Córdoba, 12 de Agosto del 2022

Examen Final

Algoritmos y Estructuras de Datos II - Taller

Colas Circulares

Una cola circular es un tipo abstracto de datos *Cola* implementado con un *array* circular. Esto quiere decir que mientras se van encolando elementos nuevos, se va llenando la cola pero al desencolarse elementos se liberan posiciones de manera circular en el arreglo. Veamos el siguiente ejemplo con una cola tamaño 3:
Inicialmente la cola estará vacía:

			0	1	2
1)	empty queue	->			
2)	enqueue(1)	->	1		
3)	enqueue(7)	->	1	7	
4)	enqueue(12)	->	1	7	12
5)	dequeue	->		7	12
6)	dequeue	->			12
7)	enqueue(17)	->	17		12

En este ejemplo es claro como la cola se va llenando y vaciando circularmente. En (1) `queue_is_empty` sería `true` pues la cola no tiene elementos. En (4) `queue_is_full` sería `true` pues la cola está llena. En este punto es imposible agregar nuevos elementos con `enqueue` a menos que se aplique una operación de tipo `dequeue` antes. Luego de aplicar (5) y (6) la cola tiene 2 nuevas posiciones disponibles para almacenar elementos, 0 y 1. Al ejecutar (7) la cola deberá guardar el elemento nuevo en la primera posición disponible: 0. De esta manera, la cola tiene el primer elemento en la posición 2: **12** y el segundo elemento en la posición 0: **17**.

Ejercicio 1

Implementar el TAD Cola con array circulares

Las funciones a implementar del TAD Cola son las siguientes:

Función	Descripción
`queue queue_empty(size_t max_size)`	Crea una cola vacía de tamaño máximo `max_size`
`queue queue_enqueue(queue q, queue_elem e)`	Inserta el elemento `e` al final de la cola
`queue queue_dequeue(queue q)`	Remueve el primer elemento de la cola. Sólo válido para colas no vacías
`size_t queue_size(queue q)`	Obtiene el tamaño de la cola. Debe ser de orden constante.
`queue_elem queue_first(queue q)`	Obtiene el primer elemento de la cola Sólo aplica a una cola no vacía.
`bool queue_is_empty(queue q)`	Verifica si la cola está vacía.
`void queue_print(queue q)`	Imprime los elementos de la cola en pantalla. Los elementos de la cola deben ser impresos en orden de la cola (no del array circular) **WARNING:** no debe alterar la estructura de la cola
`queue queue_destroy(queue q)`	Libera toda la memoria usada por la cola.
bool queue_is_full(**queue** q)	Verifica si la cola está llena

Se deben completar además las definiciones para la estructura principal . En la estructura principal se pueden incluir los campos que se necesiten para que `queue_size()` pueda calcularse en orden constante.

Para compilar el test se puede usar el **`Makefile`** incluído haciendo:

```
$ make test
```

y luego, para ejecutar el programa de prueba:

```
$ ./test
```

Este test ejecutará la secuencia de pasos descripta en el ejemplo de colas circulares, imprimiendo luego de cada paso. Se debería obtener un resultado SIMILAR al presentado acá.

Ejercicio 2

En el archivo **`main.c`** hay que implementar 3 casos de prueba.

Test1

1. Crear una cola vacía tamaño `max_size`
2. Verificar que la cola esté vacía imprimiendo la leyenda **`"Queue is empty"`** si lo está
3. Imprimir la cola

Test2

1. Crear una cola vacía tamaño `max_size`

2. Imprimir la cola
3. Introducir `max_size` elementos en orden consecutivo.
   Ejemplo: Si `max_size` es `3` introducir `1`, `2` y `3` a la cola
4. Imprimir la cola
5. Verificar que la cola esté llena llamando al método correspondiente.
   Imprimir **`"Queue is full"`** si lo está.

Test3

1. Crear una cola vacía tamaño `max_size`
2. Imprimir la cola
3. Introducir `max_size` elementos en orden consecutivo.
   Ejemplo: Si `max_size` es `3` introducir `1`, `2` y `3` a la cola
4. Imprimir la cola
5. Verificar que la cola esté llena llamando al método correspondiente. Imprimir **`"Queue is full"`** si lo esta.
6. Obtener el primer elemento de la cola. Imprimirlo
7. Desencolar 2 elementos
8. Imprimir la cola
9. Encolar 2 elementos adicionales. En el caso del ejemplo: `4` y `5`.
10. Imprimir la cola

Para compilar el ejercicio se puede usar el **`Makefile`** incluido de la siguiente manera:

```
$ make
```

Luego el programa se ejecuta como sigue:

```
$ ./queue_check 10
```

Este programa ejecutará las funciones test1, test2 y test3 para una cola de tamaño 10

Ejercicios para Libres

Los alumnos libres deberán implementar una función adicional

Función	Descripción
`queue_elem*` `queue_to_array(`**`queue`** `q)`	Crea un arreglo con memoria dinámica, guarda los elementos de la cola EN ORDEN. **WARNING** este método no debe alterar la estructura de la cola q

Además deberán implementar en **`main.c`** un Test4 para probar la funcionalidad de `queue_to_array`

Test4

1. Crear una cola vacía tamaño `max_size`
2. Imprimir la cola
3. Introducir `max_size` elementos en orden consecutivo.
   Ejemplo: Si `max_size` es `3` introducir `1`, `2` y `3` a la cola
4. Imprimir la cola
5. Verificar que la cola esté llena llamando al método correspondiente. Imprimir **`"Queue is full"`** si lo está.
6. Obtener el primer elemento de la cola. Imprimirlo
7. Desencolar 2 elementos
8. Imprimir la cola
9. Encolar 1 elemento adicional. En el caso del ejemplo: `4`.
10. Imprimir la cola
11. Construir un array a partir de la cola
12. Imprimir el arreglo construido.

Consideraciones

- Si **`queue.c`** no compila, no se aprueba el examen.
- Si **`main.c`** no compila muy difícilmente se apruebe el examen.
- Si **`queue_size()`** no es de orden constante baja muchísimos puntos
- No implementar la invariante baja puntos
- No chequear pre y post condiciones baja puntos
- Los *memory leaks* bajan puntos
- Entregar código muy improlijo resta puntos
- Se provee el archivo **`Makefile`** para facilitar la compilación.
- Se recomienda usar las herramientas **`valgrind`** y **`gdb`**.