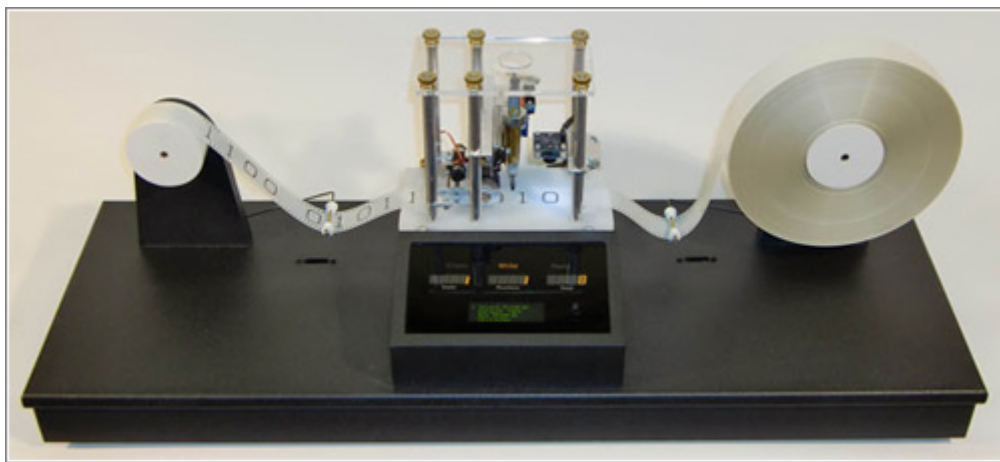


Algoritmos y Estructuras de Datos II

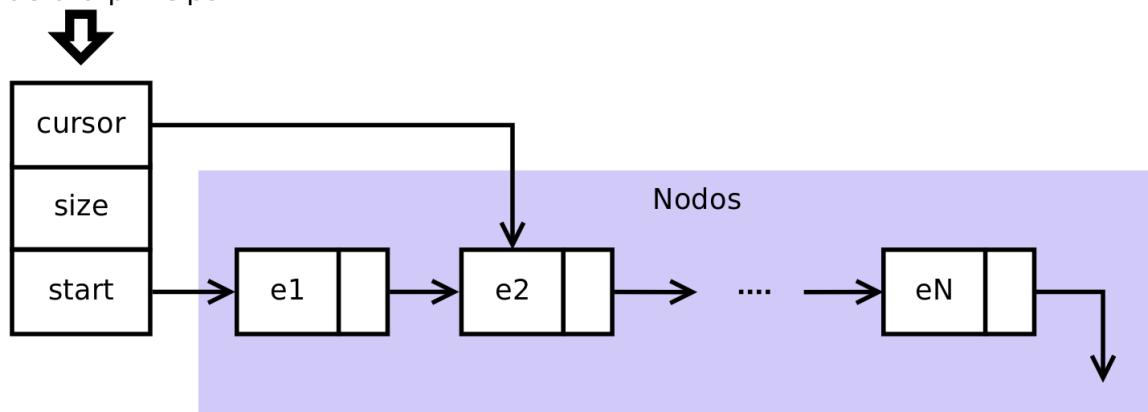
Recuperatorio Tema D - TAD Tape

Implementar el TAD **tape** que representa una *cinta de elementos*. El tipo abstracto está inspirado en las computadoras antiguas que manejaban la lectura y escritura de datos con cintas magnéticas (o a veces perforadas!).



Como se ve en la imagen, los elementos se encuentran en la cinta y hay un cabezal que lee el dato que tiene debajo. Abstractamente se puede pensar que hay un cursor, y cada vez que se mueve la cinta, el cursor apunta a otro elemento. Aunque estas máquinas pueden avanzar y retroceder la cinta, nuestro TAD sólo podrá avanzar y si necesita revisar un elemento anterior se puede “rebobinar” la cinta hasta el principio (es decir el cursor apunta al primer elemento). La representación consta de una estructura principal y una cadena de nodos simplemente enlazada:

Estructura principal

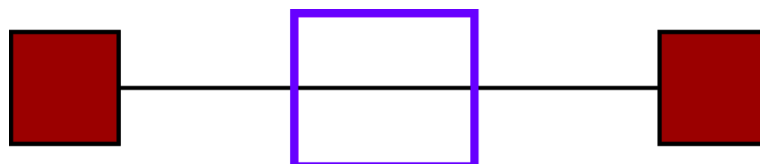


En el ejemplo la cinta cuenta con N elementos y el cursor está apuntando al segundo.

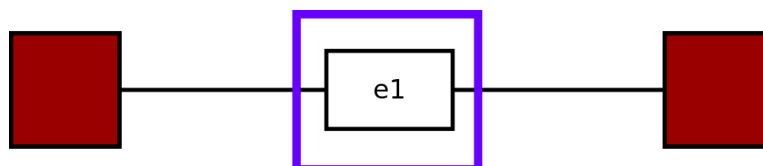
Las operaciones del TAD se listan a continuación:

Función	Descripción
<code>tape_t tape_create(void)</code>	Crea una cinta vacía
<code>tape_t tape_rewind(tape_t tape)</code>	Posiciona el cursor de la cinta en el primer elemento
<code>bool tape_at_start(tape_t tape)</code>	Indica si el cursor está al principio de la cinta
<code>bool tape_at_stop(tape_t tape)</code>	Indica si el cursor está apuntando fuera de la cinta
<code>bool tape_is_empty(tape_t tape)</code>	Indica si la cinta está vacía
<code>unsigned int tape_length(tape_t tape)</code>	Devuelve la cantidad de elementos en la cinta
<code>tape_t tape_step(tape_t tape)</code>	Avanza el cursor al siguiente elemento
<code>tape_t tape_insertl(tape_t tape, tape_elem e)</code>	Inserta un nuevo elemento e a la izquierda del elemento apuntado por el cursor.
<code>tape_t tape_insertr(tape_t tape, tape_elem e)</code>	Inserta un nuevo elemento e a la derecha del elemento apuntado por el cursor.
<code>tape_t tape_erase(tape_t tape)</code>	Elimina el elemento apuntado por el cursor
<code>tape_t tape_copy(tape_t tape)</code>	Devuelve una copia de la cinta tape en nueva memoria
<code>void tape_dump(tape_t tape);</code>	Muestra la cinta tape por pantalla
<code>tape_t tape_destroy(tape_t tape)</code>	Destruye la cinta liberando toda la memoria usada

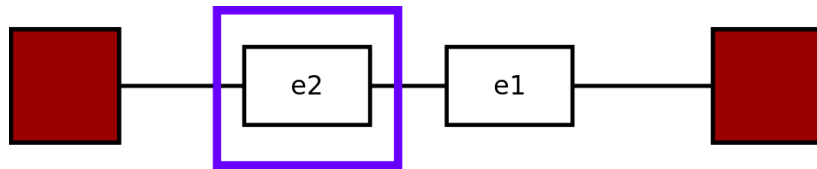
Cuando se crea una cinta nueva [`tape = tape_create()`], ésta se encuentra vacía:



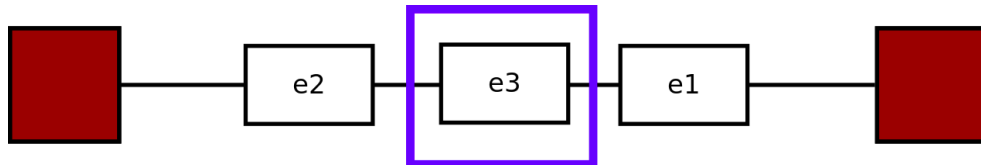
En el esquema abstracto, no hay elementos y el cursor está apuntando por fuera de los elementos. Esta situación se representa en el TAD cuando tanto **start** como **cursor** valen **NULL** (**size** vale 0). Cuando se agrega un nuevo elemento, ya sea por izquierda o por derecha, la cinta queda en el siguiente estado:



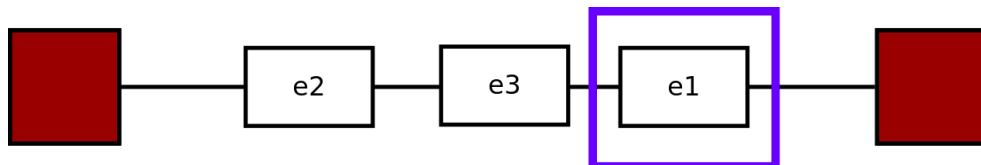
Es decir que el cursor está apuntando al elemento **e1** el cual es el único en la cinta en este momento. Si a partir de aquí si se agrega un elemento por izquierda [`tape_instertl()`] el resultado sería:



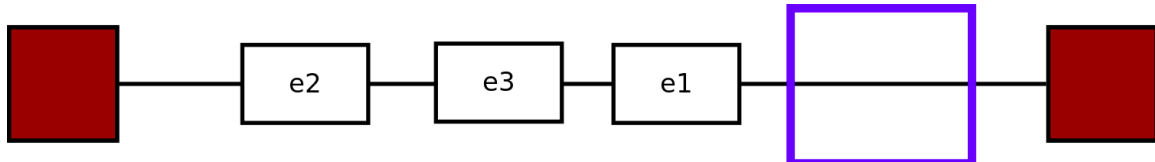
Notar que **e2** quedo a la izquierda que **e1** y el cursor apunta al nuevo elemento **e2**. Si ahora se agrega un elemento por derecha [**tape_insertr()**] el resultado:



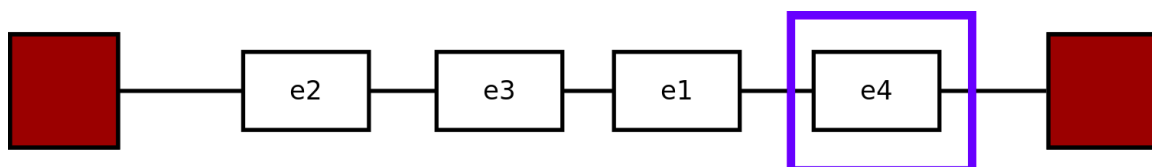
Entonces el elemento nuevo **e3** se agregó a la derecha de **e2** y el cursor se posicionó sobre el nuevo elemento. Para mover el cursor se hace de a un paso hacia adelante [**tape_step()**] que en este caso resulta en:



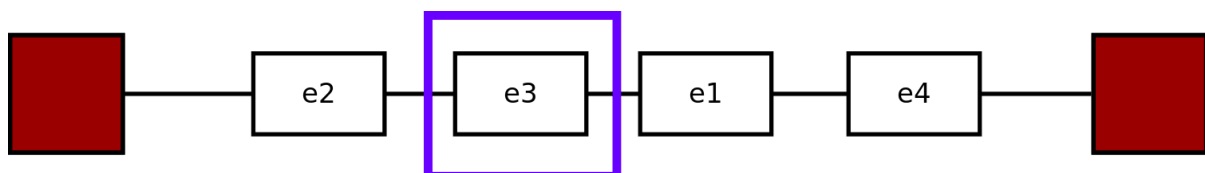
Si se da un paso más sucede lo siguiente:



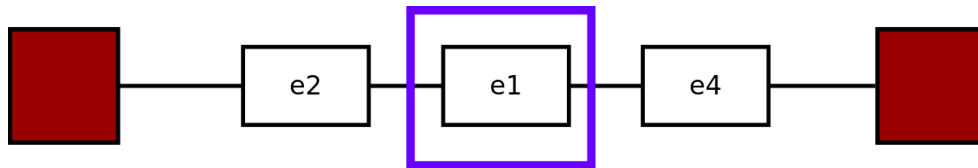
Cuando la cinta esta en este estado, **tape_at_stop(tape)** debe devolver **true** ya que no se puede seguir avanzando. En este punto en la representación el campo cursor de la estructura principal va a valer NULL. Aquí no se puede usar la operación de agregar por derecha (ya que no hay nada a la derecha el cursor), pero sí se puede agregar por izquierda:



Supongamos que llevamos el cursor a **e3** [**tape_rewind() + tape_step()**]:



Si borramos dicho elemento, el resultado sería:



Notar que el cursor quedó sobre el siguiente elemento al que se borró.

El programa resultante no debe dejar *memory leaks* ni lecturas/escrituras inválidas.

Se proveen archivos `insert1.c`, `insertr.c`, `middle1.c`, `middler.c` y `copy.c` que sirven para probar las operaciones del TAD. Además se incluye un Makefile para que sea sencillo compilar cada programa de prueba.

Archivo	Descripción
insert1.c	Prueba la operación <code>tape_insert1()</code> insertando uno a uno los caracteres de la cadena que se le pasa como parámetro en la línea de comandos: <code>\$./insert1 hola</code> Pasándole parámetros extra se le puede indicar las posiciones de elementos eliminar con sucesivas llamadas a <code>tape_erase()</code> : <code>\$./insert1 hola 1</code> en el ejemplo, primero se agregan todos los caracteres y luego se elimina el carácter en la posición <code>1</code> , es decir la <code>'o'</code> . Para posicionarse en el elemento se usan llamadas a <code>tape_rewind()</code> y <code>tape_step()</code>
insertr.c	Prueba la operación <code>tape_insertr()</code> y funciona de la misma manera que <code>insert1.c</code>
middle1.c	Sirve para probar la inserción en el medio de los elementos de la cinta. Inicialmente se insertan uno a uno los caracteres de la cadena que se le pasa primer parámetro en la línea de comandos. El segundo parámetro indica la posición a donde ubicar el cursor, donde luego se insertan los caracteres de la cadena del tercer parámetro usando sucesivas llamadas a la función <code>tape_insert1()</code> : <code>\$./middle1 hola 1 XYZ</code> inserta los caracteres XYZ con el cursor en la posición 1, es decir apuntando a <code>'o'</code> .
middler.c	Hace lo mismo que <code>middle1.c</code> pero usando <code>tape_insertr()</code>
copy.c	Sirve para probar <code>tape_copy()</code> . Se insertan uno a uno los caracteres de la cadena correspondiente al primer parámetro y luego se genera una copia de la cinta resultante. Ambas se muestran por pantalla: <code>\$./copy hola</code> Si se le da un tercer parámetro, luego de generar una copia de la cinta se agregan los caracteres uno a uno del tercer parámetro a la copia de la cinta. Finalmente muestra la cinta original y la copia: <code>\$./copy hola XYZ</code>

Para compilar el programa de `insert1.c` se usa

```
$ make insert1
```

Sí se quieren ejecutar pruebas ya definidas:

```
$ make test-insert1
```

La misma mecánica se puede usar para el resto de los programas de prueba. Para ejecutar todas las pruebas pre-definidas:

```
$ make test
```

Consideraciones:

- Solo se debe modificar el archivo `tape.c`
- Se provee el archivo `Makefile` para facilitar la compilación.
- Se recomienda usar las herramientas `valgrind` y `gdb`.
- Usando `make test` se compila y ejecutan todos los test, se recomienda solo hacerlo cuando se hayan verificado una a una cada operación
- Se recomienda probar manualmente un par de ejemplos usando `valgrind`
- Si el programa no compila, no se aprueba el parcial.
- Los *memory leaks* bajan puntos
- Entregar código muy impropio puede restar puntos
- Si `tape_length()` no es de orden constante $O(1)$ baja muchísimos puntos
- **Se debe hacer una invariante** que chequee consistencia entre los campos de la estructura principal.