75.41 Algortimos y Programación II Curso 4
TDA Cola

Implementada como nodos enlazados

31 de marzo de 2019

1. Enunciado

Se pide implementar una Cola como nodos enlazados. Para ello se brindan las firmas de las funciones públicas a implementar y se deja a criterio del alumno la creación de las funciones privadas del TDA para el correcto funcionamiento de la Cola cumpliendo con las buenas prácticas de programación.

2. cola.h

```c
#ifndef __COLA_NE_H__
#define __COLA_NE_H__

#include <stdbool.h>
#include <stdlib.h>

typedef struct nodo {
        void* elemento;
        struct nodo* siguiente;
} nodo_t;

typedef struct cola {
        nodo_t* frente;
        nodo_t* final;
} cola_t;

/*
 * Crea una cola, reservando la memoria necesearia para almacenar la estructura.
 * Devuelve una refencia a la estructura cola_t creada o NULL en caso de no poder crearla
 */
cola_t* crear_cola();

/*
 * Encola un elemento. Reservando la memoria necesaria para el nuevo nodo.
 * Devuelve 0 si pudo encolar o -1 si no pudo.
 */
int encolar(cola_t* cola, void* elemento);

/*
 * Desencola un elemento. Liberando la memoria reservada para el nodo a eliminar.
 * Devuelve 0 si pudo desencolar o -1 si no pudo.
 */
int desencolar(cola_t* cola);

/*
 * Determina si la cola está vacia.
 * Devuelve true si está vacía y false si no.
 */
bool vacia(cola_t* cola);

/*
 * Devuelve el elemento el la primera posición de la cola o NULL en caso de estar vacía.
 */
void* primero(cola_t* cola);

/*
 * Destruye la cola liberando la memoria reservada por los nodos y por la propia estructura.
 * Devuelve 0 si pudo destruirla o -1 si no pudo.
```

```
 */
int destruir_cola(cola_t*);

#endif /* __COLA_NE_H__ */
```

3. Compilación y Ejecución

El TDA entregado deberá compilar y pasar las pruebas dispuestas por la cátedra sin errores, adicionalmente estas pruebas deberán ser ejecutadas sin pérdida de memoria.

Compilación:

```
gcc *.c -o cola_ne -g -std=c99 -Wall -Wconversion -Wtype-limits -pedantic -Werror -O0
```

Ejecución:

```
valgrind --leak-check=full --track-origins=yes --show-reachable=yes ./cola_ne
```

4. Minipruebas

Se les brindará un lote de minipruebas, las cuales recomendamos fuertemente sean ampliadas ya que no son exhaustivas y no prueban los casos borde, solo son un ejemplo de como encolar, desencolar y qué debería verse en la terminal en el **caso feliz**.

Minipruebas:

```
#include "cola_ne.h"
#include <stdio.h>

int main(){
        cola_t* cola = crear_cola();

        char elemento_1 = 'A';
        char elemento_2 = 'l';
        char elemento_3 = 'g';
        char elemento_4 = 'o';
        char elemento_5 = '2';

        for (int i = 0; i < 3; i++) {
                encolar(cola, &elemento_1);
                encolar(cola, &elemento_2);
                encolar(cola, &elemento_3);
                encolar(cola, &elemento_4);
                encolar(cola, &elemento_5);
        }


        for (int i = 0; i < 5; i++) {
                printf("%c\n", *(char*)primero(cola));
                desencolar(cola);
        }

        destruir_cola(cola);
        return 0;
}
```

La salida por pantalla luego de correrlas con valgrind debería ser:

```
==13424== Memcheck, a memory error detector
==13424== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==13424== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==13424== Command: ./cola_ne
==13424==
A
l
g
o
2
==13424==
==13424== HEAP SUMMARY:
==13424==     in use at exit: 0 bytes in 0 blocks
==13424==   total heap usage: 17 allocs, 17 frees, 1,280 bytes allocated
==13424==
==13424== All heap blocks were freed -- no leaks are possible
==13424==
```

```
==13424== For counts of detected and suppressed errors, rerun with: -v
==13424== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

5. Entrega

La entrega deberá contar con todos los archivos necesarios para compilar y ejecutar correctamente el TDA.

Dichos archivos deberán formar parte de un único archivo **.zip** el cual será entregado a través de la plataforma de corrección automática **Kwyjibo**.

El archivo comprimido deberá contar, además del TDA con:

- El archivo con las pruebas agregadas para comprobar el correcto funcionamiento del TDA.
- Un **Readme.txt** donde se deberá explicar qué es lo entregado, como compilarlo (línea de compilación), como ejecutarlo (línea de ejecución) y todo lo que crea necesario aclarar.
- El enunciado.