Tipos Abstratos de Datos (TADs)

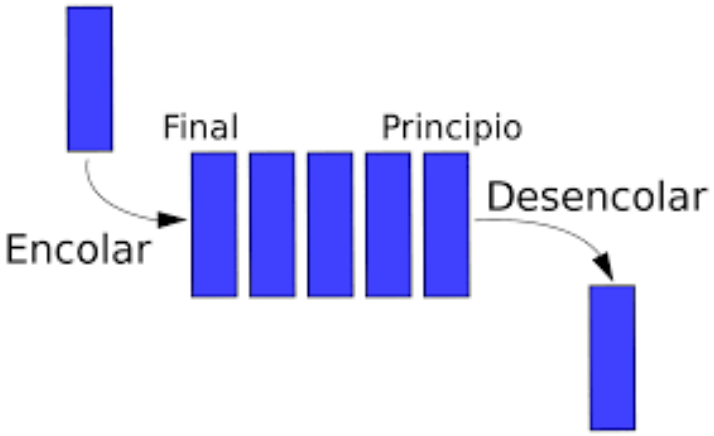
Cola

Cola

- Colección de elementos (también se las conoce como **queue**)
- Se agregan elementos por el final y se eliminan por el frente:

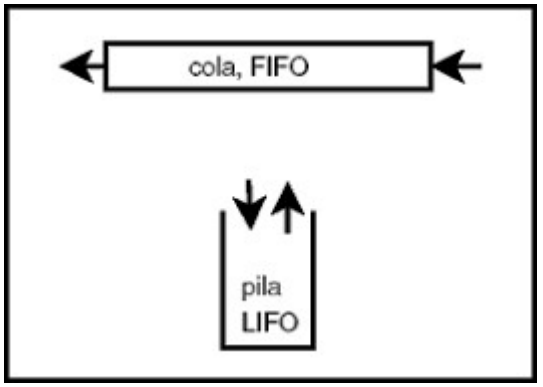
Cola

- Estructura FIFO: *first in first out*: primero en entrar primero en salir!



Cola

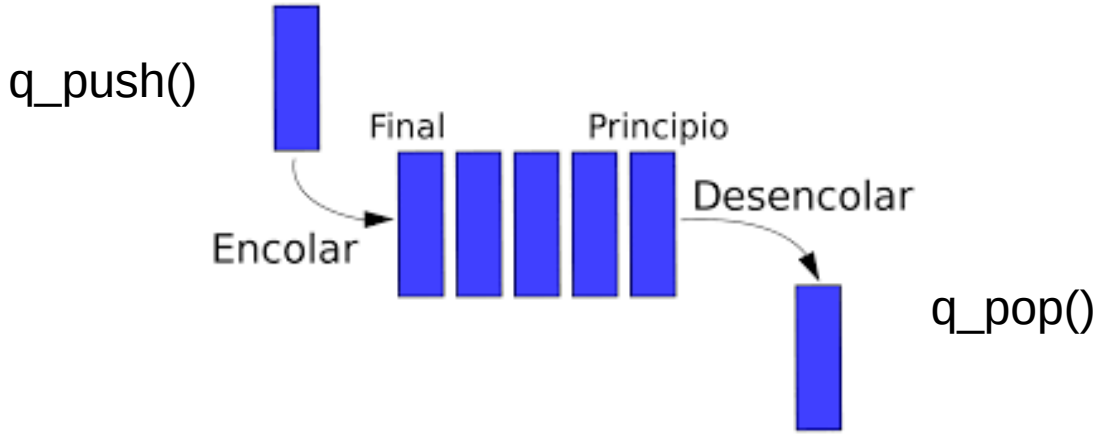
- Estructura FIFO: *first in first out*: primero en entrar primero en salir!
- Notar la diferencia entre una estructura FIFO y una LIFO.



Cola

- Las operaciones usuales en una cola (entre otras) es **agregar** y **eliminar** elementos. En este contexto se llaman **push** y **pop** respectivamente.

  Nota: se usará la **q** de **queue** para determinar: q_push() y q_pop() operaciones sobre colas.



La cola puede estar vacía → error al hacer q_pop()!!
La pila puede estar llena si se implementa con arreglos → error al hacer q_push()!!

Cola

Operaciones:

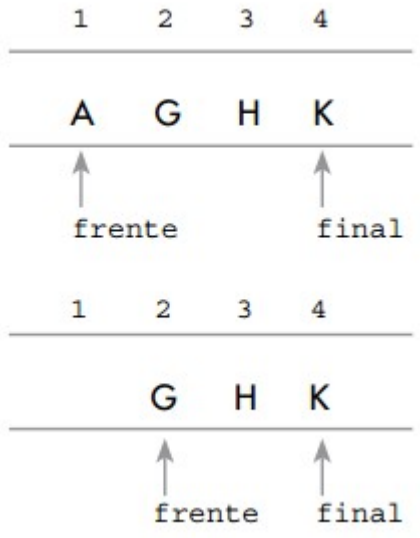
Crearcola	Inicia la cola como vacía
Insertar	Añade un dato por el final de la cola
Quitar	Retira (extrae) el elemento frente de la cola
Cola vacía	Comprobar si la cola no tiene elementos
Cola llena	Comprobar si la cola está llena de elementos
Frente	Obtiene el elemento frente o primero de la cola
Tamaño de la cola	Número de elementos máximo que puede contener la cola

Implementaciones:

- Con arreglos (implementación estática)
- Con punteros: listas enlazadas (implementación dinámica)

Cola

- Implementación con arreglos: incluye una lista o colección de elementos (arreglo), y 2 índices para mantener la información del frente y del final de la cola.

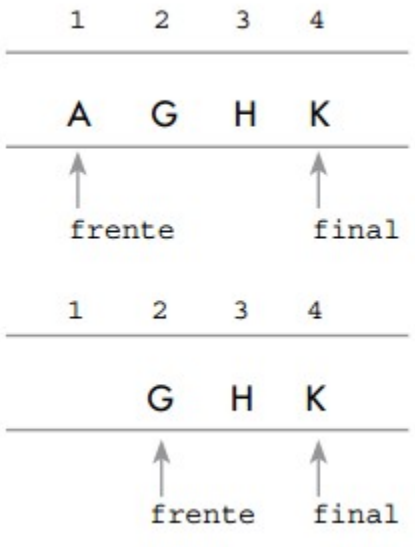


posición de `frente` y `final` después de extraer.

Se van agregando elementos al final de la cola (q_push()) y se eliminan por el frente de la cola (q_pop()). En la implementación con arreglos hace falta verificar que la cola no esté vacia al eliminar ni llena al agregar.

Cola

- Implementación con arreglos: incluye una lista o colección de elementos (arreglo), y 2 índices para mantener la información del frente y del final de la cola.

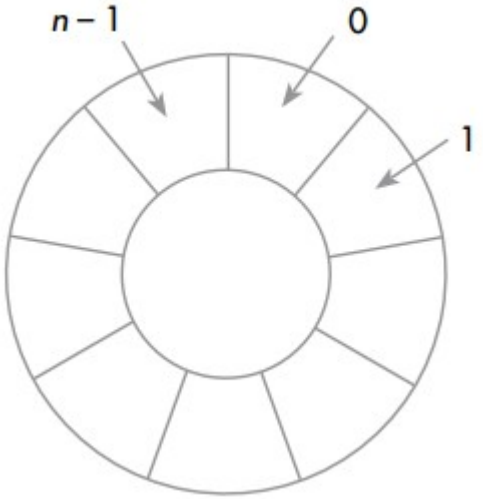


posición de `frente` y `final` después de extraer.

El avance lineal de frente y final tiene el grave problema de dejar huecos a la izquierda del arreglo, Puede ocurrir que final alcance el índice más alto del arreglo, sin que puedan insertarse nuevos elementos y sin embargo existen posiciones libres a la izquierda de frente.

Cola

- Implementación con arreglos: incluye una lista o colección de elementos (arreglo), y 2 índices para mantener la información del frente y del final de la cola.

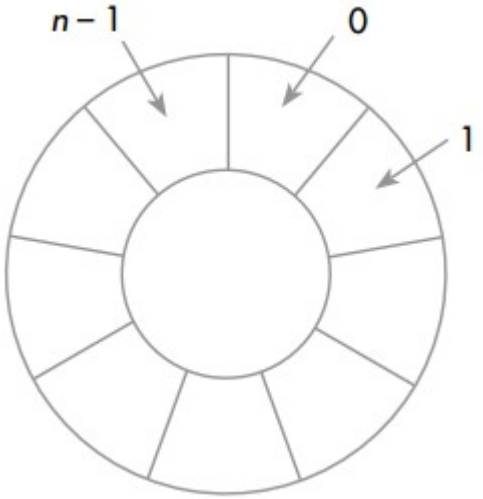


Para solucionar esto se usan arreglos circulares: se une el final del arreglo con el principio.

Esto es una visión lógica del arreglo, físicamente el arreglo sigue siendo lineal.

Cola

- Implementación con arreglos: incluye una lista o colección de elementos (arreglo), y 2 índices para mantener la información del frente y del final de la cola.



Para solucionar esto se usan arreglos circulares: se une el final del arreglo con el principio.

Esto es una visión lógica del arreglo, físicamente el arreglo sigue siendo lineal.

Se siguen usando frente y final para indicar la posición del elemento de la cabeza de la cola y la posición del elemento del final, donde se almacenó el último elemento agregado.

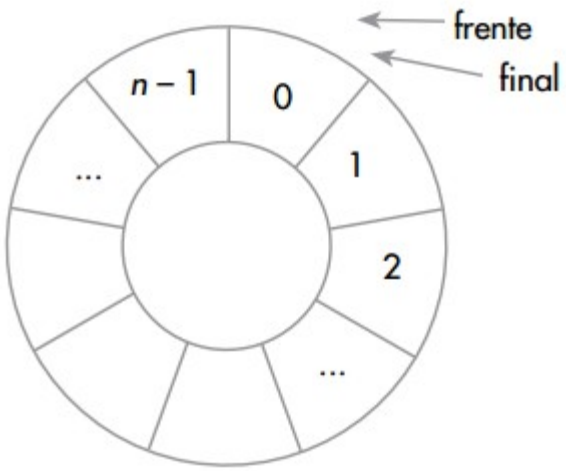
Cola

- La variable frente es siempre la posición del primer elemento de la cola (FIFO) y avanza en el sentido de las agujas del reloj. La variable final es la posición de la última inserción. Una nueva inserción supone mover final cirularmente a la derecha y asignar un nuevo elemento.
- Se usa la teoría del resto y se genera índices de 0..MAX - 1.

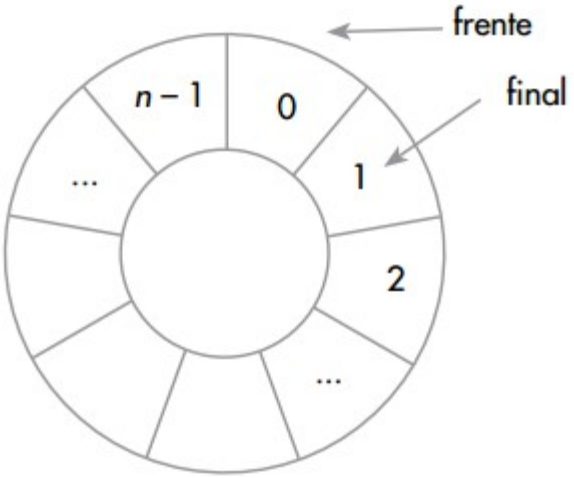
Mover `final` adelante = `(final + 1) % MAXTAMQ` → q_push()

Mover `frente` adelante = `(frente + 1) % MAXTAMQ` → q_pop()

Cola

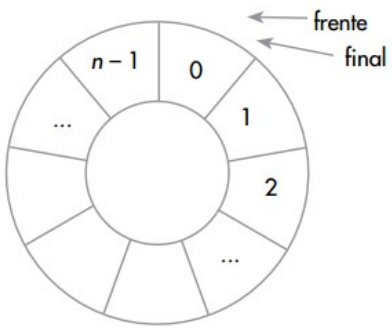


Cola vacía
(frente = final = 0)

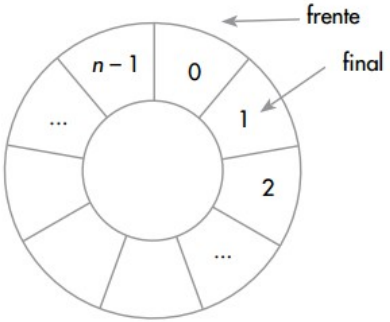


Cola después
de un q_push()

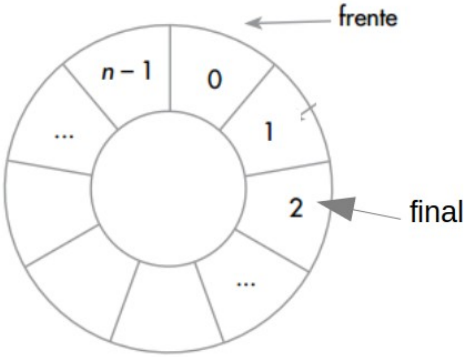
Cola



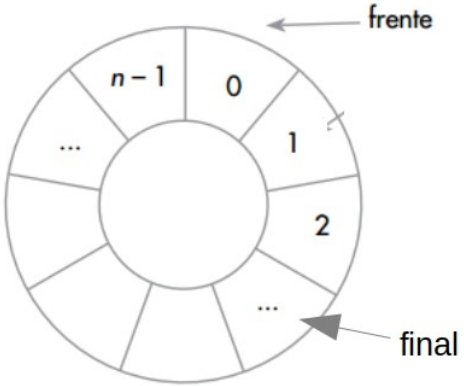
Cola vacía
(frente = final = 0)



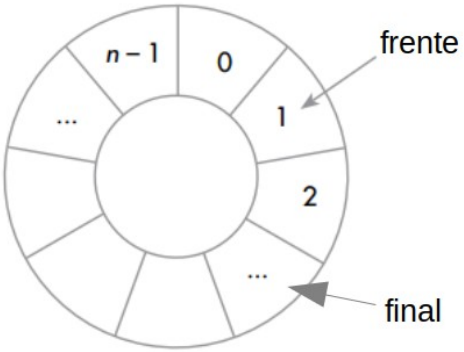
Cola después
de un q_push()
final = (final +1) % MAX



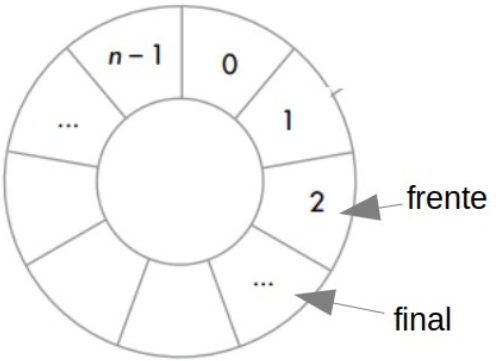
Cola después
de un segundo q_push()
final = (final +1) % MAX



Cola después
de un tercer q_push()
final = (final +1) % MAX



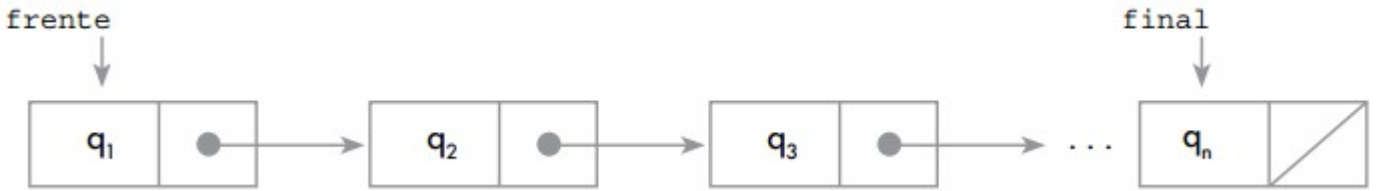
Cola después
de un q_pop()f
frente = (frente + 1) % MAX



Cola después
de otro q_pop()
frentel = (frente +1) % MAX

Cola

- Una cola implementada con arreglos fuerza a la definición de una máxima cantidad posible de elementos, lo que lleva a que una cola se llene y falte espacio en la cola o que se reserve espacio demás y se desperdicie.
- La implementación de **colas con punteros** ofrece una alternativa dinámica: el espacio se reserva y se libera en tiempo de ejecución, a medida que la aplicación lo requiera.



Cola con lista enlazada.

Cola

- Se manejarán dos tipos de datos: el tipo nodo de la lista y otro tipo para agrupar a las variables frente y final, este lo denominaremos Cola en el siguiente ejemplo:

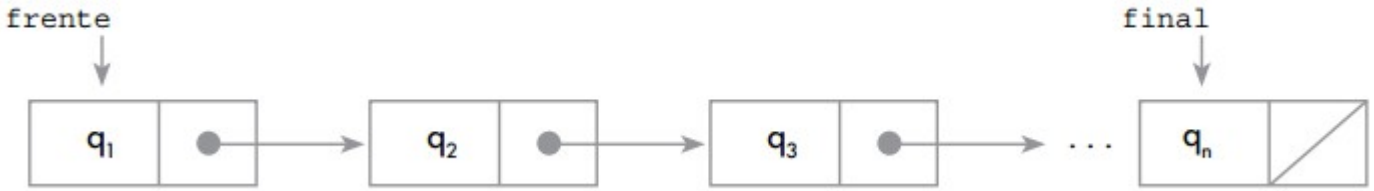
```
struct nodo
{
  TipoDato elemento;
  struct nodo* siguiente;
};
typedef struct nodo Nodo;
typedef struct
{
  Nodo* frente;
  Nodo* final;
}Cola;
```

Cola con lista enlazada.

Prototipo de las funciones

```
/* prototipos de las operaciones */
void crearCola (Cola* cola);
void insertar (Cola* cola,TipoDato entrada);
TipoDato quitar (Cola* cola);
  void borrarCola (Cola* cola); /* libera todos los nodos */
/* acceso a la cola */
TipoDato frente (Cola cola);
/* métodos de verificación del estado de la cola */
int colaVacia (Cola cola);
```

Cola




```
struct nodo
{
  TipoDato elemento;
  struct nodo* siguiente;
};
typedef struct nodo Nodo;
typedef struct
{
  Nodo* frente;
  Nodo* final;
}Cola;
```

Cola con lista enlazada.