capítulo cinco

colas

Otro caso especial de la estructura de datos de *lista lineal*, que se introdujo en el capítulo 4, es la *cola*. Mientras que en las pilas se restringe la adición y supresión de elementos a través de un solo extremo, llamado tope de la lista, a las colas se les restringe a que los elementos se supriman por el frente y se agreguen por atrás. De hecho, ya estamos familiarizados con las colas, pues muchos ejemplos aparecen en nuestra vida diaria. Encontramos ejemplos de colas en los supermercados, en los bancos y en las oficinas de correos. Los automóviles generalmente avanzan en colas a través de intersecciones. También, es frecuente encontrar colas en los sistemas de información basados en computadoras. Las órdenes de clientes entran en una cola para ser despachadas; los productos entran en colas de inventarios; las tareas a ejecutar por una computadora quedan ordenadas en una cola para esperar a ser atendidas por la Unidad Central de Procesamiento.

En este capítulo, se introducen las operaciones que definen la estructura de datos de tipo cola; después, se describen las facilidades para declarar y manipular colas en COBOL y Pascal. Por último, se analizan al final del capítulo algunos ejemplos de aplicación que usan colas.

DEFINICIONES

Una *cola* es un caso especial de la estructura general de datos definida en el capítulo anterior como una Lista lineal:

$$A = [A_1, A_2, \ldots, A_T].$$

En una cola la inserción se hace estrictamente por un extremo de la lista, al cual podemos llamar *fondo*; la supresión sólo puede hacerse por el otro extremo de la lista, al cual llamamos *frente*. Denotaremos por Frente (C) al inicio de la cola C, y por fondo (C) al extremo final de la cola C. Para la cola C donde:

$$C = [C_1, C_2, \ldots, C_T]$$

El Frente (C) es C_1 y el Fondo (C) es C_T.

Usemos Numel (C) para denotar el número de elementos en la cola C. Numel (C) es un atributo de la cola C, y tiene un valor numérico entero. Para la cola C de arriba, Numel (C) es T.

El T-ésimo elemento de la cola C está al final de la cola; es el elemento que ha permanecido en la cola el menor tiempo. El elemento de enfrente de la cola C, es el elemento que ha permanecido en la cola el mayor tiempo.

En realidad, no es importante si trazamos una cola cuyo crecimiento vaya de derecha a izquierda, como se muestra en la figura 5-1*a*.

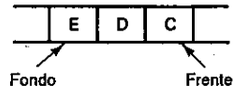


Figura 5-1a) Diagramas de una cola.

o de izquierda a derecha, como se muestra en la figura 5-1*b*

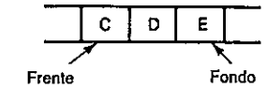


Figura 5-1b)

o, bien, en forma vertical, en tanto seamos consistentes con respecto a cuales son el frente y el fondo de la cola.

En cualquier caso, para una cola:

$$C = [C_1, C_2, \ldots, C_{Numel}]$$

diremos que C_I está *antes* que el elemento C_J, cuando $I < J$. C_I será eliminado de la cola antes que cualquier elemento que esté detrás de él. C_1 ha estado en la cola más tiempo que cualquier otro elemento que lo preceda. Esta será la forma de proceder para determinar el orden en que se atiende, por ejemplo, en una oficina de correos.

Operaciones de colas

Existen cuatro operaciones básicas válidas para los datos tipo *cola*:

1. Crear (cola),
2. Está-vacía (cola),

3. Insertar (elemento, cola),
4. Eliminar (cola).

Crear (C) produce una cola vacía cuyo nombre es C. Por definición:

Numel (crear (C)) es 0 (cero)
Frente (crear (C)) es indefinido,
Fondo (crear (C)) es indefinido.

El operador Está-vacía (C) determina si la cola está o no vacía. El operando es la cola; el resultado es un valor booleano. Está-vacía (C) es verdadero si la cola C está vacía (es decir, si Numel (C) = 0) y falso en caso contrario. Nótese que está-vacía (crear (C)) es verdadero.

Insertar (E,C) es el operador que inserta el elemento E en la cola C. Por definición, E es colocado al final de la cola, el resultado de la operación incrementa la cola.

Fondo (Insertar (E,C)) es E

Numel (C) se incrementa por la operación Insertar, y

C_{Numel} es igual a E.

El resultado de la inserción de un elemento en una cola no puede ser una cola vacía, por lo que:

Está-vacía (Insertar (E,C)) es falso.

Sólo en una situación la inserción de un elemento en la cola, afecta el frente de la cola:

Si Está-vacía (C)
entonces Frente (Insertar (E,C)) es E
si no Frente (Insertar (E,C)) es Frente (C).

Eliminar (C), saca el elemento del frente de la cola C, lo cual produce el decrecimiento de la cola. Si se debe conservar este primer elemento, se deberá tomar alguna acción antes de la operación de eliminar. Numel (C) se reduce por la operación Eliminar y el que era el segundo elemento de C se convertirá en el nuevo elemento de enfrente.

Es un error tratar de eliminar un elemento de una cola vacía:

Si Numel (C) = 0
entonces Eliminar (C) provoca una condición de error.

Nótese que:

Eliminar (Crear (C)) también produce una condición de error.

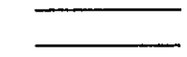
Sólo en un único caso, la eliminación de un elemento de la cola deshace la inserción previa del elemento de la cola:

Si Está-vacía (C)
entonces Eliminar (Insertar (E,C)) es C
si no Eliminar (Insertar (E,C))
= Insertar (E, Eliminar (C)).

Si C no está vacía, la inserción y eliminación actúan independientemente y no tienen efecto una sobre otra.

Ejemplo

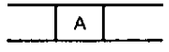
Como ejemplo de los efectos de una secuencia de operaciones en una cola, comencemos con una cola vacía llamada C. Así C = [ ] puede ser representada por la figura 5-2*a*.



Numel(C) = 0
Frente(C) indefinido
Fondo(C) indefinido

Figura 5-2a)

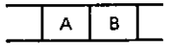
El primer elemento a insertar es A, lo que da C = [A] (Figura 5-2*b*),



Numel(C) = 1
Frente(C) = A
Fondo(C) = A

Figura 5-2b)

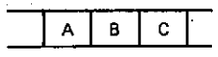
después el elemento B se inserta, así C = [A,B] (Figura 5-2*c*),



Numel(C) = 2
Frente(C) = A
Fondo(C) = B

Figura 5-2c)

ahora, el elemento C; C = [A,B,C] (Figura 5-2*d*).



Numel(C) = 3
Frente(C) = A
Fondo(C) = C

Figura 5-2d)

Si queremos eliminar un elemento de C, entonces obtenemos C = [B,C] (Figura 5-2*e*).

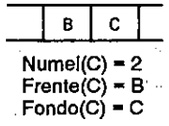


Figura 5-2e)

y después, quizás insertemos dos elementos más, es decir, D y E, así C = [B,C,D,E] (Figura 5-2*f*).

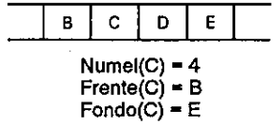


Figura 5-2f)

Finalmente si queremos suprimir un elemento de la cola, entonces C = [C,D,E] (Figura 5-2*g*).

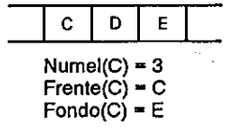


Figura 5-2g)

Como podrá observar, la cola opera de la forma *Primero que entra, primero que sale* (FIFO, del inglés First-in-first-out). Los elementos se eliminan en el orden en que fueron insertados en la cola. Puede comparar la actividad de este ejemplo de colas con la del ejemplo de pilas del capítulo anterior. Ocurre la misma secuencia de operaciones en ambos casos, sin embargo, el contenido de las listas es completamente diferente.

COLAS EN COBOL Y PASCAL

Alojamiento de colas en arreglos

Al igual que muchos lenguajes de programación que no tienen contemplada la estructura de datos de tipo pila, también carecen de la estructura de datos de tipo cola. En un programa, para la solución de un problema que utilice colas, el programador puede usar las características existentes para simular las operaciones de las colas. Una de las formas más simples para representar una cola es alojarla en un arreglo. En esta sección introducimos el método de manipulación de colas en COBOL y Pascal.

Alojar, una cola en un arreglo conlleva dos restricciones artificiales para dicha cola.

Primero, los elementos que se insertan en la cola deben ser homogéneos, porque esto es parte de la definición de un arreglo. Segundo, el programador debe especificar un límite superior sobre el número de elementos en la cola, cuando ésta se aloja en un arreglo. Una cola no tiene restricciones lógicas sobre el máximo número de elementos que puede contener. Sin embargo, una cola alojada en un arreglo queda confinada al espacio fijo reservado para el arreglo. De hecho, una cola crece y decrece en cada momento, pero el tamaño del arreglo es siempre constante.

El alojamiento de una cola en un arreglo tiene la dificultad adicional de que la integridad de la cola es responsabilidad del programador. El programador debe asegurarse de que la cola opere en forma FIFO. El compilador no asegura que las operaciones de inserción y eliminación no sean efectuadas a la mitad de la estructura.

Declaración de colas

Consideremos la declaración de una cola llamada C. Supongamos que cada elemento de C es un valor entero y que C tiene un máximo de 100 elementos. Además de la declaración del arreglo que alojará a Q, debemos declarar a las variables que apunten al frente y al fondo de la cola. FRENTE y FONDO serán variables enteras, cuyos valores serán los subíndices del primero y último elemento de la cola, respectivamente. Llamaremos al arreglo y a sus indicadores de frente y fondo como ESTRUCTURA-COLA.

En COBOL:

```
01 ESTRUCTURA-COLA.
   02 Q          OCCURS 100 TIMES PICTURE 9(5).
   02 FRENTE     PICTURE 9(3).
   02 FONDO      PICTURE 9(3).
```

En Pascal:

```
type estructura-cola =
     record cola: array [1 . . 100] of integer;
            frente,fondo: integer;
     end;
var q: estructura-cola;
```

Operaciones de colas

En un principio las variables FRENTE y FONDO son inicializadas a cero, para una cola vacía. La inserción y eliminación se efectúan a través de operadores bastante sencillos implantados en COBOL y Pascal. La principal complicación reside en la verificación de que FRENTE y FONDO no caigan fuera de los límites declarados para el arreglo que aloja a la cola. Usemos la variable NUMEL-MAX para indicar el número máximo de elementos que el arreglo puede contener; aquí NUMEL-MAX = 100. La aplicación

determina las acciones a tomar cuando ocurre una condición de desborde o subdesborde, al intentar inserciones en una cola completamente llena, o eliminaciones en una cola vacía, respectivamente.

Note que la inserción de un elemento en una cola vacía, requiere mover los apuntadores FRENTE y FONDO. Cuando se remueve un elemento de la cola y esto produce una cola vacía, a FRENTE y a FONDO se les reasigna el valor cero, de tal manera que la inserción a la cola se reinicie a partir del límite inferior del arreglo. Usemos EIN para el elemento insertado dentro de la cola y EREM para el elemento removido de la cola. Note que conservamos EREM y que suponemos que C es una variable global conocida en los ejemplos escritos en Pascal.

En COBOL:

```
INSERTAR.
        IF FONDO = NUMEL-MAX
                condición-de-desborde
ELSE COMPUTE FONDO = FONDO + 1
        MOVE EIN TO C(FONDO)
        IF FRENTE = 0
                COMPUTE FRENTE = 1.
ELIMINAR.
    IF FRENTE >0
        MOVE C (FRENTE) TO EREM
        IF FRENTE = FONDO
                COMPUTE FRENTE = 0
                COMPUTE FONDO = 0
        ELSE COMPUTE FRENTE = FRENTE + 1
    ELSE condición de subdesborde.
```

En Pascal:

```
procedure insertar (ein:integer);
begin if(c.fondo < numel - máx)
        then begin c.fondo := c.fondo atrás + 1;
                c.cola[c.fondo] : = ein;
                if(c.frente = 0)
                then c.frente := 1
            end
        else CONDICION-DESBORDE
end ;
procedure eliminar(var erem : integer)
begin if(c.frente > 0)
        then begin erem : = c.cola[c.frente];
                if(c.frente = q.fondo)
                then begin c.frente : = 0;
                        c.fondo : = 0
                    end
                else c.frente : = c.frente + 1
            end
        else CONDICION-SUBDESBORDE
end;
```


El número de elementos en la cola se puede calcular en todo momento a partir de los valores de los apuntadores Fondo y Frente.

Si FRENTE = 0
entonces Numel (C) es 0
Si no Numel (C) es FRENTE–FONDO + 1

Desplazamiento a través del almacenamiento

Examinemos ahora, con más cuidado, estos algoritmos y la utilización de un arreglo para alojar una cola. Considere inserciones y eliminaciones de elementos sucesivamente; por ejemplo, inserte cuatro elementos (Figura 5-3*a*),

Figura 5-3a)

elimina dos elementos (Figura 5-3*b*)

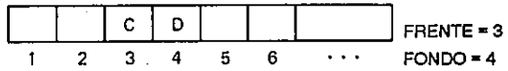


Figura 5-3b)

inserta tres elementos más (Figura 5-3*c*)

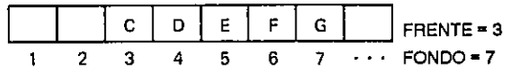


Figura 5-3c)

elimina dos elementos más (Figura 5-3*d*)

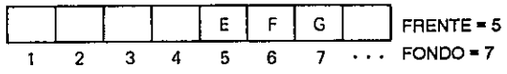


Figura 5-3d)

y así sucesivamente. La cola se recorre de izquierda a derecha a través del arreglo. ¿Qué sucede si FONDO = 100 y necesitamos insertar más elementos en la cola? Habremos alcanzado el límite del arreglo. La cola está llena sólo si FRENTE = 1 y se ha alcanzado el máximo, es decir, Numel (C) = 100. Sin embargo, es probable que el frente de la cola

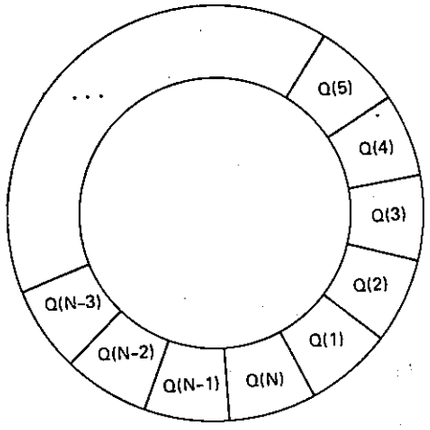


Figura 5-4 Cola circular.

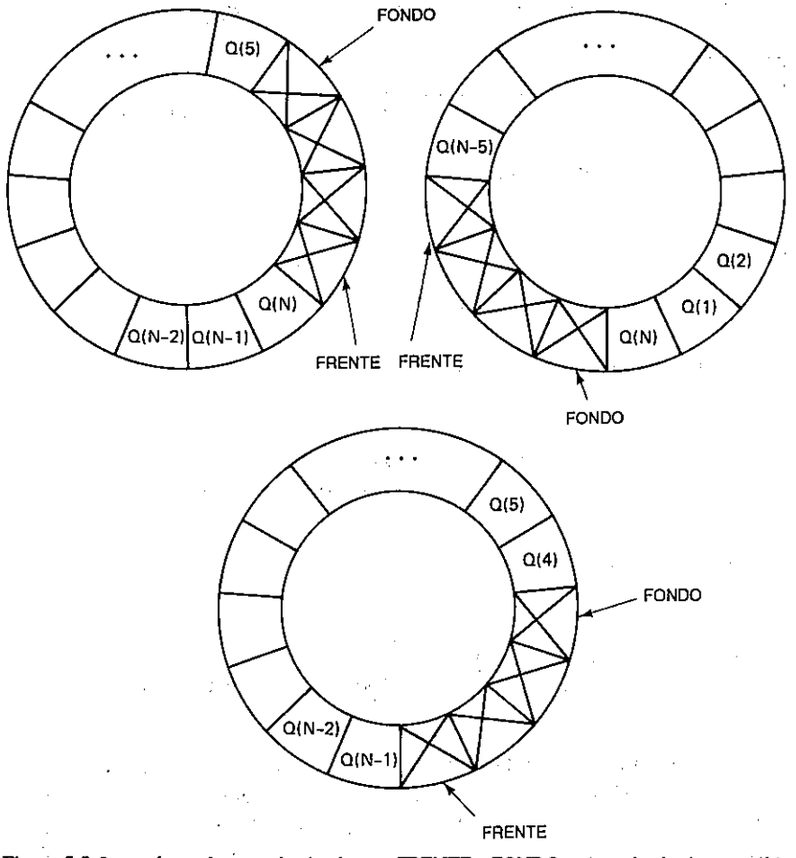


Figura 5-5 Instantáneas de una cola circular con FRENTE y FONDO apuntando al primero y último elementos, respectivamente.

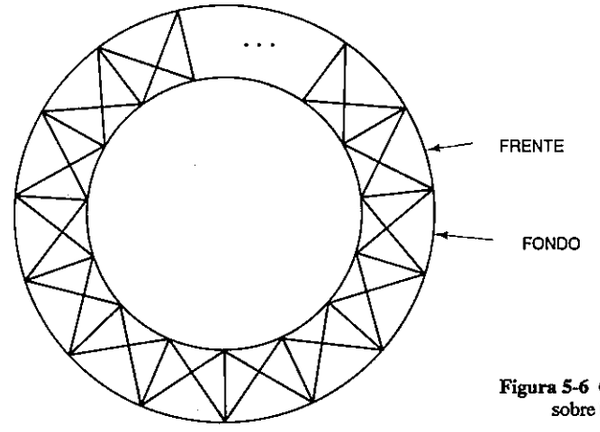


Figura 5-6 Cola llena, usando las convenciones sobre los apuntadores de la figura 5-5.

haya emigrado "a la derecha", debido a la remoción de uno o más elementos y FRENTE será mayor que 1. Puede suceder el caso de que FRENTE = 100 y que haya sólo un elemento en C, aunque aparentemente C esté llena.

Una forma de resolver este problema es alojar la cola en un arreglo mucho más grande. Esta no es una solución muy atractiva, a menos que se tenga espacio de memoria infinitamente grande que pueda ser dedicado al arreglo, o se pueda aprovechar la ventaja del conocimiento *a priori* del comportamiento futuro de la cola.

COLAS CIRCULARES

Una mejor solución es convertir a C en una cola circular, considerando que C(1) está a continuación de C(NUMEL-MAX), como en la figura 5-4. (En lo sucesivo, substitui-

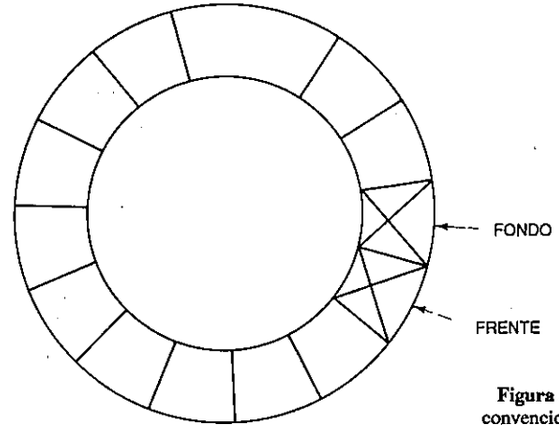


Figura 5-7 Cola con dos elementos, usando las convenciones sobre los apuntadores de la figura 5-5.

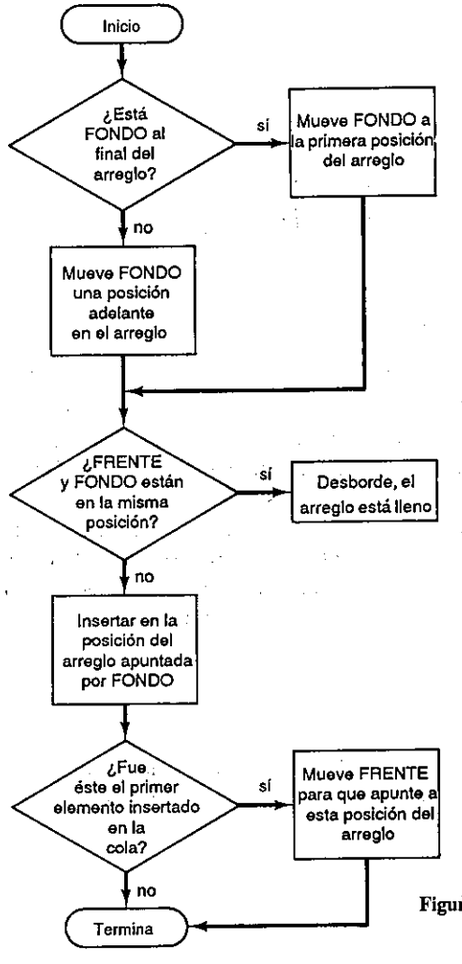


Figura 5-8 Lógica para la inserción de un elemento en una cola circular.

remos la variable NUMEL-MAX por la variable N, para representar los diagramas más fácilmente.) Ahora, dependiendo de cómo se implanten las operaciones de inserción y eliminación, la cola usará arreglos circulares de manera diferente. Aquí, presentamos dos alternativas; otras alternativas se dejan como ejercicio.

Primero, continuamos con nuestra convención de tener una variable FRENTE apuntando hacia el primer elemento de la cola y la variable FONDO hacia el último elemento. En la figura 5-5 se muestran varias configuraciones de la cola, en diversos momentos, donde X marca una localidad que está ocupada por un elemento de la cola. Note que los elementos de una cola pueden abarcar los límites entre C(1) y C(N), como se muestra en la figura 5-6. Los operadores de inserción y supresión deberán ocuparse de reinicializar los apuntadores FRENTE y FONDO, cuando se sobrepase el límite del valor N al valor 1.

La condición para que una cola esté vacía es que FRENTE = FONDO = 0. Una cola

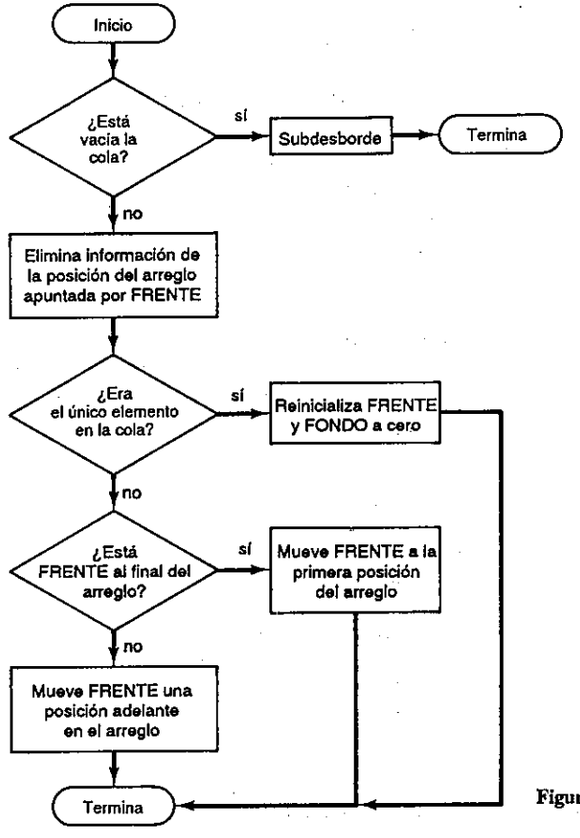


Figura 5-9 Lógica para la eliminación de un elemento de la cola circular.

llena se muestra en la figura 5-6. Note que esta cola "crece" en dirección contraria a las manecillas del reloj. ¿Cómo se ve una cola cuando contiene un solo elemento? Compare las posiciones de los apuntadores FRENTE y FONDO cuando la cola está llena y cuando contiene solamente dos elementos, como se muestra en la figura 5-7.

Uso de colas circulares

Los algoritmos para los operadores de Inserción y Eliminación en colas circulares se dan en las figuras 5-8 y 5-9, respectivamente.

Los algoritmos se pueden implantar en COBOL como sigue:

```
INSERTAR.
        IF FONDO = N
                COMPUTE FONDO = 1
        ELSE COMPUTE FONDO = FONDO + 1.
        IF FONDO = FRENTE
                Condición-de-desborde
```

```
            ELSE MOVE EIN TO C(FONDO)
                    IF FRENTE = 0
                            COMPUTE FRENTE = 1.
        ELIMINAR.
            IF FRENTE = 0
                    condición-de-subdesborde
            ELSE MOVE C(FRENTE) TO EREM
                    IF FRENTE = FONDO
                            COMPUTE FRENTE = 0
                            COMPUTE FONDO = 0
                    ELSE IF FRENTE = N
                                    COMPUTE FRENTE = 1
                            ELSE COMPUTE FRENTE = FRENTE + 1.
```

En Pascal:

```
procedure insertar(ein:integer);
begin if(c.fondo = n)
        then c.fondo : = 1
        else c.fondo : = c.fondo + 1;
        if(c.fondo = c.frente)
        then CONDICION-DESBORDE
        else begin c.cola[c.fondo] : = ein ;
                if(c.frente = 0)
                then c.frente : = 1
        end;
end;

procedure eliminar(var erem : integer) ;
begin if(c.frente = 0)
        then CONDICION-SUBDESBORDE
        else begin erem : = c.cola (c.frente).
                if (c.frente = c.fondo)
                then begin c.frente : = 0;
                            c.fondo : = 0
                    end
                else if (c.frente = n)
                    then c.frente : = 1
                    else c.frente : = c.frente + 1
            end;
end;
```

Variaciones

Si usamos los mismos algoritmos pero numeramos las localidades en el sentido de las manecillas del reloj, en lugar del sentido contrario, parecerá que la cola se mueve en sentido de las manecillas del reloj. Si mantenemos el esquema de numeración en el sentido contrario al de las manecillas, pero cambiamos los códigos de inserción y

eliminación, de tal manera que los apuntadores FONDO y FRENTE sean decrementados, en lugar de incrementados, la cola parecerá moverse en el sentido de las manecillas del reloj.

Algunos programadores prefieren numerar las localidades del arreglo de 0 a N - 1, en lugar de 1 a N (Figura 5-10). Así, se puede manipular la transición sobre el límite C(N - 1) - C(0), utilizando el módulo aritmético, el cual es una función predefinida de los lenguajes Pascal, FORTRAN y PL/1. En Pascal, A mod B es el residuo entero que resulta cuando A se divide entre B. En Pascal I := A mod B es equivalente en COBOL a:

```
DIVIDE A BY B GIVING C REMAINDER I.
```

Las operaciones de inserción y eliminación se pueden implantar en programas escritos en lenguaje Pascal como sigue. Reemplace el código:

```
if(c.fondo = n)
then c.fondo = 1
else c.fondo := c.fondo + 1
```

por:

```
c.fondo : = (c.fondo + 1)mod n;
```

en el procedimiento de inserción. Reemplace el código:

```
if(c.frente = n)
then c.frente := 1
else c.frente := c.frente + 1
```

por:

```
c.frente := (c.frente + 1) mod n;
```

en el procedimiento de eliminación.

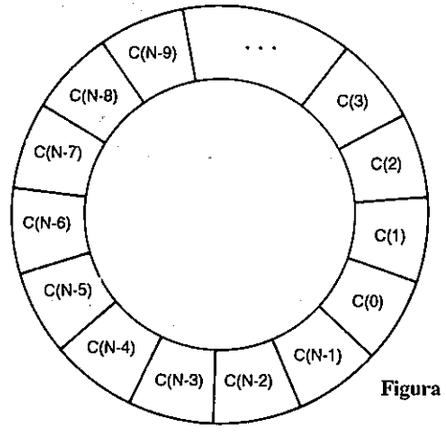


Figura 5-10 Cola circular con base en la localidad cero del arreglo.

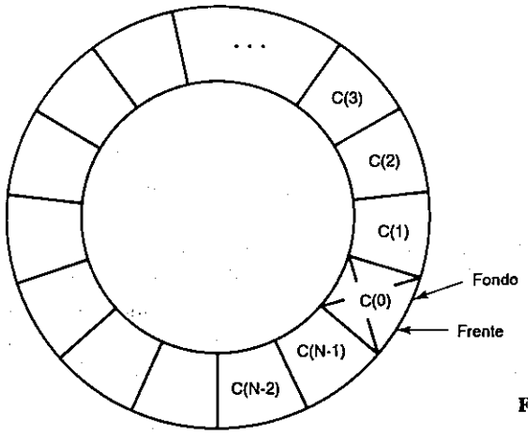


Figura 5-11 Cola circular de la figura 5-10 con FRENTE = FONDO = 0.

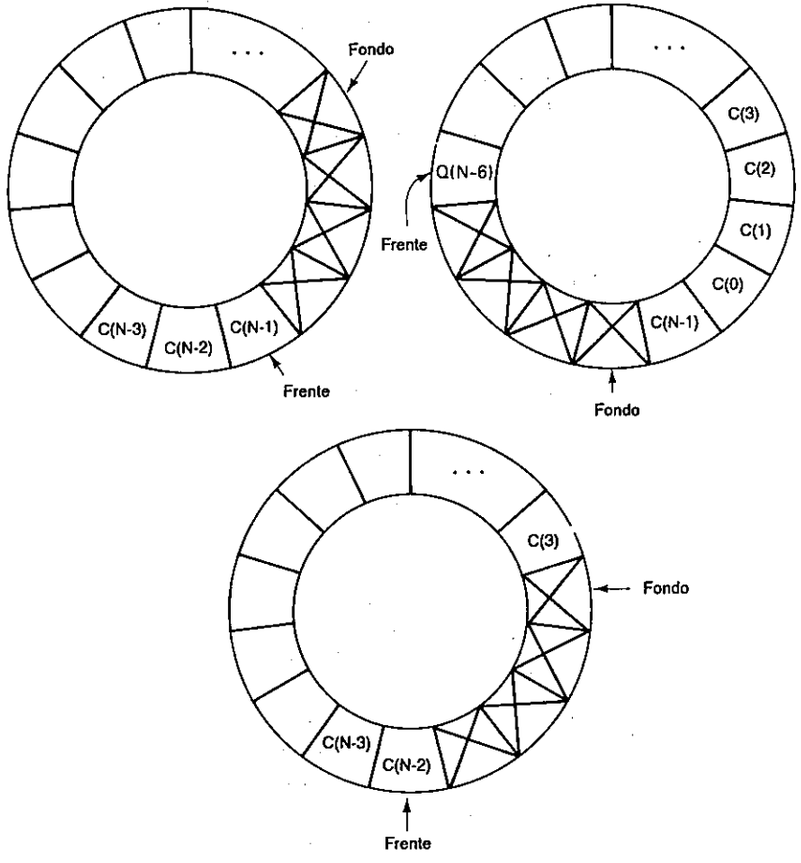


Figura 5-12 Instantáneas de cola circular con FRENTE apuntando a la localidad que precede al primer elemento y FONDO apuntando al último elemento.

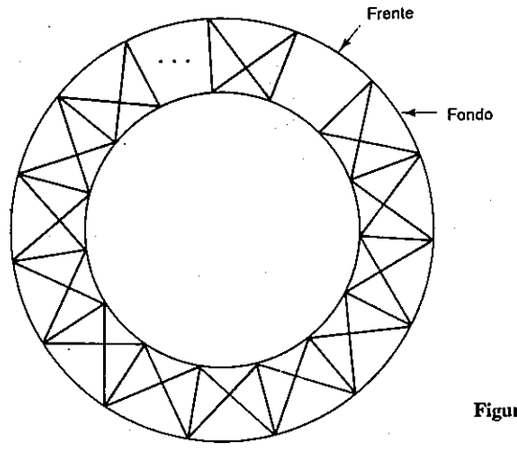


Figura 5-13 Cola llena, usando las convenciones sobre apuntadores de la figura 5-12.

En realidad, este cambio en el esquema de numeración de localidades hace un poco más difícil detectar si la cola está vacía, pues ahora, FRENTE = FONDO = 0 indican que la cola tiene un elemento en la localidad 0 (Figura 5-11). Recordemos que, cuando el arreglo de la cola está basado en 1, en lugar de cero, FRENTE = FONDO = 0 indica que la cola está vacía.

Una representación alterna

Para remediar esta dificultad es común utilizar una segunda técnica para representar una cola en un arreglo circular. En lugar de tener a FRENTE apuntando *al* elemento de enfrente de la cola, usemos a FRENTE para apuntar a la localidad que precede a la que contiene el elemento de enfrente de la cola. Se muestran ejemplos en la figura 5-12. Ahora la cola está vacía cuando FRENTE = FONDO, y está llena cuando FONDO = (FRENTE + 1) mod N (Figura 5-13). El número máximo de elementos que puede contener esta cola es N - 1, aun cuando hay N localidades. Si permitiéramos que entraran N elementos a la cola, entonces la condición de "llena" FRENTE = FONDO sería la misma que la condición de "vacía". Por supuesto, esta paradoja podría manipularse reconociendo cuando FRENTE = FONDO fuese el resultado de una inserción (la cola está llena) o eliminación (la cola está vacía), o simplemente manteniendo el valor de Numel(C).

Las operaciones de inserción y eliminación para una cola circular en Pascal, con índices de 0 a n - 1, y frente apuntando al primer elemento se muestran a continuación:

```
procedure insertar ( ein:integer);
begin if(c.frente <>(c.fondo + 1 ) mod n)
        then begin c.fondo := (c.fondo + 1) mod n;
                   c.cola[c.fondo] := ein
             end
```

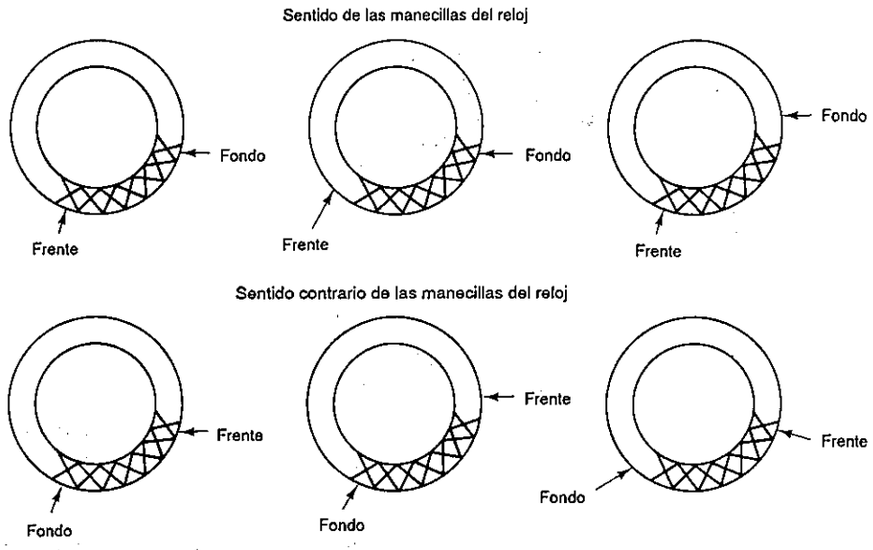


Figura 5-14 Convenciones alternas para representar una cola en un arreglo circular.

```
            else CONDICION-DESBORDE
end;
procedure eliminar(var erem : integer);
begin if(c.fondo = c.frente)
            then CONDICION-SUBDESBORDE
            else begin c.frente : = (c.frente + 1) mod n;
                        erem: = c.cola[c.frente]
                 end;
end;
```

Debe comparar estos algoritmos con los algoritmos presentados anteriormente, donde frente apunta al primer elemento de la cola.

Una representación alternativa podría tener a FONDO apuntando a la localidad anterior a la última de la cola, con FRENTE apuntando al primer elemento. Desarrolle los algoritmos para inserción y eliminación para este método.

Resumen

Hemos propuesto al menos seis formas para representar una cola en un arreglo circular (Figura 5-14). El modo de representación se determina por la manera de manejar los apuntadores en los algoritmos de inserción y eliminación. En todos los casos es posible detectar cuando la cola está vacía o llena. Los algoritmos para manipular estas alternativas tienen esencialmente la misma complejidad. Hemos presentado una variedad de técnicas, porque *todas* son muy utilizadas; para dar mantenimiento, el programador puede tener que utilizar alguna o todas las técnicas.

Hay otras formas de representar una cola en lugar de alojarla en un arreglo. Discutiremos estas representaciones más adelante, principalmente en el siguiente capítulo de listas ligadas.

COMPORTAMIENTO DE COLAS

Las colas son muy usadas en aplicaciones que simulan el comportamiento de sistemas reales en los cuales hay actividad de tipo FIFO. Las colas se constituyen en muchas situaciones comunes: colas en tráfico vehicular, esperando la luz de siga del semáforo; patrones de espera en oficinas de correo; colas formadas por clientes esperando pagar su cuenta en un almacén; colas de mensajes en redes de telecomunicación; archivos haciendo cola para salir en orden FIFO por la impresora; y muchas otras.

Parámetros de comportamiento

Los diseñadores con frecuencia se interesan en mejorar la calidad de tales sistemas, donde el comportamiento puede medirse de acuerdo a los siguientes parámetros.

- Número promedio de elementos en la cola.
- Tiempo promedio en la cola.
- Probabilidad de que la longitud de la cola sea mayor a un número especificado.
- Probabilidad de que el tiempo en la cola sea mayor de cierta cifra.
- Longitud de la cola mínima y máxima durante un periodo.
- Tiempo mínimo y máximo en la cola durante un periodo.

Existen tres métodos básicos para determinar el valor de los parámetros mencionados: observación, simulación y teoría de colas.

Observación

La observación se puede aplicar para estudiar un sistema existente. Por ejemplo, puede haber visto contadores electrónicos de automóviles que esperan la luz verde en la intersección de calles. La observación no es apropiada para estudiar sistemas que se encuentran todavía en la fase de diseño o para estudiar el efecto de cambios propuestos a sistemas existentes.

Simulación

La simulación es una herramienta poderosa para estudiar el comportamiento de sistemas. La simulación pone a prueba un modelo del sistema bajo estudio. Por ejemplo, una simulación del sistema de cajas de cobro en un supermercado puede generar llegadas a las colas de cajas registradoras, cada llegada se puede caracterizar de acuerdo con el total de tiempo requerido para ser atendido, puede representar el promedio de servicios prestados en cada caja de cobro, etc. El simulador puede generar estadísticas durante la

operación del modelo en algún periodo. Estas estadísticas pueden caracterizar el comportamiento de las colas en proceso de desarrollo.

Para desarrollar un modelo simulador, es necesario tener estadísticas substanciales disponibles para caracterizar el sistema bajo estudio. Estas estadísticas incluyen la distribución de llegadas, los promedios y desviaciones estándar de los tiempos de servicio, además de otros resultados. Estos datos pueden recolectarse a través de observaciones o se pueden plantear hipotéticamente en el "papel" para un sistema que aún no existe.

Una vez que el modelo de simulación se ha desarrollado, se pueden ajustar y experimentar sus parámetros con suma facilidad. Los efectos de cambios al sistema pueden investigarse. Por ejemplo, ¿qué sucede si agregamos al sistema otra caja de cobro? ¿Qué sucede si tenemos dos líneas rápidas de cajas de cobro en lugar de una? ¿Qué sucede si disminuimos un poco la velocidad del servicio, digamos al solicitar que todas las cuentas de cobro sean supervisadas por el administrador de la tienda?

Teoría de colas

La teoría de colas es un enfoque analítico para el estudio del comportamiento de sistemas que involucran colas. Se han desarrollado un conjunto de ecuaciones y curvas con bases matemáticas sólidas, para estimar el tamaño de las colas, tiempos de espera, probabilidades de demora, y muchos otros parámetros. Como en el caso de la simulación, se establecen suposiciones sobre las características del sistema en estudio, para aplicar los resultados de la teoría de colas. Por ejemplo, se debe establecer el porcentaje de llegadas a la cola; el número de colas y el número de servidores (quienes dan salida a los elementos de las colas); se debe caracterizar la disciplina de servicio (es decir, atender hasta completar el servicio —como en un supermercado— o bien atender un elemento durante cierto tiempo y después ponerlo al final de la cola, para continuar atendiéndolo después —como en los sistemas de tiempo compartido de computadoras); y seleccionar la distribución de demandas de servicio.

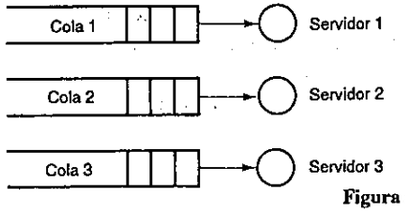


Figura 5-15 Tres servidores, cada uno con su propia cola.

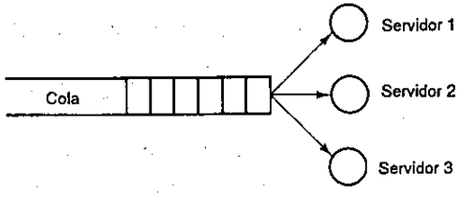


Figura 5-16 Tres servidores con una cola compartida.

Uno de los resultados interesantes de la teoría de colas, muestra que la configuración de colas para múltiples servidores, cada uno con su propia cola (Figura 5-15) tiene un promedio de tiempo de espera mayor que el tiempo requerido por el sistema con una sola cola (Figura 5-16), donde el elemento al frente de la cola es atendido por cualquiera de los servidores que se desocupe primero. Tal vez haya visto este tipo de administración de colas implantado en la oficina de correos o en el banco. ¿Por qué este tipo de atención por lo general no se usa en supermercados?

RESUMEN

Una *cola* es un caso especial de una lista lineal en la cual las inserciones son restringidas a ocurrir sólo por un extremo de la cola, el cual se identifica como el final o fondo de la cola, y las eliminaciones pueden ocurrir sólo por el otro extremo de la cola, el cual se identifica como frente de la cola.

Las colas se usan ampliamente en operaciones de computadora; por ejemplo, en la selección del siguiente trabajo a atender, en la selección del siguiente archivo a imprimir en la impresora, en la selección del siguiente paquete a procesar en una línea de teleproceso. Las colas son muy utilizadas para diversos propósitos, como en modelos de simulación de tráfico, en almacenes comerciales, oficinas de correo, peluquerías, y muchos otros lugares. Una cola se puede usar en la solución de cualquier problema que necesita estructuras de datos de tipo primero que entra-primero que sale. Los inventarios se comportan a menudo de esta manera.

Las colas pueden representarse en las memorias en forma similar que las pilas alojadas en arreglos. Cuando las colas se alojan en arreglos, la integridad de la cola debe ser responsabilidad del programador, que debe asegurar que la inserción y eliminación ocurra sólo en los extremos final y frontal, respectivamente.

Debido a que las colas se extienden a lo largo de la memoria por las inserciones y eliminaciones sucesivas, utilizan por lo general arreglos circulares. Los algoritmos para manipular estas colas son aparentemente sencillos; el lector deberá tener cuidado de comprender bien la forma de operar de estos algoritmos.

TERMINOLOGIA

Arreglos	Lista lineal
Cola	Pila
Cola circular	Primero en entrar primero en salir (FIFO)
Fondo	Simulación
Frente	Teoría de colas

EJERCICIOS DE REPASO

1. Describa las similitudes y diferencias entre colas y pilas.
2. ¿Por cuál extremo se efectúan inserciones a la cola y por cuál eliminaciones?

3. ¿Por qué se prefiere que el alojamiento de las colas sea en arreglos circulares en lugar de arreglos lineales?
4. ¿Puede ser circular una pila? ¿Por qué sí o por qué no?
5. Al observar el algoritmo de inserción a la cola, ¿cómo podemos identificar cuándo se mueve la cola en el sentido de las manecillas del reloj, o en sentido contrario?
6. Cuándo es conveniente usar la función *mod* en colas circulares?
7. ¿Cómo se prueba si la cola está vacía? Describa su colocación del apuntador del frente.
8. Escriba un procedimiento para agregar un elemento a una cola circular cuando el primer elemento es designado FRENTE y cuando FONDO indica el espacio inmediatamente posterior al final de la cola.
9. Escriba un algoritmo para contar el número de elementos en una cola circular.
10. Escriba un algoritmo para eliminar un elemento de la cola circular cuando FONDO apunta al final de la cola y FRENTE apunta al espacio anterior al frente de la cola.
11. Escriba un algoritmo para agregar un elemento a la cola circular cuando FONDO apunta al final de la cola y FRENTE apunta al frente.
12. Dibuje una cola con 10 elementos, indicando cuál es el frente y cuál es el fondo. Escriba un algoritmo para verificar cuando la cola está llena.
13. Escriba un algoritmo para poner dos colas de longitud variable (pero que en total puedan contener N elementos) en un arreglo de N localidades.
14. Una unidad central de procesamiento (CPU) concede dos unidades de tiempo de procesamiento como máximo a cada una de las transacciones; si la tarea se completa, entonces se envía a una salida hacia un disco o una terminal y, si no, entonces se coloca en el extremo final de la cola para atenderlo después. Dado que todas las tareas son enviadas a cola para la CPU y que existen 2 prioridades —prioridad 1, la cual da preferencia, y prioridad 2, la cual se subordina a la prioridad 1 — y dadas las siguientes transacciones:

Trans	Prioridad	Tiempo de entrada	Servicio requerido
A	1	0	1
B	2	0	2
C	2	1	6
D	1	2	4
E	1	2	3
F	2	3	2
G	1	5	1

Describa la historia de las colas aplicadas y el procesamiento de la CPU en intervalos de una unidad, hasta que todo el procesamiento se haya completado.

15. a) Hay un banco con dos cajeros. Uno con experiencia y el otro sin experiencia. El tiempo promedio de transacción para el nuevo cajero es el doble que el del cajero experimentado. Escriba un algoritmo para dirigir a los clientes al cajero apropiado, de tal forma que el tiempo promedio de espera sea igual en ambas colas.
    b) Ahora, suponga que el nuevo cajero "aprende" después de cada transacción y, su tiempo de transacción disminuye $\frac{1}{n}$ (donde n = tiempo de transacción para el cajero experimentado) después de cada transacción. Escriba un algoritmo para equilibrar los tiempos de espera en cola ante esta situación.

capítulo seis

listas ligadas

En este capítulo expondremos el uso de las listas ligadas para representar estructuras lineales de datos: listas lineales en general, pilas y colas. Después enfocaremos el uso de listas ligadas para representar estructuras no lineales de datos, como: árboles y gráficas.

REPRESENTACION DE LISTAS LIGADAS

Problemas con la representación secuencial

En el capítulo anterior representamos pilas y colas alojándolas en arreglos. Una restricción básica impuesta por este modo de representación es que la cantidad reservada de almacenamiento para la pila o la cola es fija. El espacio se reserva sin tomar en cuenta si la cola o la pila están vacías y existe la posibilidad de desborde si éstas crecen más de lo esperado. Los mismos problemas se presentan cuando se utiliza un arreglo para alojar una lista lineal general.

Sin embargo, existe una restricción todavía más severa, impuesta en el caso general. ¿Qué sucede cuando el elemento 43, digamos "IGOR", se elimina de una lista lineal de 2492 nombres? Los elementos del 1 al 42 no son afectados, pero los elementos 44, 'IVAN', hasta el 2492 'ZELDA', lógicamente deben recorrerse una posición en el arreglo; el elemento 44, ahora debe ser el elemento 43, el 45 ocupar el lugar 44, . . . , y el 2492 pasar a la localidad 2491. Ahora ¿qué sucede cuando un nuevo elemento digamos "HORACIO", se deba insertar después del elemento 41, "HOMERO"? Los elementos 42, "HORACLES", hasta el 2491 "ZELDA" se deben recorrer una posición a la derecha