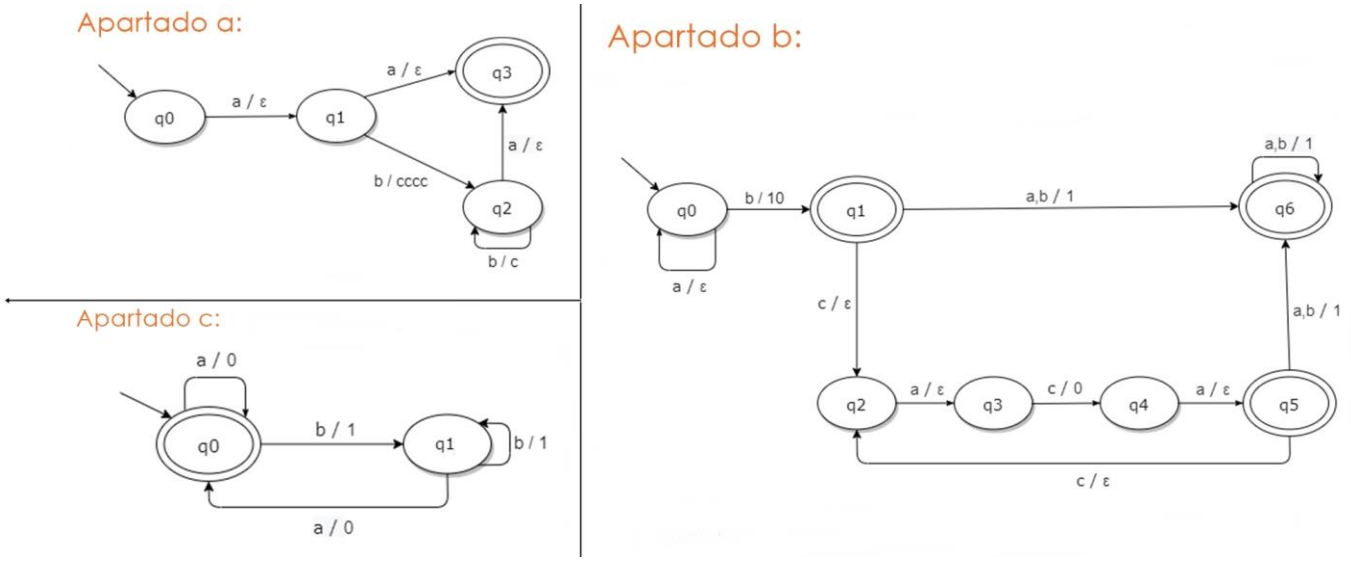
Para llevar a cabo el proyecto en primer lugar he partido de tres autómatas divididos por apartados del enunciado, uno para L, otro para T(M) y un tercero resultado de relacionar los anteriores, el autómata de traducción. Este último se expone a continuación con su apartado:



El código en cuestión de la practica está compuesto por un *enum* para definir los booleanos, varios *struct*, uno para almacenar las traducciones y tamaño de cada estado de cada autómata leído y otro en el que tiene la información necesaria que debe tener cada estado de un AFDT. Lógicamente este ultimo es destinado a crear un vector de este *struct* donde cada posición será un estado del autómata.

```
typedef struct
{
    int tamTraduccion;//Cantidad de simbolos de la traduccion
    char *traduccion;//cadena de caracteres con la traduccion
}traducciones;

typedef struct
{
     int numeroEstadosAFDT;//N° de estados del automata elegido.
     int numeroTransiciones;//N° de transiciones de un
estado=traducciones=destinos.
     enum Boolean esFinal;//TRUE es final, FALSE no es final.
     int nombreEstado;//Identificador del estado.
     char *transiciones;//Guarda las transiciones que salen del estado.
     traducciones *cad;//Guarda las traducciones respectivas a las
transiciones.
     int *destinos;//Guarda el destino de cada transicion respectivamente.
}AFDT;

AFDT *traductor;//Vector dinamico de tipo AFDT cada posicion es un estado.
```

Dentro del *main()*:
Se le pide al usuario con *imprimirMenu()* que traducción/AFDT quiere probar/usar.
Esta opción se le pasa al *aperturaFichero()* y este se encarga de abrir los *.txt* que almacenan cada uno un autómata, este método invoca a varios que se encargan de asignar memoria dinámica y vaciar memorias anteriores o temporales.

Por ejemplo, para leer el autómata del *apartado a)*, el fichero *Apartado1.txt* tiene este contenido:

```
n-1a/!:1
n-2a/!:3,b/4.cccc:2
n-2a/!:3,b/1.c:2
f*
```

Cada línea corresponde a un estado del autómata.
El **primer** carácter corresponde a:

- n: Estado no final.
- f: Estado final.

Dependiendo de si es o no es, se cambiará la variable *esfinal* a *TRUE* o *FALSE*.

El **segundo** carácter corresponde a:

- -: Este estado tiene transiciones.
- *: Este estado no tiene transiciones.

Dependiendo del carácter se asigna la memoria requerida en las variables y vectores, **transiciones*, **cad* y **destinos*, según lo leido en el **tercer** carácter.

El **tercer** carácter (solo en el caso de haber leído – como segundo carácter) corresponde al numero de transiciones. En el caso del *apartado a)*, la primera línea corresponde al estado *q1* y es por eso por lo que tras leer – hay un *1* pues solo tiene una transición en este caso a *q1*. Con esto ya podemos rellenar la demás información de este estado puestos todos los vectores y variables dinámicas de este estado (posición del vector de tipo *struct AFDT*) han sido asignados en memoria de manera correcta.

Tras esto guardamos en un vector *temp* todos los caracteres que faltan de esta línea/estado, en este caso *a/!:1* que es la información referente a todas las transiciones de este estado. Mediante el método *meterCharAlprincipio()*.

Este *temp* junto al número del linea/estado actual (iteración de un *for*) y numero de transiciones es pasado al método *rellenarVectores()* encargado de **transiciones*, **cad* y **destinos*.

El **cuarto** carácter es el símbolo que activa la transición en este caso *a* y se meterá en **transiciones*.

El **quinto** carácter es un centinela, es / e indica que el siguiente/s carácter/es son la traducción de la transición actual, este carácter se puede ver en otras líneas/estado del ejemplo.

El **sexto** carácter corresponde a:

- !: Esta transición no tiene traducción.
- *numero*: Indica el número de símbolos de la traducción.

Dependiendo de cada uno asignaremos la memoria correspondiente a **traduccion* con la que después leeremos la traducción d esta transición y la meteremos y también meteremos el número de símbolos de esta traducción en *tamTraduccion* para recorrerla más tarde.

El carácter ***.*** es otro valor centinela que se pone después de poner el número de símbolos de la traducción de una transición.

El carácter ***:*** va siempre después de la traducción de una transición para indicar la línea/estado destino de esta transición.

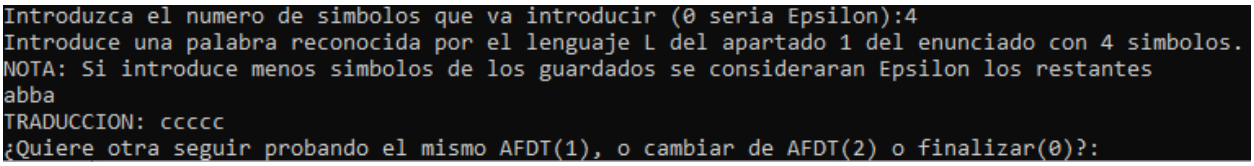
Tras esto el vector de tipo *struct AFDT* habrá sido rellenado y el autómata que hemos representado en un *.txt* ahora está cargado y listo para traducir “palabras” validas a su traducción correspondiente. El programa mostrara por salida estándar toda la información al automata elegido al principio, en este caso el que hemos puesto de ejemplo en esta explicación seria así:

```
               EL AFDT 1 TIENE 4 ESTADOS

DESGLOSE DE ESTADOS:
        - Estado: 0 (NO FINAL)
                - Transicion 0
                        - Mediante el simbolo 'a'
                        - Estado destino '1'
                        - Traduccion a --> palabra vacia (Epsilon)
        - Estado: 1 (NO FINAL)
                - Transicion 0
                        - Mediante el simbolo 'a'
                        - Estado destino '3'
                        - Traduccion a --> palabra vacia (Epsilon)
                - Transicion 1
                        - Mediante el simbolo 'b'
                        - Estado destino '2'
                        - Traduccion b --> cccc
        - Estado: 2 (NO FINAL)
                - Transicion 0
                        - Mediante el simbolo 'a'
                        - Estado destino '3'
                        - Traduccion a --> palabra vacia (Epsilon)
                - Transicion 1
                        - Mediante el simbolo 'b'
                        - Estado destino '2'
                        - Traduccion b --> c
        - Estado: 3 (ES FINAL)
                - NO tiene transiciones

Introduzca el numero de simbolos que va introducir (0 seria Epsilon):
```

Ahora meteremos una palabra invalida para este autómata, es decir, una palabra que nunca llegue al estado final, por ejemplo, épsilon, así que pondremos 0.
Automáticamente nos dirá que no hay traducción para esa palabra como es lógico y el programa terminara.

```
NO HAY TRADUCCION: La palabra vacia (Epsilon) no pertenece a L

Process returned -1073740940 (0xC0000374)   execution time : 4.447 s
```

Si metemos una palabra valida, por ejemplo, *abba* la traducción debería ser *ccccc* pero para ello antes tenemos que indicar que nuestra palabra tiene 4 símbolos.



Como vemos funciona.
Por último nos pregunta si queremos seguir con el mismo *AFDT* o cambiar a otro.

El *main* en si son 2 *do while* los cuales se encargan de traducir y son 2 porque puede ser que se vaya a querer volver a traducir o no. Se le introduce a mano lo que se quiere traducir y se traduce.