

# Programmation d'automates à base de conditions

Systeme et environnement de programmation

Université Grenoble Alpes

# Plan

- 1 Exemple
- 2 Algorithme général
- 3 Codage en langage C

# Plan

- 1 Exemple
- 2 Algorithme général
- 3 Codage en langage C

# Un exemple de transducteur : suppression des lignes vides

On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate :

# Un exemple de transducteur : suppression des lignes vides

On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate :

# Un exemple de transducteur : suppression des lignes vides

On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

**Les sorties seront indiquées entre [ ] pour éviter les confusions.**

(*c* désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :



# Un exemple de transducteur : suppression des lignes vides

On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :



# Un exemple de transducteur : suppression des lignes vides

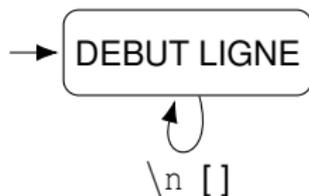
On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :



# Un exemple de transducteur : suppression des lignes vides

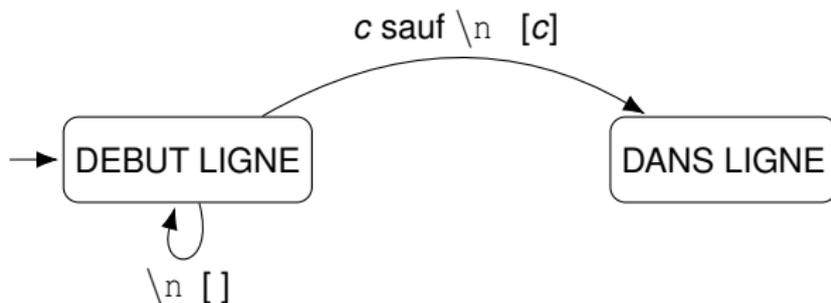
On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :



# Un exemple de transducteur : suppression des lignes vides

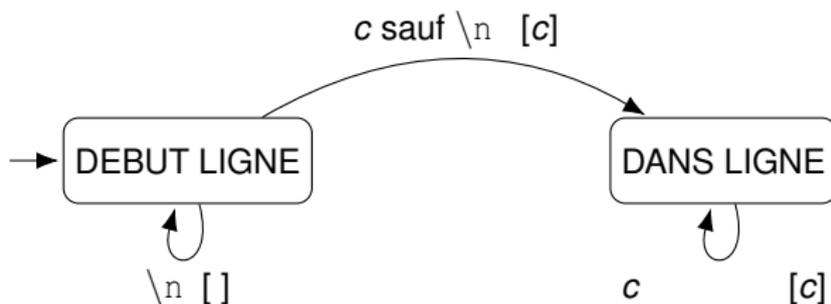
On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :



# Un exemple de transducteur : suppression des lignes vides

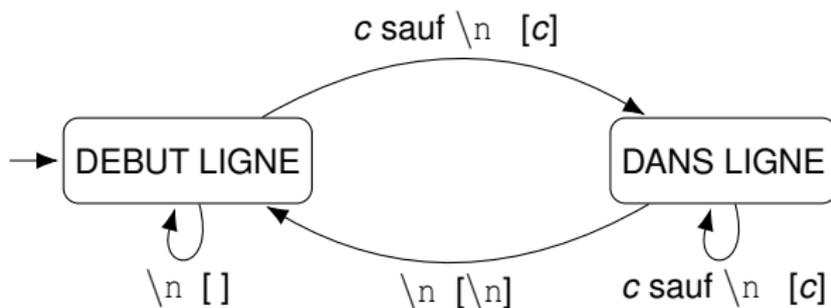
On traduit par un automate un traitement appliqué à un fichier texte :

- Entrées de l'automate : les caractères ASCII (lus dans le fichier)
- Sorties de l'automate : les caractères écrits dans le fichier résultat

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

Suppression des lignes vides :

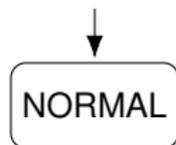


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)



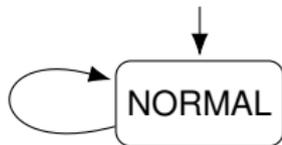
# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre `[ ]` pour éviter les confusions.

(`c` désignera un caractère quelconque, éventuellement avec des conditions)

`c`  
`[c]`

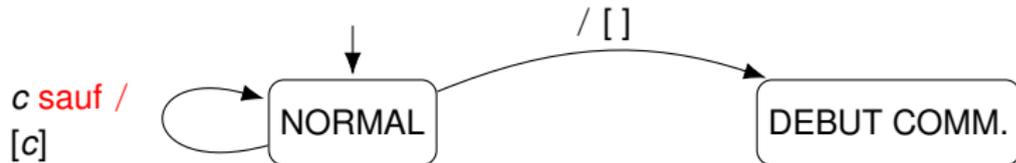


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

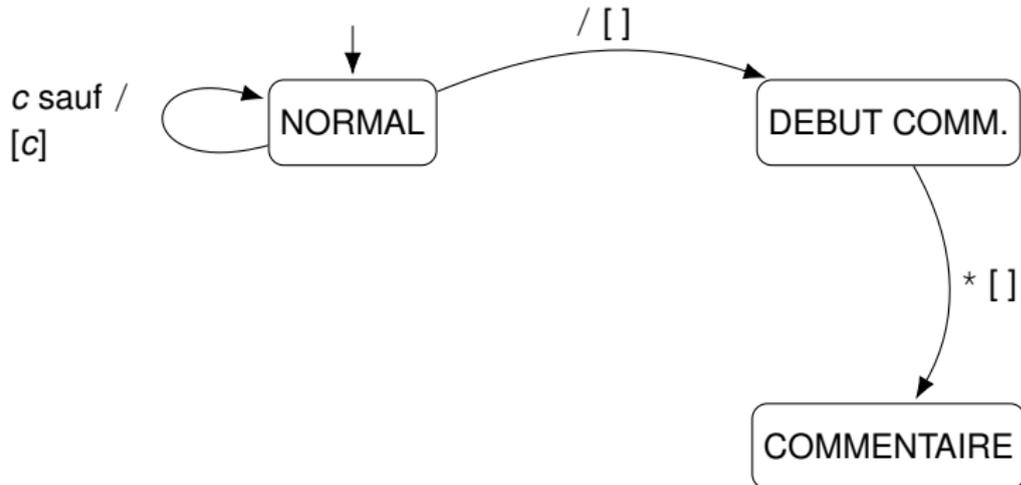


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

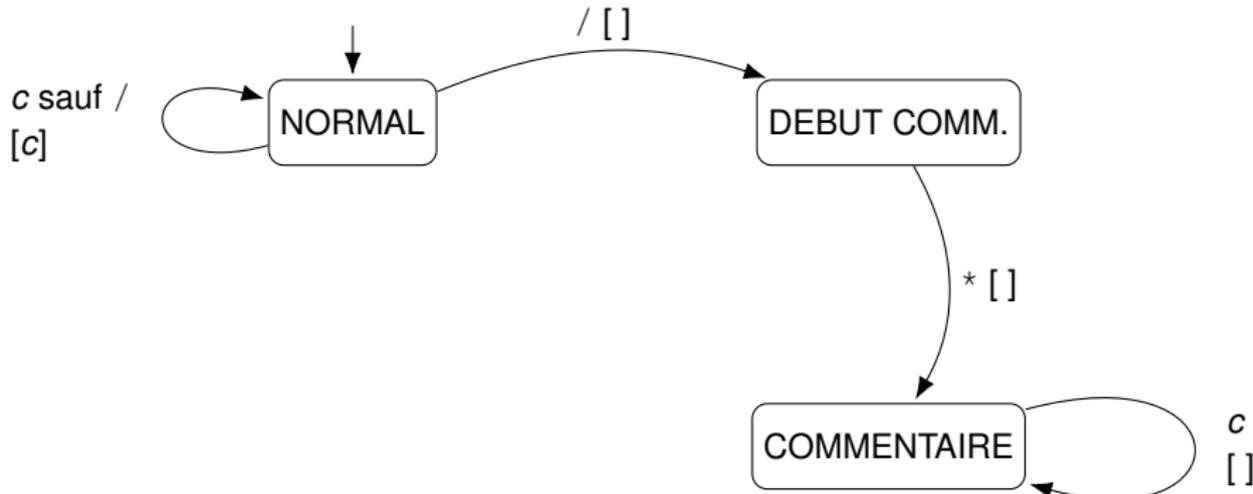


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

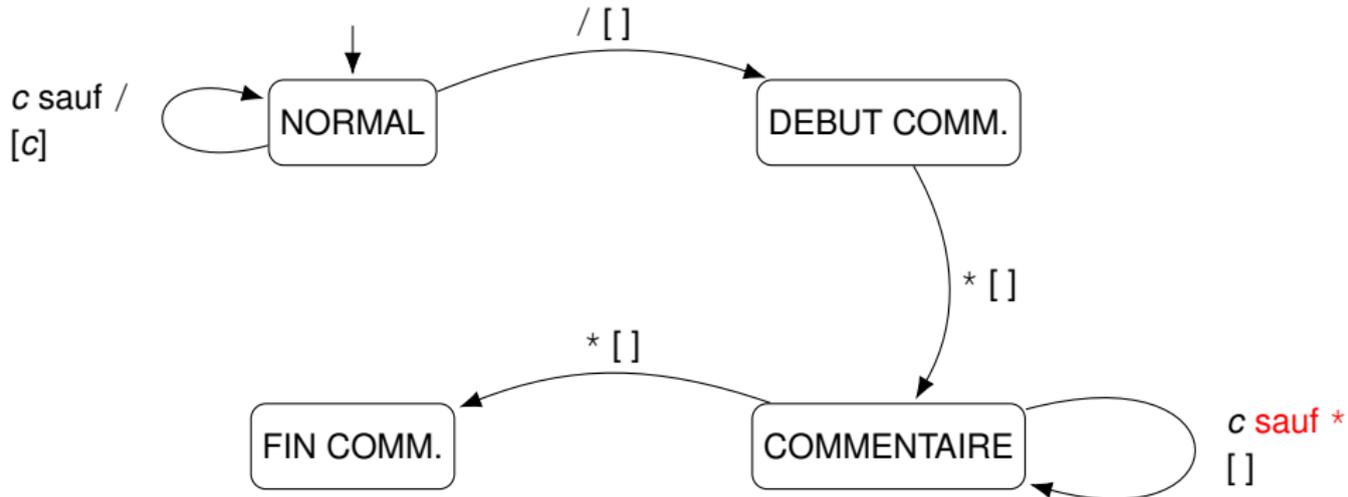


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre `[ ]` pour éviter les confusions.

(`c` désignera un caractère quelconque, éventuellement avec des conditions)

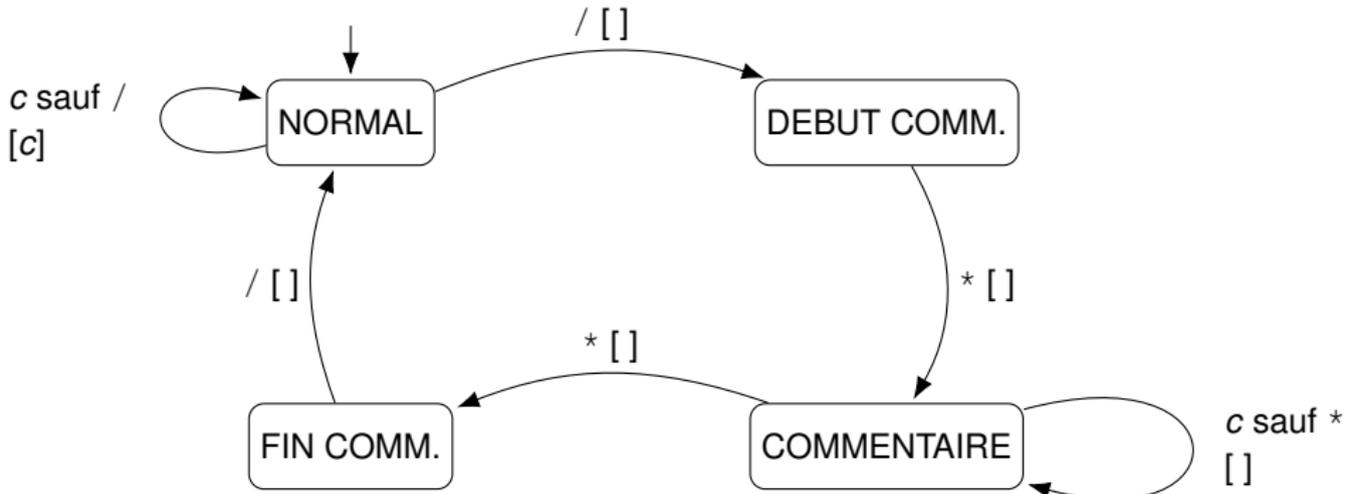


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre `[ ]` pour éviter les confusions.

(`c` désignera un caractère quelconque, éventuellement avec des conditions)

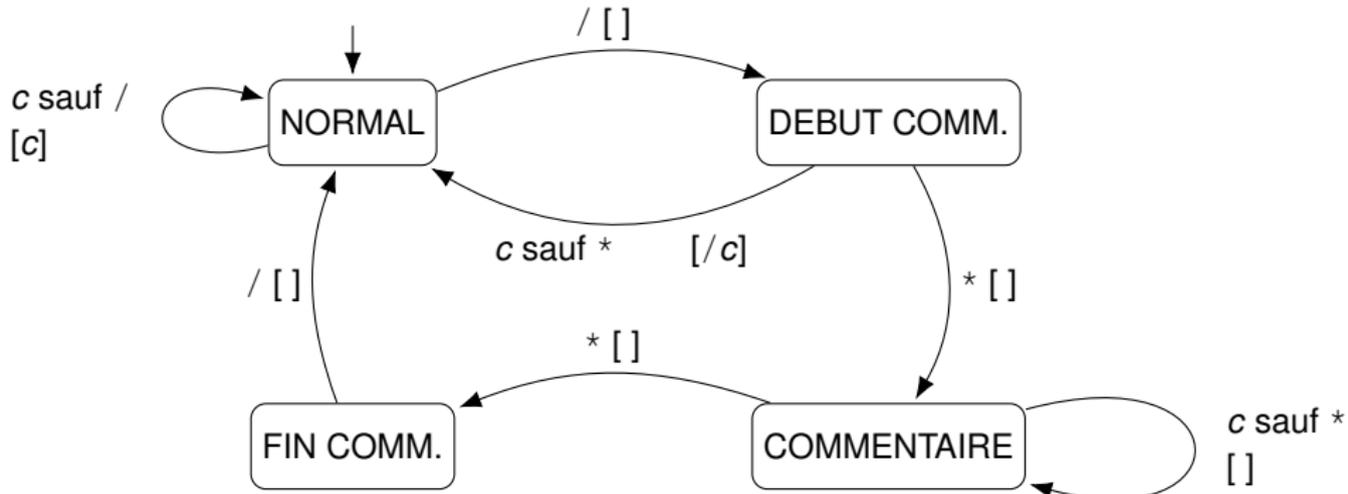


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** /\* ... \*/ en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

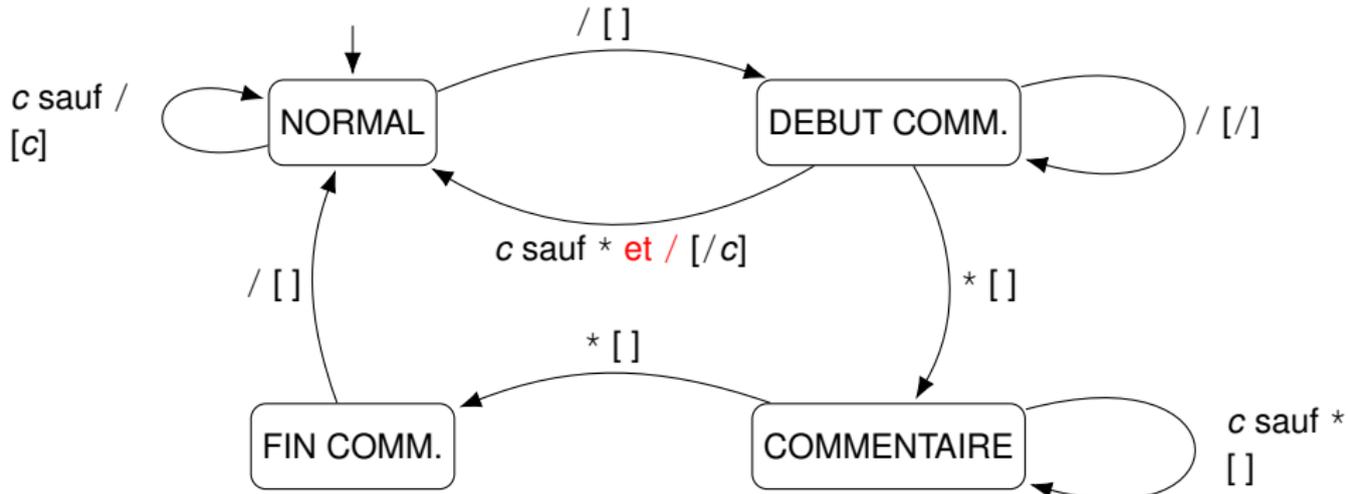


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** /\* ... \*/ en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

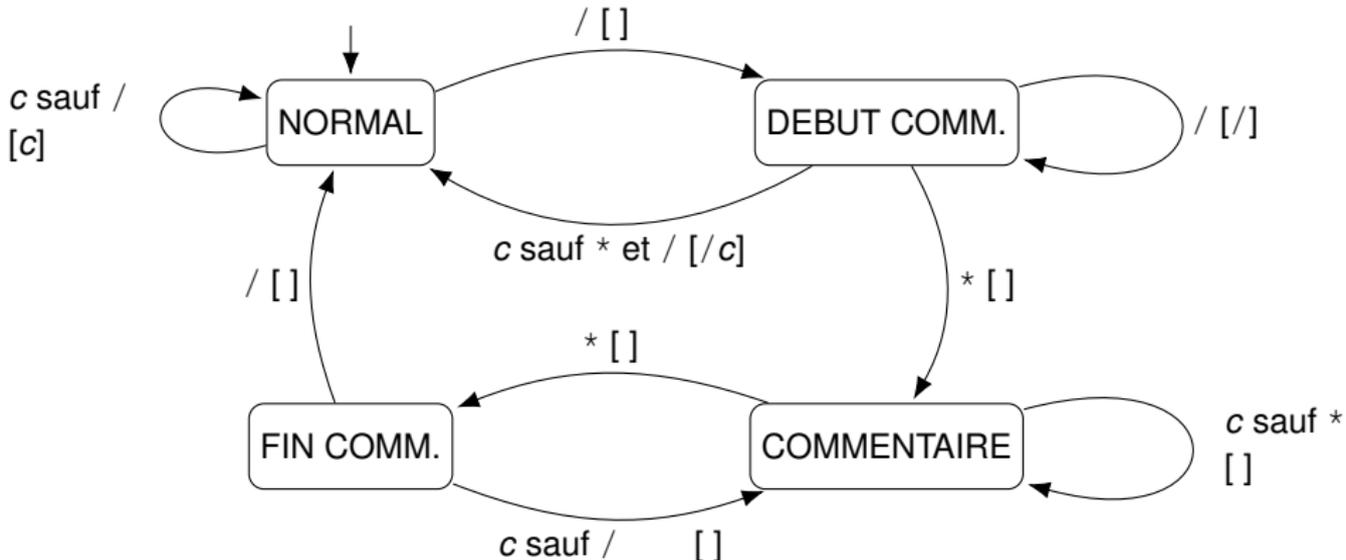


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** /\* ... \*/ en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)

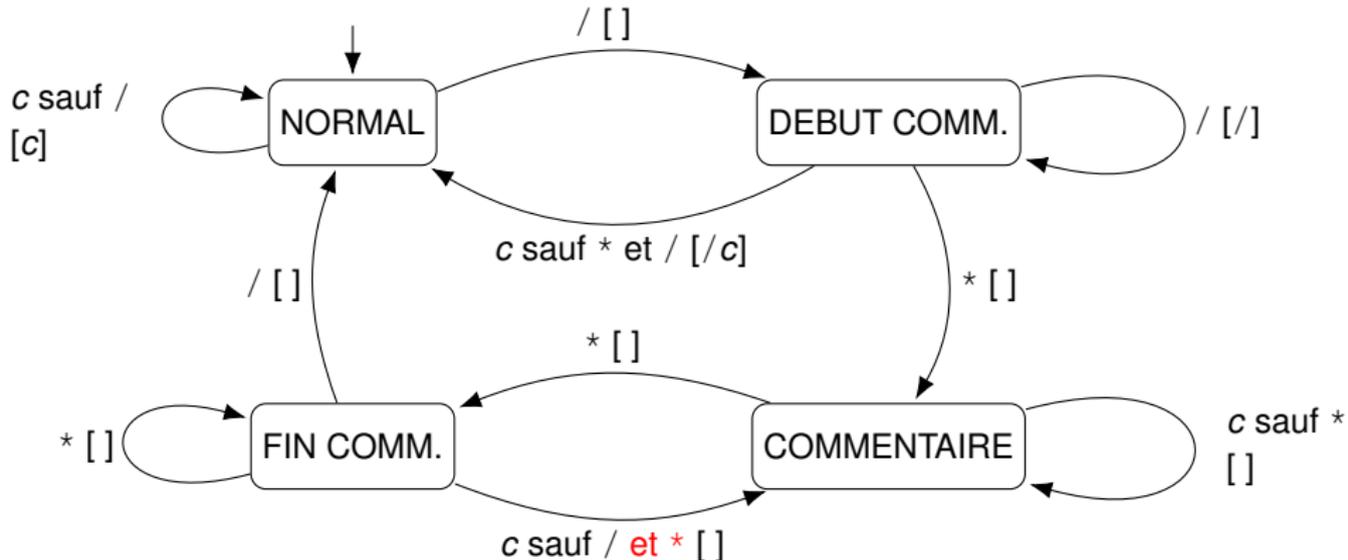


# Suppression des commentaires en C

Dessiner l'automate qui **supprime des commentaires** `/* ... */` en C.

Les sorties seront indiquées entre [ ] pour éviter les confusions.

(c désignera un caractère quelconque, éventuellement avec des conditions)



# Plan

- 1 Exemple
- 2 Algorithme général**
- 3 Codage en langage C

## Rappel : algorithme

```
etat_courant = Init ;

while (! FINI ) {
    entree = lire_entree() ;
    sortie = sortie(etat_courant, entree) ;
    etat_suivant = transition(etat_courant, entree) ;
    traiter_sortie(sortie) ;
    etat_courant = etat_suivant ;
    mise a jour de FINI
}
```

# Plan

- 1 Exemple
- 2 Algorithme général
- 3 Codage en langage C**

# États

```
#include <stdio.h>
#include <stdlib.h>

#define NORM 1
#define DEB_COM 2
#define COM 3
#define FIN_COM 4
```

# Fonction de sortie (1/2)

```
void f_sortie (int etat, char symb, char sortie[])
{
  switch (etat)
  {
    case NORM :
      switch (symb)
      {
        case '/' :          /* pas de sortie */
          sortie[0]='\0' ;
          break ;
        default :          /* texte "normal", sortie=symb */
          sortie[0]=symb ;
          sortie[1]='\0' ;
          break ;
      }
      break ;

    /* suite : slide suivant */
  }
}
```

## Fonction de sortie (2/2)

```
case DEB_COM :
    switch (symb)
    {
        case '**' :          /* pas de sortie */
            sortie[0]='\0' ;
            break ;
        case '/' :          /* sortie= '/' */
            sortie[0]='/' ;
            sortie[1]='\0' ;
            break ;
        default :          /* sortie= '/', symb */
            sortie[0]='/' ;
            sortie[1]=symb ;
            sortie[2]='\0' ;
            break ;
    }
    break ;

case COM :
case FIN_COM :          /* pas de sortie */
    sortie[0]='\0' ;
    break ;

}
}
```

# Fonction de transition (1/4)

```
int f_transition (int etat, char symb)
```

# Fonction de transition (1/4)

```
int f_transition (int etat, char symb)

{
int etat_suiv ;
  switch (etat)
  {
    case NORM :
      switch (symb)
      {
        case '/' :      /* debut de commentaire ??? */
          etat_suiv = DEB_COM ;
          break ;

          default :     /* texte "normal" */
            etat_suiv = NORM ;
            break ;
      }
      break ;

      /* suite : slide suivant */
```

## Fonction de transition (2/4)

```
case DEB_COM :
  switch (symb)
  {
    case '*' :      /* debut de commentaire */
      etat_suiv = COM ;
      break ;
    case '/' :      /* debut de commentaire ??? */
      etat_suiv = DEB_COM ;
      break ;
    default :      /* retour au texte "normal" */
      etat_suiv = NORM ;
      break ;
  }
  break ;

/* suite : slide suivant */
```

## Fonction de transition (3/4)

```
case COM :
  switch (symb)
  {
    case '*' :      /* fin de commentaire ??? */
      etat_suiv = FIN_COM ;
      break ;
    default :      /* texte commente */
      etat_suiv = COM ;
      break ;
  }
  break ;

/* suite : slide suivant */
```

## Fonction de transition (4/4)

```
case FIN_COM :
  switch (symb)
  {
    case '/' :      /* fin de commentaire */
      etat_suiv = NORM ;
      break ;
    case '*' :      /* fin de commentaire ??? */
      etat_suiv = FIN_COM ;
      break ;
    default :      /* retour au texte commente */
      etat_suiv = COM ;
      break ;
  }
  break ;
}
return etat_suiv ;
}
```

# Programme principal (1/2)

```
int main (int argc, char *argv[])
{
FILE *fichier_entree, *fichier_sortie ;
char cc ;
char sortie[3] ; /* deux caracteres au maximum, plus le '\0' */
int etat_courant, etat_suivant ;

if (argc != 3) {
    printf(" il faut 2 arguments !\n") ;
    return 1 ;
} ;

fichier_entree = fopen(argv[1], "r") ;
if (fichier_entree == NULL) {
    /* si le fichier n'a pas pu etre ouvert */
    printf("Le fichier %s n'existe pas\n", argv[1]) ;
    return 2 ;
} ;

fichier_sortie = fopen(argv[2], "w") ;

/* suite : slide suivant */
```

## Programme principal (2/2)

```
etat_courant = NORM ;

fscanf(fichier_entree, "%c", &cc) ; /* cc est le 1er caractere du texte */

while (!feof(fichier_entree)) {

    f_sortie(etat_courant, cc, sortie) ;

    etat_suivant = f_transition (etat_courant, cc) ;

    fprintf(fichier_sortie, "%s", sortie) ;

    etat_courant = etat_suivant ;

    fscanf(fichier_entree, "%c", &cc) ;
}

fclose(fichier_entree) ;
fclose(fichier_sortie) ;

return 0 ;
}
```