Estructuras Colas

Queues

Estructura Cola (FIFO)

- La cola (queue) es también una estructura lineal restrictiva en la que solo podemos poner y sacar elementos respetando la siguiente restricción:

- El primer elemento en llegar a la cola será también el primero en salir.

FIFO – First In First Out

- Para comprender este concepto, basta con imaginar la cola que se forma en la caja de un supermercado donde la cajera atiende a los clientes que forman la cola respetando el orden de llegada; por lo tanto, el primero que llegó también será el primero en ser atendido y en salir.

- Una cola es otro tipo especial de lista en la cual los elementos se insertan en un extremo (el posterior) y se suprimen en el otro (el anterior o frente).

- Las operaciones para las colas son análogas a las de las pilas; las diferencias sustanciales consisten en que las inserciones se hacen al final de la lista, y no al principio.

Operaciones con colas

1. Anula(C) convierte la cola C en una lista vacía.
2. Frente(C) es una función que devuelve el valor del primer elemento de la cola C.
3. Pone_En_Cola(x, C) inserta el elemento *x* al final de la cola C.
4. Quita_De_Cola(C) suprime el primer elemento de C.
5. Vacia(C) devuelve verdadero si, y solo si, C es una cola vacía.

Implementación Estática

Colas

Queues

Implementación Estática – Colas

- Usaremos un simple arreglo para implementar una cola estática.
- Llamamos a la operación insertar un elemento en una cola ENQUEUE
- Llamamos DEQUEUE a la operación eliminar un elemento de la cola.
- La cola tiene cabeza y cola.

- La propiedad FIFO de una cola hace que funcione como una línea de clientes esperando pagarle a un cajero.

- Cuando un elemento se pone en cola, ocupa su lugar al final de la cola, al igual que un cliente recién llegado ocupa un lugar al final de la fila.

- El elemento quitado de la cola es siempre el que está en la cabecera de la cola, como el cliente en la cabecera de la fila que ha esperado el tiempo más largo.

- La Figura 1 muestra una forma de implementar una cola de como máximo ***n-1*** elementos utilizando un arreglo **Q[1..n]**.

- La cola tiene un atributo ***Q.head*** que indexa, o apunta a su cabeza.

- El atributo ***Q.tail*** indexa a la siguiente ubicación en la que un elemento recién llegado se insertará en la cola.

- Los elementos de la cola residen en las ubicaciones **Q.head**, **Q.head+1**, ..., **Q.tail-1**, donde se entiende que la ubicación **1** sigue inmediatamente después de la ubicación **n** en un orden circular.

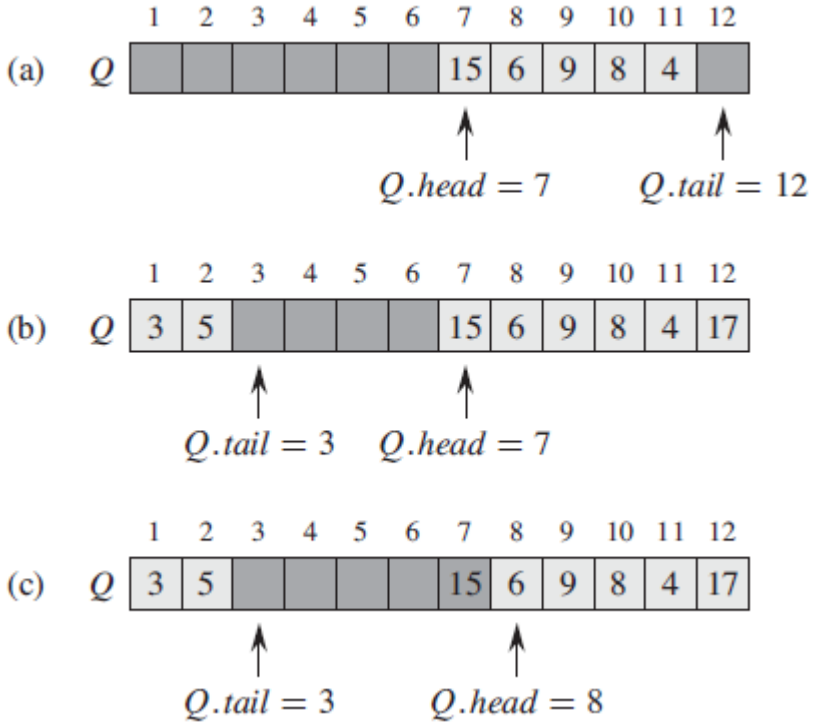


Figura 1. Una cola implementada usando un arreglo Q[1...12].

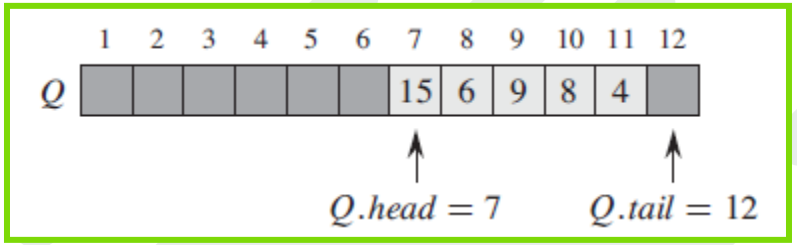
Los elementos de la cola aparecen solo en las posiciones ligeramente sombreadas.

A. La cola tiene 5 elementos, en las ubicaciones Q[7...11].

B. La configuración de la cola después de las llamadas:

ENQUEUE(Q,17)
ENQUEUE(Q, 3)
ENQUEUE(Q, 5)

C. La configuración de la cola después de la llamada DEQUEUE(Q) devuelve el valor de 15 que esta formado en la cabeza de la cola.
La nueva cabeza tiene el valor de 6.



- La **cola está vacía** cuando se cumple que:

  Q.head = Q.tail

  Inicialmente, tenemos ***Q.head = Q.tail = 1***.

  Si intentamos sacar un elemento de una cola vacía, la cola se desborda.

- La **cola está llena** cuando:

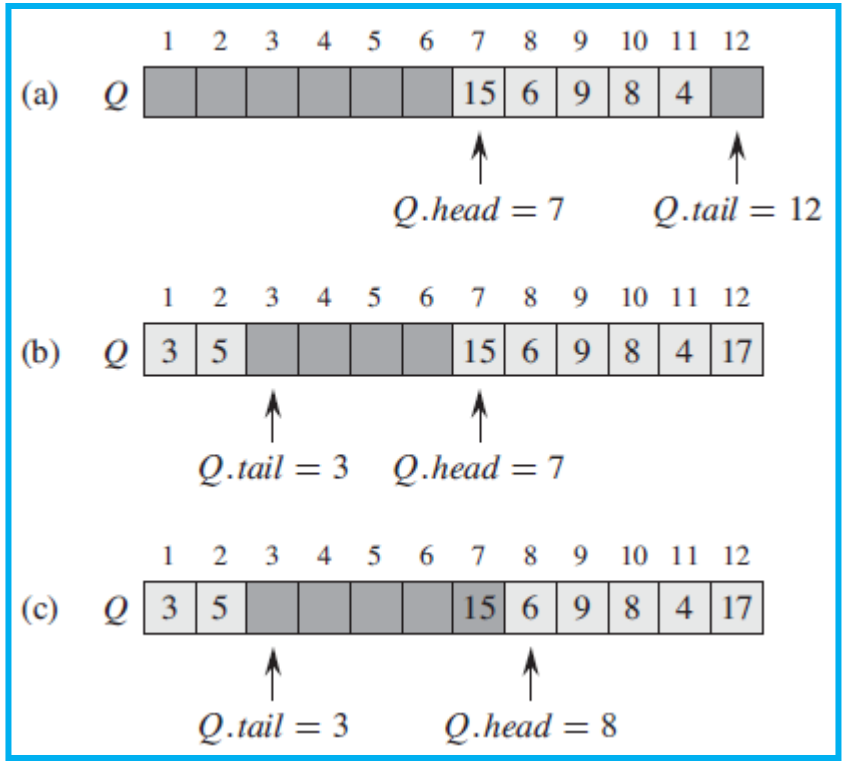
  Q.head = Q.tail+1
  Q.head = 1 y Q.tail = Q.length

  Si intentamos poner en cola un elemento, entonces la cola se desborda.

- El pseudocódigo asume que ***n = Q.length***

```
ENQUEUE(Q, x)
1  Q[Q.tail] = x
2  if Q.tail == Q.length
3        Q.tail = 1
4  else Q.tail = Q.tail + 1
```

```
DEQUEUE(Q)
1  x = Q[Q.head]
2  if Q.head == Q.length
3        Q.head = 1
4  else Q.head = Q.head + 1
5  return x
```



- En nuestros procedimientos ENQUEUE y DEQUEUE, hemos omitido la comprobación de errores para:

  - Subdesbordamiento (cuando se quiere sacar un elemento y la cola está vacía)

  - Desbordamiento (cuando se quiere agregar un elemento y la cola está llena).

Implementación Dinámica

Colas

Queues

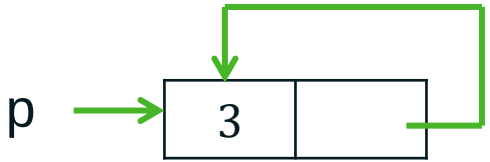
Implementación Dinámica - Colas

- La cola se implementará sobre un caso particular de lista enlazada: la lista enlazada circular.

Lista enlazada circular

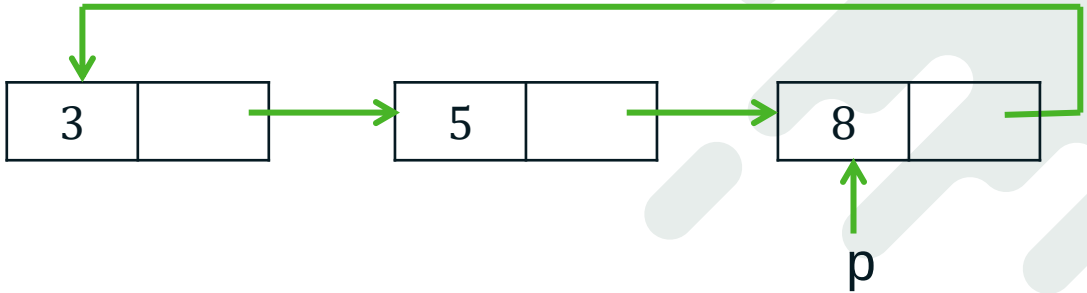
- Una lista circular es una lista enlazada en la que el último nodo apunta al primero.

- Una lista circular con un único elemento.



- Una lista circular con más de un elemento.

- Dado que en una lista circular el último nodo tiene una referencia al primero resultará más provechoso mantener a **p** apuntando al último nodo de la lista ya que desde allí podremos acceder al primero, que estará referenciado por **p->sig**.

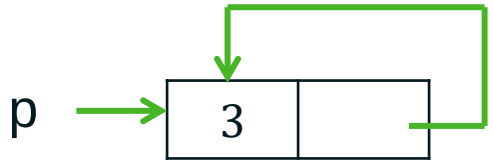


- Así, en una lista circular siempre tenemos referenciado el último nodo que agregamos (**p**) e, indirectamente, también tenemos referenciado el primero (**p->sig**)

Implementar una cola sobre una lista circular

- Para la implementación de la estructura cola. Necesitamos una lista enlazada con dos punteros: uno al primer nodo y uno al último. Justamente esta característica la encontramos en una lista circular.
- Luego, para encolar un elemento lo agregamos al final de la lista y para desencolar eliminaremos al primero de la lista.
- Analicemos una cola inicialmente vacía.

- Ahora encolamos el valor 3:

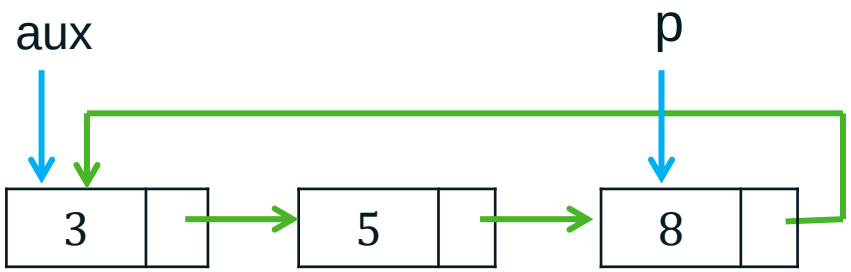


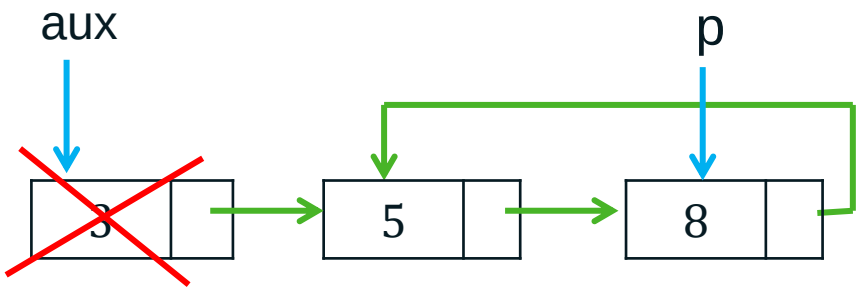
- Encolemos el valor 5:

- Encolemos el valor 8

Como vemos, el final de la lista siempre está apuntado por **p** y el inicio esta referenciado por el **p->sig**.

- El primer valor que encolamos es el 3 y es el que debe salir si queremos desencolar un elemento.
- Para esto, según nuestro ejemplo, tenemos que hacer que el siguiente de 8 apunte al siguiente de 3 (es decir, al 5).
- Si con una variable ***aux*** apuntamos al siguiente de **p** entonces para desencolar simplemente hacemos que el siguiente de **p** apunte al siguiente de ***aux*** y luego liberamos ***aux***.





Operación encolar

- La función encolar agrega un elemento al final de la lista circular direccionada por **p**.

```
void encolar(Nodo** p, int v)
{
        Nodo* nuevo = (Nodo*) malloc(sizeof(Nodo));
        nuevo->valor = v;
        if( *p == NULL )
        {
                nuevo->sig = nuevo;
        }
        else
        {
                nuevo->sig = (*p)->sig;
                (*p)->sig = nuevo;
        }
        *p = nuevo;
}
```

Operación desencolar

- Veamos ahora cómo desencolar considerando el caso particular que se da al desencolar el último elemento de la cola.

- Esta situación la identificamos cuando resulta que:

  (*p)->sig es igual a **p**.

Operación desencolar

```
int desencolar(Nodo** p)
{
        int ret = (*p)->sig->valor;
        if( *p == (*p)->sig )
        {
                free(*p);
                *p = NULL;
        }
        else
        {
                Nodo* aux = (*p)->sig;
                (*p)->sig = aux->sig;
                free(aux);
        }
        return ret;
}
```

```
int main()
{
        Nodo* p = NULL;

        // encolamos varios elementos
        encolar(&p, 1);
        encolar(&p, 2);
        encolar(&p, 3);

        // desencolamos un elemento (sale el 1)
        printf("%d\n", desencolar(&p));

        // desencolamos un elemento (sale el 2)
        printf("%d\n", desencolar(&p));

        // encolamos más elementos
        encolar(&p, 4);
        encolar(&p, 5);
        encolar(&p, 6);

        // desencolamos un elemento (sale el 3)
        printf("%d\n", desencolar(&p));

        // desencolamos mientras queden elementos en la cola
        while( p != NULL )
                printf("%d\n", desencolar(&p));
}
```

La salida será:

1
2
3
4
5
6

Aplicaciones

Colas

Aplicaciones de la estructura Cola

1. Cuando los trabajos se envían a una impresora, se organizan por orden de llegada. Por lo tanto, esencialmente, los trabajos enviados a una impresora se colocan en una cola.

2. Prácticamente todas las filas de la vida real son una cola. Por ejemplo, las filas en las taquillas de boletos, son colas, porque el servicio se asigna por orden de llegada.

3. Otro ejemplo se refiere a las redes informáticas. Hay muchas configuraciones de red de computadoras personales en las que el disco está conectado a una máquina, conocida como servidor de archivos.

   Los usuarios de otras máquinas tienen acceso a los archivos por orden de llegada, por lo que la estructura de datos es una cola.

4. Las llamadas a grandes empresas generalmente se colocan en una cola cuando todos los operadores están ocupados.

5. Existe toda una rama de las matemáticas conocida como teoría de las colas se ocupa del cómputo probabilístico, cuánto tiempo esperan los usuarios en una cola, que tan larga es la fila y preguntas similares.

- La respuesta depende de la frecuencia con la que llegan los usuarios a la fila y el tiempo que se tarda en procesar a un usuario una vez que se empieza a atender.

- Ambos parámetros se dan como funciones de distribución de probabilidad.