Examen Final del Taller

Algoritmos y Estructuras de Datos II

Tarea

El alumno deberá implementar el tipo abstracto de dato *rcola* (ver especificación abajo) en el lenguaje de programación C, utilizando la técnica de ocultamiento de información visto en el taller de la materia.

Es requisito mínimo para aprobar lo siguiente:

- Implementar en C el TAD *rcola* (utilizando punteros a estructuras y manejo dinámico de memoria).
- Implementar en C una interfaz de línea de comando para que usuarios finales puedan usarlo (ver detalles más abajo).
- El programa resultado no debe tener *memory leaks* ni accesos inválidos a memoria, se chequeará tal condición usando `valgrind`.

Especificaciones

Se define el TAD *rcola*, que representa las colas reversibles.

TAD rcola[elem]
constructores
vacía : rcola
encolar : rcola $\times$ elem $\longrightarrow$ rcola
operaciones
es_vacía : rcola $\longrightarrow$ bool
primero : rcola $\longrightarrow$ elem
decolar : rcola $\longrightarrow$ rcola
revertir : rcola $\longrightarrow$ rcola

donde las operaciones primero y decolar se aplican solamente a colas no vacías.

operaciones auxiliares
colar : elem $\times$ rcola $\longrightarrow$ rcola
colar(e,vacía) = encolar(vacía,e)
colar(e_0,encolar(q,e_1)) = encolar(colar(e_0,q),e_1)

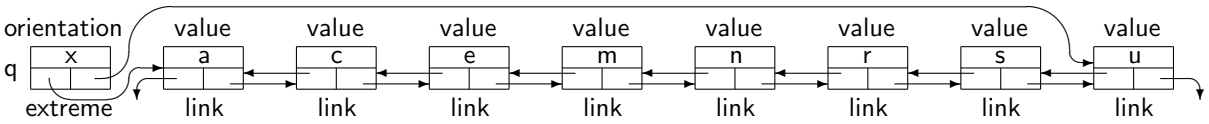
Si se implementara el TAD rcola con cualquiera de las representaciones vistas para cola, la implementación de la operación revertir resultaría lineal. Se pide implementar la rcola de modo de que todas las operaciones resulten constantes, por medio de la siguiente representación.

type node = **tuple**
value: elem
link: **array**[0,1] **of pointer to** node
end
type rqueue = **tuple**
orientation: {0,1}
extreme: **array**[0,1] **of pointer to** node
end

Cada nodo de la rcola tiene, además del valor que se aloja en él, un puntero link[0] al siguiente nodo de la izquierda y otro link[1] al siguiente nodo de la derecha. El nodo que se encuentra más a la izquierda de todos tiene link[0] = null, y el que se encuentra más a la derecha de todos, tiene link[1] = null. Esto se llama, lista doblemente enlazada.

La rcola está dada por un campo orientation, que según su valor sea 0 o 1 indica si la lista anterior debe leerse de izquierda a derecha o de derecha a izquierda, un puntero extreme[0] al nodo que se encuentra más a la izquierda de todos y otro extreme[1] al nodo que se encuentra más a la derecha. Si la rcola es vacía, extreme[0] = extreme[1] = null.

Gráficamente, la siguiente rcola q



implementa la rcola a, c, e, m, n, r, s, u si x es 0, y la rcola u, s, r, n, m, e, c, a si x es 1.

La magia capaz de hacer que la operación revertir sea constante es el campo orientation, que determina si la rcola debe leerse de izquierda a derecha o de derecha a izquierda. Para revertir, alcanza con indicar que la rcola se lea al revés de lo que se estaba leyendo.

Esta magia, sin embargo, complica un poco algunas de las demás operaciones. Por esa razón en algunas operaciones puede ser conveniente definir todas o algunas de las variables prev, next, fst, lst de tipo $\{0, 1\}$ para indicar el índice en que se encuentra el puntero al nodo anterior, siguiente, primero o último. En efecto, dada la variable q de tipo rcola y n, un nodo de la rcola q, después de las asignaciones:

prev:= q.orientation
next:= 1 - q.orientation
fst:= q.orientaton
lst:= 1 - q.orientation

se tiene que q.extreme[fst] es un puntero al primer nodo de la rcola, q.extreme[lst], al último, z.link[prev] es un puntero al nodo anterior al nodo z y z.link[next], al nodo posterior al nodo z.

Algunos puntos a tener en cuenta:

- Recordar que la estructura especificada es una estructura teórica, y que al momento de escribir el código y cumplir con los requerimientos solicitados puede ser necesario extender la respectiva estructura en C.

Importante: la especificación teórica del TAD no incluye un destructor del tipo, pero la implementación en C si debe proveerlo (y la interfaz de línea de comando debería usarlo).

La interfaz de línea de comandos (Libres)

Se deberá proveer además una interfaz de línea de comandos para manipular el TAD. Al inicio del programa deberá existir una cola vacía, sobre la cual se va a operar.

Las opciones a presentar al usuario final son:

- Agregar un elemento a la cola.
- Sacar e imprimir por pantalla el próximo elemento de la cola.
- Revertir la cola.
- Vaciar la cola.
- Imprimir la cola por pantalla.
- Salir de la aplicación.

El Makefile (libres)

Se deberá proveer un archivo `Makefile` que tenga dos reglas:

- La de compilación del ejecutable
- La de limpieza de los .o y otros archivos generados