Laboratorio 5: TAD cola
Algoritmos y Estructura de Datos II 2020

El objetivo de este laboratorio es AFIANZAR y REPASAR los conocimientos de encapsulamiento, abstracción y punteros adquiridos en laboratorios anteriores. La realización de este laboratorio será **OPCIONAL** y **SIN ENTREGA** pero la cátedra aconseja fuertemente hacer este laboratorio para clarificar conceptos importantes para la defensa general de los laboratorios.

Se pide como práctica implementar el TAD queue en lenguaje de programación c
Se aconseja estructurar el proyecto de la siguiente manera:

Main.c
queue.c
queue.h
list.c
list.h
helpers…

Tener en cuenta que una cola o queue es un TAD en el cual se remueve un elemento desde adelante y se agrega un elemento por detrás.

Las operaciones que deberán implementar de queue son:

- queue **queue_empty**(void): devuelve una nueva cola creada, vacía
- queue **queue_enqueue**(queue q, elem_t elem): encola el elemento elem a la cola q
- size_t **queue_size**(queue q): retorna el tamaño de la cola
- bool **queue_is_empty**(queue q): retorna True si la cola es vacía, falso en caso contrario.
- elem_t **queue_first**(queue q): retorna el primer elemento en la cola (el primer elemento agregado). No modica la queue q, solo retorna una COPIA del primer elemento
- queue **queue_dequeue(**queue q): desencola el primer elemento de la cola.
- queue **queue_free**(queue q): Libera los recursos pedidos al sistema por la cola q

Ejercicio 1

Con la definición de las operaciones de queue crea el archivo queue.h, que deberá tener los métodos declarados anteriormente. Completá la especificación de estos métodos con PRE condiciones y POS condiciones.

Ejercicio 2

Definí el archivo el archivo queue.c que contendrá la IMPLEMENTACIÓN de la cola especificada en queue.h. El TAD debe ser opaco (implementación oculta, recordar uso de typedef y de punteros para este fin). La operación queue_size debe ser de orden CONSTANTE (O(1)) por lo que se debe definir una estructura **struct queue_t** inteligente para este fin. Puedes utilizar para la implementación el TAD list definido en el práctico de set.

Ejercicio 3

Definí un archivo “main.c” con una función main que implemente al menos 3 casos de prueba de los métodos de cola. Un caso de prueba representa un CASO DE USO de nuestro tad cola. Por ejemplo, un caso de prueba podría ser verificar que en una cola con 2 elementos al llamar queue_first se obtenga EFECTIVAMENTE el primer elemento que se agregó a la cola. Al ejecutar los casos de prueba no se deberá observar ningún Memory leak o invalid read/write en la herramienta valgrind

Punto Estrella 1

Resulta tedioso construir manualmente las líneas de compilación. Existe una herramienta para evitar que tengamos que escribir las líneas una y otra vez llamada Make (https://es.wikipedia.org/wiki/Make). Este comando utiliza para compilar la información provista en un archivo especial llamado “Makefile” que suele estar en el mismo directorio donde está tu proyecto. Definí un archivo Makefile para compilar tu proyecto usando el comando “make”. Ayuda: **https://www.gnu.org/software/make/manual/html_node/Simple-Makefile.html**

Punto estrella 2

La implementación usando lista como se presenta en este ejercicio tiene la desventaja que el método dequeue/first no será de orden constante. ¿Como cambiaría la implementación del tad queue para todas la operaciones de queue sean de orden constante? Ayuda: Quizás necesites dejar de usar la implementación de list como la conocemos. Considerar una implementación de listas con puntero al primer y último elemento.