Corrigé TP8

[a] Le code de la porte est 1234

[b] L'arrêt de la simulation est commandé par l'entrée "-"

[c] Oui

[d] On stocke la seule sortie au milieu d'un immense tableau qui autorise le stockage d'autant de sorties qu'il y a de transitions. Voir en annexe le code la version 3 d'automate.c.
À chaque transition d'état, on affiche la sortie correspondante.

[e] On constate que l'implémentation n'autorise pas le changement facile du code de la porte. Pour cela, il faut modifier le tableau des transitions et le tableau des sorties.
On peut configurer un code comportant plusieurs fois le même chiffre.

[f] La version 1 est catastrophique car le code de gestion de l'automate et l'automate lui-même sont mélangés dans une seule fonction. Il est souvent bon, en programmation, d'isoler les concepts afin de pouvoir les modifier séparément les uns des autres. On n'a pas envie de modifier le code des transitions lorsqu'on souhaite simplement changer le message d'une sortie, par exemple.

La version 2 propose une amélioration basique consistant à isoler le calcul des transitions dans une fonction. On gagne déjà en lisibilité, mais le fonctionnement reste identique.

La version 3 apporte la vraie amélioration : la tabulation. L'automate est complètement défini par un tableau. Le code ne sert qu'à l'interpréter pour reproduire le comportement attendu. Cela va notamment faciliter la sauvegarde et le chargement de l'automate dans un fichier externe.

[g] Le format retenu pour le fichier contenant le code de la porte est de quatre chiffres séparés par une espace.

```
1 2 3 4
```

Une implémentation possible de la fonction de chargement de l'automate Version 4 à partir d'un tel fichier :

```c
int load_automate (char * filename) {
   FILE * file = NULL;
   char buffer[10];
   int a=0, b=0, c=0, d=0;

   if (NULL == (file = fopen (filename, "r"))) {
      return 1;
   }
   fgets (buffer, 10, file);
   fclose (file);

   sscanf (buffer, "%d %d %d %d", &a, &b, &c, &d);
   if (invalide (a) ||
       invalide (b) ||
       invalide (c) ||
       invalide (d)) {
      return 2;
   }

   // Init transitions with 0
   init_null ();
   init_sorties (d);

   transition[Q0][a] = 1;
   transition[Q1][a] = 1;
```

```
    transition[Q2][a] = 1;
    transition[Q3][a] = 1;
    transition[Q4][a] = 1;

    transition[Q1][b] = 2;
    transition[Q2][c] = 3;
    transition[Q3][d] = 4;

    return 0;
}
```

[h] Voir en annexe pour le code complet du programme supprimant les commentaires en C.

[i] Voir en annexe pour le code complet du programme supprimant les commentaires en Shell.

Annexe 1 - Automates

Implémentation possible de automate.c Version 1. Moralité : c'est super long et pas très clair, vu que tout est mélangé.

```c
#include <stdio.h>
#include <string.h>
#include "automate.h"

#define NB_ETATS 5
#define NB_ENTREES 10
#define LG_MAX_SORTIE 128

#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define Q4 4

typedef int etat; // Intérêt pédagogique ...

etat etat_initial() {
   return Q0;
}

int est_final(etat Q) {
   return Q == Q4;
}

int lire_entree() {
   char c;
   scanf("\n%c", &c);
   while (((c < '0') || (c > '9')) && (c != '-')) {
      printf("entree invalide ! un chiffre entre 0 et 9 (- pour finir) ?\n");
      scanf("\n%c", &c);
   }
   if (c == '-') {
      return -1;
   }
   else {
      return c-'0';
   }
}

void simule_automate() {
   int etat_courant=0, etat_suivant=0, entree=0;
   etat_courant = etat_initial();

   while (entree != -1) {
      entree = lire_entree();
      switch (etat_courant) {
      case Q0 :
         switch (entree) {
         case 1:
            etat_suivant = Q1;
            break;
         default:
            etat_suivant = Q0;
            break;
         }
         break;

      case Q1 :
         switch (entree) {
         case 1 :
            etat_suivant = Q1;
            break;
         case 2 :
            etat_suivant = Q2;
            break;
         default :
            etat_suivant = Q0;
            break;
         }
```

```
            break;

        case Q2 :
            switch (entree) {
            case 1 :
                etat_suivant = Q1;
                break;
            case 3 :
                etat_suivant = Q3;
                break;
            default :
                etat_suivant = Q0;
                break;
            }
            break;

        case Q3 :
            switch (entree) {
            case 1 :
                etat_suivant = Q1;
                break;
            case 4 :
                etat_suivant = Q4;
                break;
            default :
                etat_suivant = Q0;
                break;
            }
            break;

        case Q4 :
            switch (entree) {
            case 1 :
                etat_suivant = Q1;
                break;
            default :
                etat_suivant = Q0;
                break;
            }
            break;

        default : break;
        }
        printf("état courant : %d, état suivant : %d, entree : %d\n", etat_courant, etat_suivant
            , entree);

        if (est_final(etat_suivant)) {
            printf ("Vous pouvez entrer !\n");
        }
        etat_courant = etat_suivant;
    }
}
```

Implémentation possible de automate.c Version 2.

```
#include <stdio.h>
#include <string.h>
#include "automate.h"

#define NB_ETATS 5
#define NB_ENTREES 10
#define LG_MAX_SORTIE 128

#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define Q4 4

typedef int etat; // Intérêt pédagogique ...

etat etat_initial() {
```

```
    return Q0;
}

int est_final(etat Q) {
    return Q == Q4;
}

etat transition (etat Q, int entree) {
    switch (Q) {
    case Q0 :
        switch (entree) {
        case 1:
            return Q1;
        default:
            return Q0;
        }
        break;

    case Q1 :
        switch (entree) {
        case 1 :
            return Q1;
        case 2 :
            return Q2;
        default :
            return Q0;
        }
        break;

    case Q2 :
        switch (entree) {
        case 1 :
            return Q1;
        case 3 :
            return Q3;
        default :
            return Q0;
        }
        break;

    case Q3 :
        switch (entree) {
        case 1 :
            return Q1;
        case 4 :
            return Q4;
        default :
            return Q0;
        }
        break;

    case Q4 :
        switch (entree) {
        case 1 :
            return Q1;
        default :
            return Q0;
        }
        break;

    default :
        return Q0;
    }
}

int lire_entree() {
    char c;
    scanf("\n%c", &c);
    while (((c < '0') || (c > '9')) && (c != '-')) {
        printf("entree invalide ! un chiffre entre 0 et 9 (- pour finir) ?\n");
        scanf("\n%c", &c);
    }
    if (c == '-') {
        return -1;
```

```
    }
    else {
        return c-'0';
    }
}

void simule_automate() {
    int etat_courant=0, etat_suivant=0, entree=0;
    etat_courant = etat_initial();

    while (entree != -1) {
        entree = lire_entree();

        etat_suivant = transition (etat_courant, entree);
        printf("état courant : %d, état suivant : %d, entree : %d\n", etat_courant, etat_suivant
            , entree);

        if (est_final(etat_suivant)) {
            printf ("Vous pouvez entrer !\n");
        }
        etat_courant = etat_suivant;
    }
}
```

Implémentation possible de automate.c Version 3.

```
#include <stdio.h>
#include <string.h>
#include "automate.h"

#define NB_ETATS 5
#define NB_ENTREES 10
#define LG_MAX_SORTIE 128

#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define Q4 4

typedef int etat; // Intérêt pédagogique ...

int transition[NB_ETATS][NB_ENTREES] = {
    { 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 1, 2, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 1, 0, 3, 0, 0, 0, 0, 0, 0 },
    { 0, 1, 0, 0, 4, 0, 0, 0, 0, 0 },
    { 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
};

char sortie[NB_ETATS][NB_ENTREES][LG_MAX_SORTIE];

void init_sorties (void) {
    for (int i=0; i<NB_ETATS; i++) {
        for (int j=0; j<NB_ENTREES; j++) {
            sortie[i][j][0] = '\0';
        }
    }
    strcpy(sortie[Q3][4], "Code bon, vous pouvez entrer.");
}

etat etat_initial() {
    return Q0;
}

int est_final(etat Q) {
    return Q == Q4;
}

int lire_entree() {
    char c;
    scanf("\n%c", &c);
```

```
    while (((c < '0') || (c > '9')) && (c != '-')) {
        printf("entree invalide ! un chiffre entre 0 et 9 (- pour finir) ?\n");
        scanf("\n%c", &c);
    }
    if (c == '-') {
        return -1;
    }
    else {
        return c-'0';
    }
}

void simule_automate() {
    int etat_courant=0, etat_suivant=0, entree=0;
    etat_courant = etat_initial();
    init_sorties();

    while (entree != -1) {
        entree = lire_entree();

        etat_suivant = transition[etat_courant][entree];
        printf("%s\n", sortie[etat_courant][entree]);
        printf("état courant : %d, état suivant : %d, entree : %d\n", etat_courant, etat_suivant
            , entree);
        etat_courant = etat_suivant;
    }
}
```

Annexe 2 - Suppression commentaires C

Le code du programme est réparti en 3 fichiers : `automate.c`, `automate.h`, `supp_comm_C.c`. Voici le contenu de ces fichiers :

Listing 1 – supp_comm_sh.c

```c
/**
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>
#include "automate.h"

int main (int argc, char *argv[]) {
   FILE * source = NULL;
   if (argc != 2) {
      printf ("USAGE: ./supp_comm_C file.c\n");
      return 1;
   }

   char * filename = argv[1];
   source = fopen (filename, "r");
   if (source == NULL) {
      printf ("Impossible de lire %s\n", filename);
      return 1;
   }

   /* Appel de l'automate */
   traiter_fichier (source, stdout);

   fclose (source);
   return 0;
}
```

Listing 2 – automate.h

```c
/**
 * @autor <corentin@marciau.fr>
 */

#ifndef _AUTOMATE_H_
#define _AUTOMATE_H_

#define NB_ETAT 4
#define Q1 0
#define Q2 1
#define Q3 2
#define Q4 3
typedef int Etat;

#define NB_ENTREE 3
#define E_SLASH  0              /* caractère : /      */
#define E_ETOILE 1              /* caractère : *      */
#define E_AUTRE  2              /* caractère : autre */
typedef int Entree;

#define S_IDENTITE        0     /* sortie : car       */
#define S_SLASH           1     /* sortie : /         */
#define S_SLASH_IDENTITE  2     /* sortie : / + car   */
#define S_NEANT           3     /* sortie : (néant)   */
typedef int Sortie;

void traiter_fichier (FILE *, FILE *);

#endif
```

Listing 3 – automate.c

```c
/**
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>
#include "automate.h"

int transitions[NB_ETAT][NB_ENTREE] = {
   { Q2, Q1, Q1 },
   { Q2, Q3, Q1 },
   { Q3, Q4, Q3 },
   { Q1, Q4, Q3 }
};

int sorties[NB_ETAT][NB_ENTREE] = {
   { S_NEANT, S_IDENTITE, S_IDENTITE       },
   { S_SLASH, S_NEANT,    S_SLASH_IDENTITE },
   { S_NEANT, S_NEANT,    S_NEANT          },
   { S_NEANT, S_NEANT,    S_NEANT          }
};

void executer_sortie (Sortie sortie, FILE * destination, char caractere) {
   switch (sortie) {
   case S_IDENTITE:
      fprintf (destination, "%c", caractere);
      return;
   case S_SLASH:
      fprintf (destination, "%c", '/');
      return;
   case S_SLASH_IDENTITE:
      fprintf (destination, "/%c", caractere);
      return;
   case S_NEANT:
      return;
   }
}

int lire_entree (FILE * source, Entree * entree, char * caractere_ptr) {
   char caractere = fgetc(source);
   if (feof (source)) {
      return 0;
   }

   switch (caractere) {
   case '/':
      *entree = E_SLASH;
      break;
   case '*':
      *entree = E_ETOILE;
      break;
   default:
      *entree = E_AUTRE;
   }

   *caractere_ptr = caractere;
   return 1;
}

void traiter_fichier (FILE * source, FILE * destination) {
   char   caractere    = 0;
   Etat   etat_courant = Q1, etat_suivant = Q1;
   Entree entree       = E_AUTRE;
   Sortie sortie       = S_NEANT;

   while (lire_entree (source, &entree, &caractere)) {
      etat_suivant = transitions[etat_courant][entree];
      sortie       = sorties[etat_courant][entree];

      executer_sortie (sortie, destination, caractere);
      etat_courant = etat_suivant;
   }
}
```

Annexe 3 - Suppression commentaires Shell

On reprend largement le code de l'automate précédent pour supprimer les commentaires en Shell.

Listing 4 – supp_comm_sh.c

```c
/**
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>
#include "automate.h"

int main (int argc, char *argv[]) {
   FILE * source = NULL;
   if (argc != 2) {
      printf ("USAGE: ./supp_comm_sh file.c\n");
      return 1;
   }

   char * filename = argv[1];
   source = fopen (filename, "r");
   if (source == NULL) {
      printf ("Impossible de lire %s\n", filename);
      return 1;
   }

   /* Appel de l'automate */
   traiter_fichier (source, stdout);

   fclose (source);
   return 0;
}
```

Listing 5 – automate.h

```c
/**
 * @autor <corentin@marciau.fr>
 */

#ifndef _AUTOMATE_H_
#define _AUTOMATE_H_

#define NB_ETAT 4
#define Q1 0
#define Q2 1
#define Q3 2
#define Q4 3
typedef int Etat;

#define NB_ENTREE 5
#define E_DIESE     0           /* caractère : #      */
#define E_DOLLAR    1           /* caractère : $      */
#define E_ANTISLASH 2           /* caractère : \      */
#define E_RETOUR    3           /* caractère : \n     */
#define E_AUTRE     4           /* caractère : autre  */
typedef int Entree;

#define S_IDENTITE       0      /* sortie : car       */
#define S_NEANT          1      /* sortie : (néant)   */
typedef int Sortie;

void traiter_fichier (FILE *, FILE *);

#endif
```

Listing 6 – automate.c

```c
/**
 * @autor <corentin@marciau.fr>
 */

#include <stdio.h>
#include "automate.h"

int transitions[NB_ETAT][NB_ENTREE] = {
   { Q4, Q2, Q3, Q1, Q1 },
   { Q1, Q1, Q1, Q1, Q1 },
   { Q1, Q1, Q1, Q1, Q1 },
   { Q4, Q4, Q4, Q1, Q4 }
};

int sorties[NB_ETAT][NB_ENTREE] = {
   { S_NEANT,    S_IDENTITE, S_IDENTITE, S_IDENTITE, S_IDENTITE },
   { S_IDENTITE, S_IDENTITE, S_IDENTITE, S_IDENTITE, S_IDENTITE },
   { S_IDENTITE, S_IDENTITE, S_IDENTITE, S_IDENTITE, S_IDENTITE },
   { S_NEANT,    S_NEANT,    S_NEANT,    S_IDENTITE, S_NEANT    }
};

void executer_sortie (Sortie sortie, FILE * destination, char caractere) {
   switch (sortie) {
   case S_IDENTITE:
      fprintf (destination, "%c", caractere);
      return;
   case S_NEANT:
      return;
   }
}

int lire_entree (FILE * source, Entree * entree, char * caractere_ptr) {
   char caractere = fgetc(source);
   if (feof (source)) {
      return 0;
   }

   switch (caractere) {
   case '#':
      *entree = E_DIESE;
      break;
   case '$':
      *entree = E_DOLLAR;
      break;
   case '\\':
      *entree = E_ANTISLASH;
      break;
   case '\n':
      *entree = E_RETOUR;
      break;
   default:
      *entree = E_AUTRE;
   }

   *caractere_ptr = caractere;
   return 1;
}

void traiter_fichier (FILE * source, FILE * destination) {
   char   caractere    = 0;
   Etat   etat_courant = Q1, etat_suivant = Q1;
   Entree entree       = E_AUTRE;
   Sortie sortie       = S_NEANT;

   while (lire_entree (source, &entree, &caractere)) {
      etat_suivant = transitions[etat_courant][entree];
      sortie       = sorties[etat_courant][entree];

      executer_sortie (sortie, destination, caractere);
      etat_courant = etat_suivant;
   }
}
```