où l'on choisira comme exemple le choix numéro 5 associé au fichier ajoute_1.txt.
La table de transition est alors chargée dans la structure turing.

5.0.3 Choix d'une valeur

```
Rappel des lettre de l'alphabet de ce programme :
 1 0

choix valeur pour le calcul : 1011
choix : 1011 Le mot 1011 est valide !
```

Par sécurité, on rappelle l'alphabet utilisé par le programme ici {0,1}
Puis la valeur à incrémenter est demandée puis vérifiée.

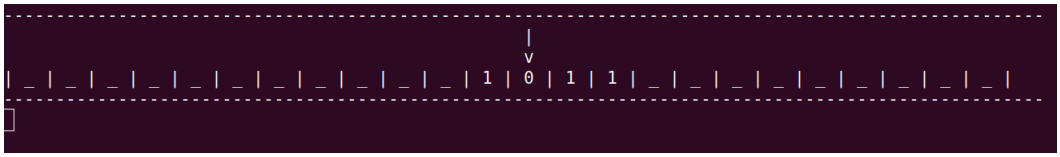
5.0.4 Affichage de la bande initialisée

Le mot entré est chargé sur la bande, et le programme demande vérification de la part de l'utilisateur avant exécution de l'incrémentation.
Une fois validé, la machine de Turing attend une vitesse d'affichage variant de 0 à 3 :

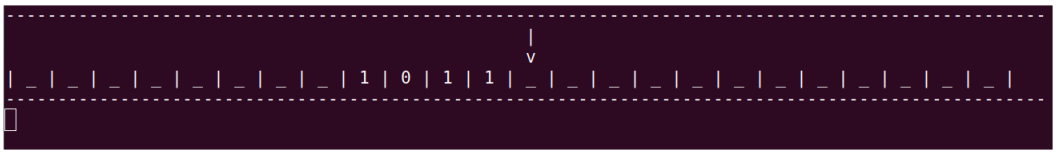
```
-------------------------------------------------------------------------------------------------------
                                                |
                                                v
| _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | 1 | 0 | 1 | 1 | _ | _ | _ | _ | _ | _ | _ | _ |
-------------------------------------------------------------------------------------------------------

Cette saisie vous convient-elle? [O/N]
o

Veuillez selectionner une vitesse d'éxécution :
        0 pour instantannée
        1 pour très rapide
        2 pour moyen
        3 pour lent
3
```

5.0.5 Calcul

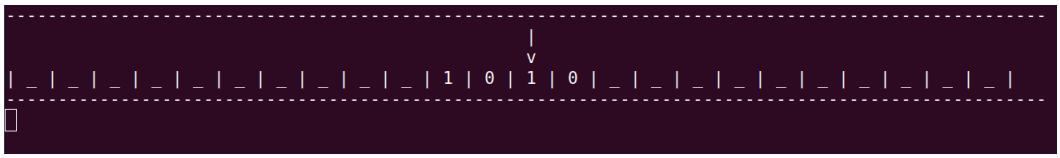
Le calcul lancé, on ne peut faire de pause dans le programme.
Dans le cas d'incrémentation l'état 0 déplace la tête de lecture sur la prochaine case vide.



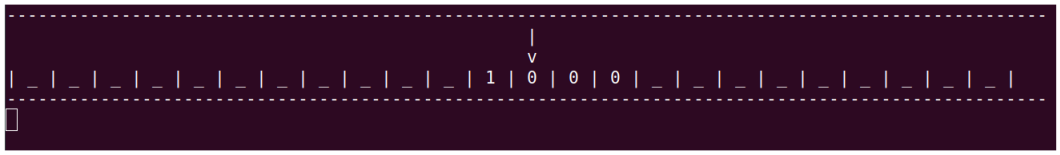
Une fois la première case vide atteinte, l'état 1 (de la table de transition) fait remonter la tête de lecture sur le dernier nombre.

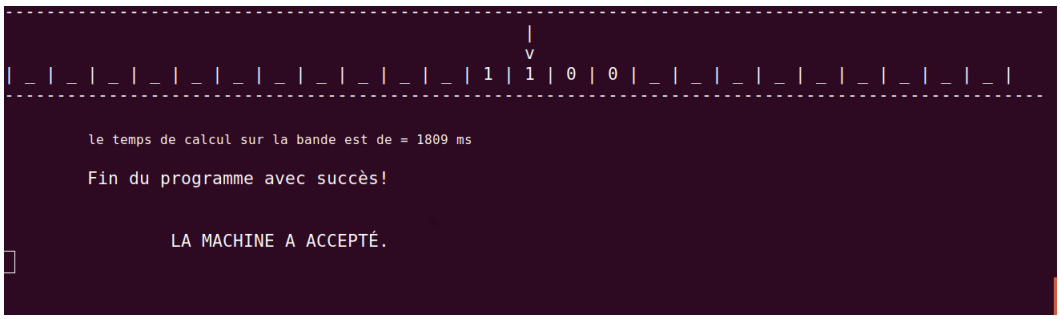


Tant qu'on lit 1 on met 0 et et remonte à gauche.



Dès qu'on lit 0, on remplace par 1 et le calcul est fini.

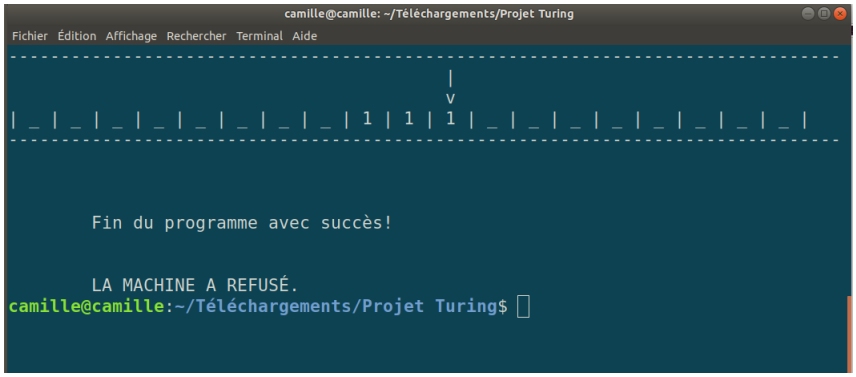




Le programme se termine, ici, en affichant le temps de calcul sur la bande et en validant la valeur.

Dans cet exemple divisibilite_par_4.txt permet de vérifier si un nombre binaire est divisible par 4.

Ici la machine de Turing refuse le nombre 111 qui n'est pas divisible par 4



6 Vers une machine de Turing à 3 bandes

Le passage de ce programme vers un système à 3 bandes n'est pas possible en l'état actuel du code source que nous avons implémenté.

Pour faire évoluer notre programme, il nous faudrait utiliser simultanément 3 structures turing et réécrire les tables de transitions pour prendre en compte les bandes supplémentaires.
De plus, il nous faudrait inclure la bibliothèque include¡pthread.h¿ qui permettrait de gerer au sein d'un même processus 3 threads pour une gestion simultanée des 3 bandes et assurant une fluidité dans l'execution du programme.

Faute de temps, nous n'avons pas poussé l'évolution de notre machine de Turing vers la gestion des 3 bandes.

7 Conclusion et organisation au sein du groupe

Ce projet de programmation en groupe nous a permis de découvrir la création d'un programme fonctionel dans le language C tout en respectant les délais imposés ainsi que la gestion du partage de tâches.
La rédaction de ce pdf en ligne a également contribué à notre perfectionnement dans le langage LaTeX.
Cette machine de Turing à d'abord été initiée par Thomas et nous avons pu participer au projet par diverses améliorations dans l'écriture du code comme dans l'impémentation de nouvelles fonctions et de nouveaux programmes de calcul.
En outre, l'inclusion des diverses bibliothèques et leurs gestions ont mis en exergue la force de ce langage de programmation et nous ouvre la voie vers de nouveaux paradigmes de programmation en C.