3.3 COLAS

Una **cola** constituye una estructura lineal de datos en la que los nuevos elementos se introducen por un extremo y los ya existentes se eliminan por el otro. Es importante señalar que los componentes de la cola se eliminan en el mismo orden en el cual se insertaron. Es decir, el primer elemento que se introduce en la estructura será el que se eliminará en primer orden. Debido a esta característica, las colas también reciben el nombre de estructuras **FIFO** (*First-In, First-Out*: el primero en entrar es el primero en salir).

FIGURA 3.11
Ejemplos prácticos de colas.

Existen numerosos casos de la vida real en los cuales se usa este concepto. Por ejemplo, la cola de los bancos en las que los clientes esperan para ser atendidos —la primera persona de la cola será la primera en recibir el servicio—, la cola de los niños que esperan a veces pacientemente para subir a un juego mecánico, las colas de los vehículos esperando la luz verde del semáforo, las colas para entrar a un cine, teatro o estadio de fútbol, etcétera.

3.3.1 Representación de colas

Las colas, al igual que las pilas, no existen como estructuras de datos estándar en los lenguajes de programación. Este tipo de estructura de datos se puede representar mediante el uso de:

- Arreglos
- Listas

Al igual que en el caso de las pilas, en este libro se utilizarán arreglos para mostrar su funcionamiento. Sin embargo, la implementación mediante listas es incluso más sencilla. El lector puede implementar los algoritmos necesarios para colas, después de estudiar el capítulo que se dedica a la estructura lineal de datos.

Cuando se implementan con arreglos unidimensionales, es importante definir un tamaño máximo para la cola y dos variables auxiliares. Una de ellas para que almacene la posición del primer elemento de la cola —FRENTE— y otra para que guarde la posición del último elemento de la cola —FINAL—. En la figura 3.12 se muestra la representación de una cola en la cual se han insertado tres elementos: 111, 222 y 333, en ese orden.

El elemento 111 está en el FRENTE ya que fue el primero que se insertó; mientras que el elemento 333, que fue el último en entrar, está en el FINAL de la cola.

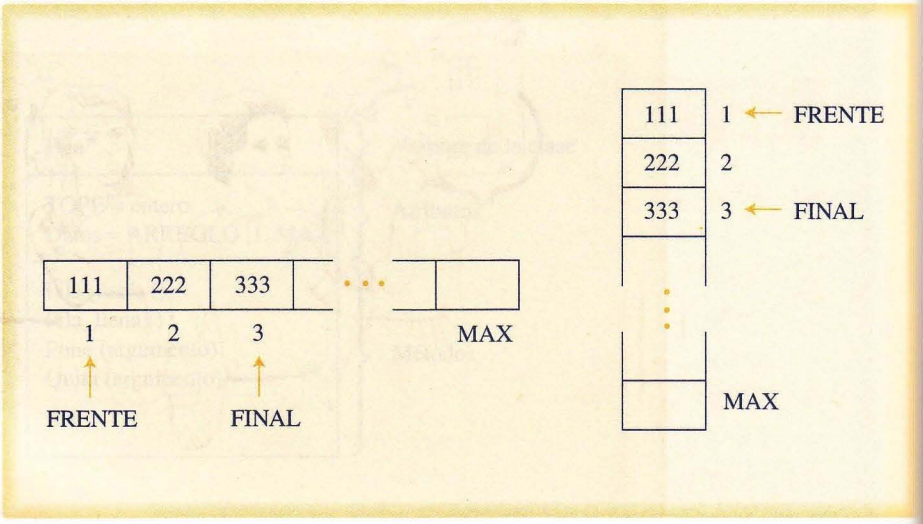


FIGURA 3.12
Representación de colas.

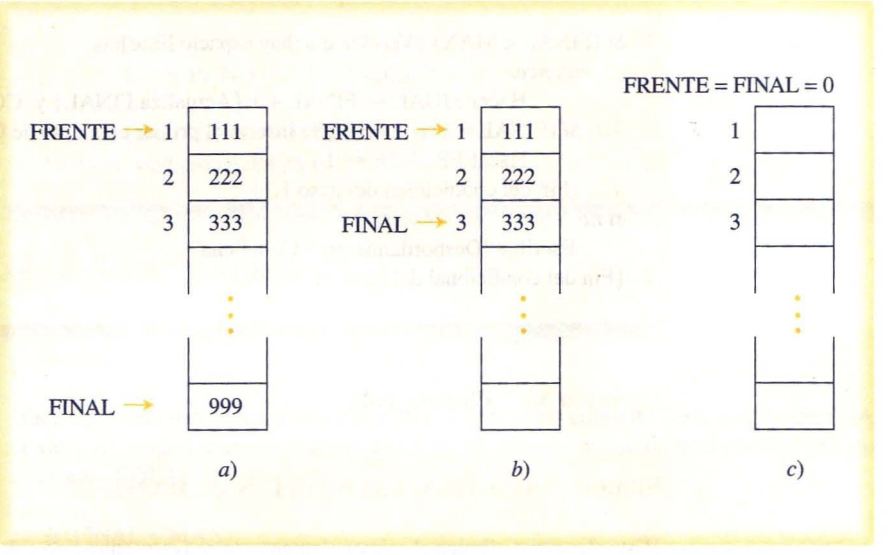


FIGURA 3.13
Representación de colas, *a)* Cola llena. *b)* Cola con algunos elementos. *c)* Cola vacía.

En la figura 3.13, por otra parte, se presentan ejemplos de: *a)* cola llena, *b)* cola con algunos elementos y *c)* cola vacía.

3.3.2 Operaciones con colas

La definición de la estructura de datos tipo cola queda completa al incluir las operaciones que se pueden realizar en ella. Las operaciones básicas que pueden efectuarse son:

- Insertar un elemento en la cola
- Eliminar un elemento de la cola

Las inserciones se llevarán a cabo por el FINAL de la cola, mientras que las eliminaciones se harán por el FRENTE —recuerde que el primero en entrar es el primero en salir—.

Considerando que una cola puede almacenar un máximo número de elementos y que además FRENTE indica la posición del primer elemento y FINAL la posición del último, se presentan a continuación los algoritmos correspondientes a las operaciones mencionadas.

Algoritmo 3.7 Inserta_cola

Inserta_cola (COLA, MAX, FRENTE, FINAL, DATO)

{Este algoritmo inserta el elemento DATO al final de una estructura tipo cola. FRENTE y FINAL son los punteros que indican, respectivamente, el inicio y fin de COLA. La primera vez FRENTE y FINAL tienen el valor 0, ya que la cola está vacía. MAX es el máximo número de elementos que puede almacenar la cola}

```
1. Si (FINAL < MAX) {Verifica que hay espacio libre}
     entonces
          Hacer FINAL ← FINAL + 1 {Actualiza FINAL} y COLA[FINAL] ← DATO
   1.1 Si (FINAL = 1) entonces {Se insertó el primer elemento de COLA}
          Hacer FRENTE ← 1
   1.2 {Fin del condicional del paso 1.1}
   si no
       Escribir "Desbordamiento – Cola llena"
2. {Fin del condicional del paso 1}
```

Algoritmo 3.8 Elimina_cola

```
Elimina_cola (COLA, FRENTE, FINAL, DATO)

{Este algoritmo elimina el primer elemento de una estructura tipo cola y lo almacena en DATO.
FRENTE y FINAL son los punteros que indican, respectivamente, el inicio y fin de la cola}

1. Si (FRENTE ≠ 0) {Verifica que la cola no esté vacía}
     entonces
          Hacer DATO ← COLA [FRENTE]
   1.1 Si (FRENTE = FINAL) {Si hay un solo elemento}
          entonces
               Hacer FRENTE ← 0 y FINAL ← 0 {Indica COLA vacía}
          si no
               Hacer FRENTE ← FRENTE + 1
   1.2 {Fin del condicional del paso 1.1}
          si no
       Escribir "Subdesbordamiento – Cola vacía"
2. {Fin del condicional del paso 1}
```

Es posible definir algoritmos auxiliares para determinar si una cola está llena o vacía. A partir de estos algoritmos se podrían reescribir los algoritmos 3.7 y 3.8.

A continuación se presentan los algoritmos que permiten verificar el estado de una cola, quedando como tarea sugerida la reescritura de los dos algoritmos anteriores.

Algoritmo 3.9 Cola_vacía

```
Cola_vacía (COLA, FRENTE, BAND)

{Este algoritmo determina si una estructura de datos tipo cola está vacía, asignando a BAND
el valor de verdad correspondiente}

1. Si (FRENTE = 0)
```

```
        entonces
            Hacer BAND ← VERDADERO
        si no
            Hacer BAND ← FALSO
2. {Fin del condicional del paso 1}
```

Algoritmo 3.10 Cola_llena

```
Cola_llena (COLA, FINAL, MAX, BAND)

{Este algoritmo determina si una estructura de datos tipo cola está llena, asignando a BAND
el valor de verdad correspondiente. MAX es el número máximo de elementos que puede
almacenar COLA}

1. Si (FINAL = MAX)
        entonces
            Hacer BAND ← VERDADERO
        si no
            Hacer BAND ← FALSO
2. {Fin del condicional del paso 1}
```

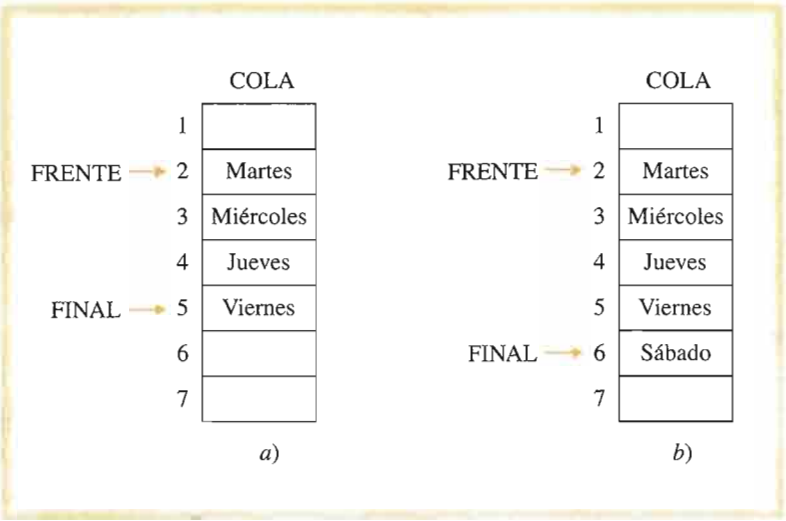
Aquí se incluye un ejemplo para ilustrar el funcionamiento de las operaciones de inserción y eliminación en colas.

Ejemplo 3.6

Retome el ejemplo 3.1 de la sección 3.1.2. Se insertan en COLA los elementos: *lunes*, *martes*, *miércoles*, *jueves* y *viernes* —en ese orden—, de modo que la estructura queda como se muestra en la figura 3.14. Para este ejemplo MAX = 7.

FIGURA 3.14
...ción y eliminación
... colas.

FIGURA 3.15
Inserción y eliminación en colas. *a)* Luego de eliminar lunes. *b)* Luego de insertar sábado.



El elemento *lunes* es el primero que se puede eliminar por ser el primero que se insertó en la cola. Luego de la eliminación, FRENTE guarda la posición del siguiente elemento (fig. 3.15*a*). Si ahora se insertara *sábado*, éste ocuparía ahora la posición siguiente al elemento *viernes* (fig. 3.15*b*).

Analice lo que ocurre en la cola, si se llevan a cabo las siguientes operaciones:

Se eliminan *martes*, *miércoles*, *jueves* y *viernes* (fig. 3.16*a*).
Se inserta *domingo* (fig. 3.16*b*).
Se elimina *sábado* (fig. 3.16*c*).

FIGURA 3.16
Inserción y eliminación en colas. *a)* Luego de eliminar martes, miércoles, jueves y viernes. *b)* Luego de insertar domingo. *c)* Luego de eliminar sábado.

Después de insertar al elemento *domingo*, ya no se pueden insertar nuevos elementos a la cola porque FINAL es igual que MAX (FINAL = MAX = 7). Sin embargo, como lo refleja la figura 3.16*c*, existe espacio disponible desperdiciado.

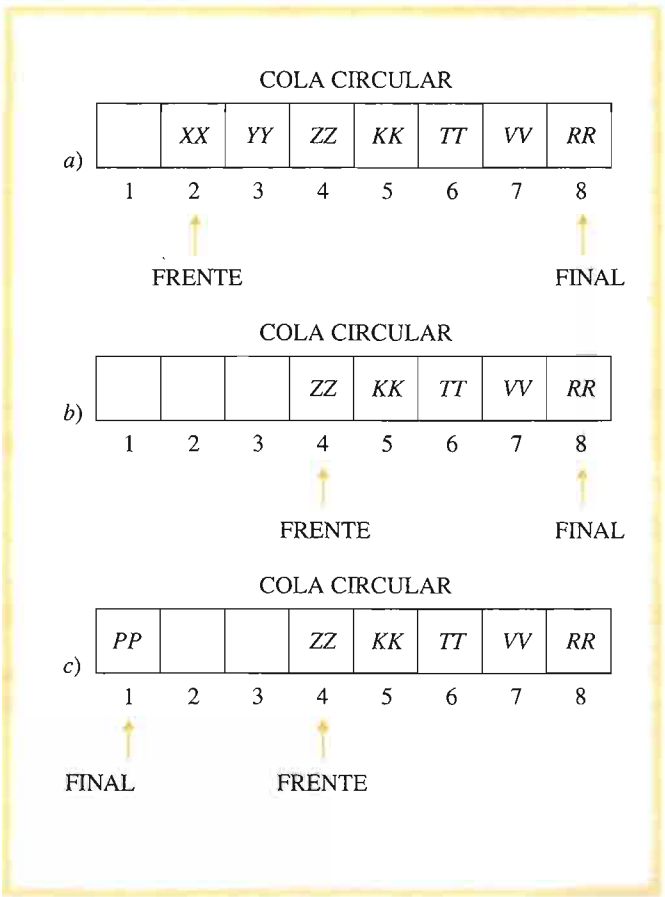
Observe que luego de insertar *domingo* se llegó a una situación conflictiva porque a pesar de que hay espacio disponible, no se pueden insertar otros elementos. Este inconveniente se puede superar perfectamente si manejamos las colas en forma circular.

3.3.3 Colas circulares

Una **cola circular** constituye una estructura de datos lineal en la cual el siguiente elemento del último en realidad es el primero. De esta forma se utiliza de manera más eficiente la memoria de la computadora. En la figura 3.17 se muestra la representación gráfica de una cola circular.

En la figura 3.17 se ilustra cómo se actualizan los punteros FRENTE y FINAL en una cola circular, a medida que se insertan o eliminan elementos. En la figura 3.17*a* la cola tiene algunos elementos (FRENTE = 2 y FINAL = 8). En la figura 3.17*b* se han eli-

FIGURA 3.17
Representación de colas circulares. *a*) Frente < Final. *b*) Frente < Final. *c*) Frente > Final.



minado de la cola dos elementos —primero se quitó *XX* y luego *YY*—, quedando FRENTE = 4. Por último, en la figura 3.17*c* se ha insertado un nuevo elemento —*PP*— en la cola. Como FINAL = MAX se llevó el apuntador a la primera posición que estaba vacía (FINAL = 1). De esta manera se logra mejor aprovechamiento del espacio de memoria ya que al eliminar un elemento la casilla correspondiente de la cola queda disponible para futuras inserciones.

A continuación se presentan los algoritmos de inserción y eliminación en colas circulares.

Algoritmo 3.11 Inserta_circular

Inserta_circular (COLACIR, MAX, FRENTE, FINAL, DATO)

{Este algoritmo inserta el elemento DATO al final de una estructura tipo cola circular —COLACIR—. FRENTE y FINAL son los punteros que indican, respectivamente, el inicio y el fin de la cola circular. MAX es el número máximo de elementos que puede almacenar COLACIR}

```
1. Si ((FINAL = MAX) y (FRENTE = 1)) o ((FINAL +1) = FRENTE)
       entonces
           Escribir "Desbordamiento – Cola llena"
       si no
   1.1 Si (FINAL = MAX)
           entonces
               Hacer FINAL ← 1
           si no
               Hacer FINAL ← FINAL + 1
   1.2 {Fin del condicional del paso 1.1}
       Hacer COLACIR[FINAL] ← DATO
   1.3 Si (FRENTE = 0) entonces
           Hacer FRENTE ← 1
   1.4 {Fin del condicional del paso 1.3}
2. {Fin del condicional del paso 1}
```

Algoritmo 3.12 Elimina_circular

Elimina_circular (COLACIR, MAX, FRENTE, FINAL, DATO)

{Este algoritmo elimina el primer elemento de una estructura tipo cola circular —COLACIR— y lo almacena en DATO. FRENTE y FINAL son los punteros que indican, respectivamente, el inicio y fin de la estructura. MAX es el tamaño de COLACIR}

```
1. Si (FRENTE = 0) {Verifica si la cola está vacía}
       entonces
           Escribir "Subdesbordamiento – Cola vacía"
```

```
    si no
    Hacer DATO ← COLACIR[FRENTE]
1.1 Si FRENTE = FINAL {Si hay sólo un elemento}
        entonces
            Hacer FRENTE ← 0 y FINAL ← 0
        si no
    1.1.1 Si (FRENTE = MAX)
            entonces
                Hacer FRENTE ← 1
            si no
                Hacer FRENTE ← FRENTE + 1
    1.1.2 {Fin del condicional del paso 1.1.1}
1.2 {Fin del condicional del paso 1.1}
2. {Fin del condicional del paso 1}
```

A continuación se presenta un ejemplo para ilustrar el funcionamiento de las operaciones de inserción y eliminación en colas circulares.

Ejemplo 3.7

En la figura 3.18*a* se presenta una estructura tipo cola circular de máximo 8 elementos (MAX = 8), en la cual ya se han almacenado algunos valores. En la figura 3.18*b* se muestra el estado de la cola luego de insertar el elemento *NN*.

Si se quisiera insertar otro elemento se presentaría un error de desbordamiento, ya que FINAL + 1 = FRENTE.

A continuación se eliminan los valores *XX*, *YY*, *ZZ*, *KK*, *TT* y *VV* en ese orden. La cola queda como se muestra en la figura 3.19.

FIGURA 3.18

Inserción y eliminación en colas circulares. *a*) Estado inicial de la cola circular. *b*) Luego de insertar *NN*.

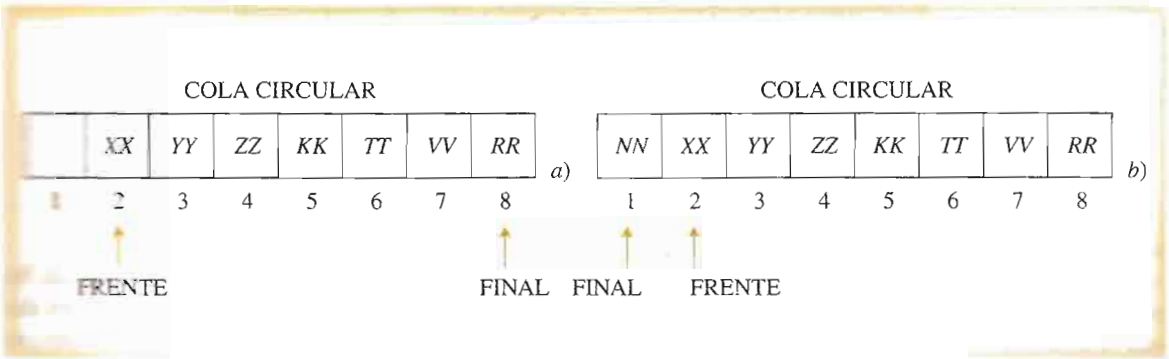
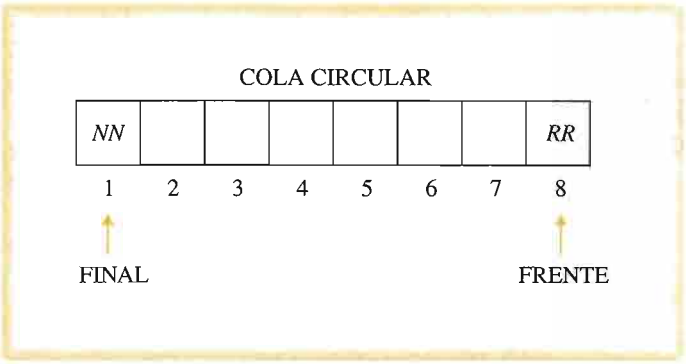
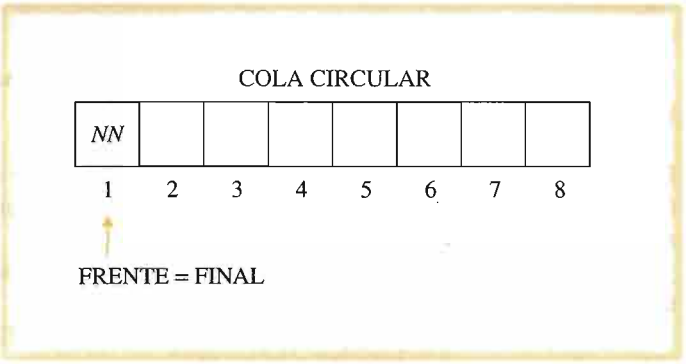


FIGURA 3.19
Inserción y eliminación en colas circulares.



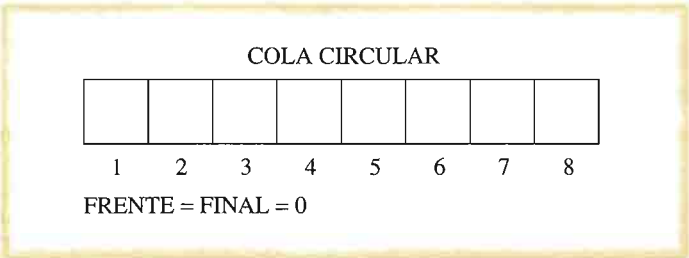
Ahora se elimina el siguiente elemento *RR*. Al ser FRENTE = MAX, se le da el valor de 1, figura 3.20.

FIGURA 3.20
Inserción y eliminación en colas circulares.



Al eliminar *NN*, como FRENTE = FINAL, es decir, sólo quedaba un elemento en la cola, actualizamos los dos punteros en cero. La cola queda vacía, figura 3.21.

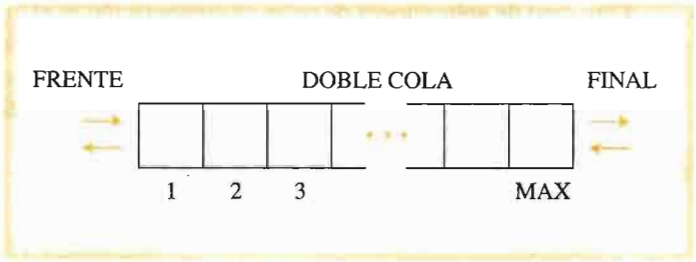
FIGURA 3.21
Inserción y eliminación en colas circulares.



3.3.4 Doble cola

Una **doble cola** o **bicola** es una generalización de una estructura de datos tipo cola. En una doble cola, los elementos se pueden insertar o eliminar por cualquiera de los dos extremos. Es decir, se pueden insertar y eliminar valores tanto por el FRENTE como por el FINAL de la cola. Una doble cola se representa como se muestra en la figura 3.22. Observe que las dos flechas en cada extremo indican que ambas operaciones son posibles.

FIGURA 3.22
Representación de doble cola



Existen dos variantes de las dobles colas:

- Doble cola con entrada restringida
- Doble cola con salida restringida

La primera de ellas permite que las eliminaciones se realicen por cualesquiera de los dos extremos, mientras que las inserciones sólo por el FINAL de la cola (fig. 3.23).

La segunda variante permite que las inserciones se realicen por cualquiera de los dos extremos, mientras que las eliminaciones sólo por el FRENTE de la cola (fig. 3.24).

FIGURA 3.23
Doble cola con entrada restringida

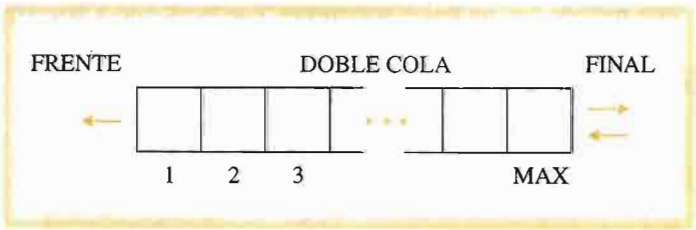
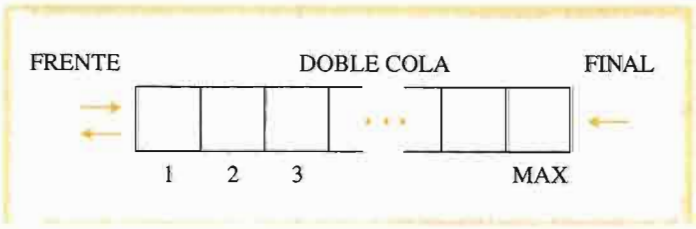


FIGURA 3.24
Doble cola con salida restringida



3.3.5 Aplicaciones de colas

El concepto de cola está ligado a computación. Una aplicación común de las colas se presenta cuando se envía a imprimir algún documento o programa en las colas de impresión. Cuando hay una sola impresora para atender a varios usuarios, suele suceder que algunos de ellos soliciten los servicios de impresión al mismo tiempo o mientras el dispositivo está ocupado. En estos casos se forma una cola con los trabajos que esperan para ser impresos; éstos se procesarán en el orden en el cual fueron introducidos en la cola.

Otro caso de aplicaciones de colas en computación es el que se presenta en los sistemas de tiempo compartido. Varios usuarios comparten ciertos recursos, como CPU, memoria de la computadora. Los recursos se asignan a los procesos que están en cola de espera, suponiendo que todos tienen una misma prioridad, en el orden en el cual fueron introducidos en la cola.

3.3.6 La clase *Cola*

La **clase *Cola*** tiene atributos y métodos, como cualquier clase. Los atributos son la colección de elementos y los punteros FRENTE y FINAL. Los métodos, por otra parte, son todas las operaciones analizadas en las secciones anteriores: Cola_vacía, Cola_llena, Inserta_cola y Elimina_cola.

En la figura 3.25 se puede observar gráficamente a la clase *Cola*. Se tiene acceso a los miembros de un objeto de la clase *Cola* por medio de la notación de puntos. Cuando se asume que la variable COOBJ es un objeto de la clase Cola, previamente creado, se puede hacer:

COOBJ.Cola_llena para invocar el método que determina si la cola está o no llena. En este método no hay argumentos, ya que todos los valores requeridos son miembros de la clase.

COOBJ.Inserta_cola(argumento) para insertar un nuevo elemento en la cola. En este método hay un único argumento, que es el dato a insertar; todos los otros valores requeridos son miembros de la clase.

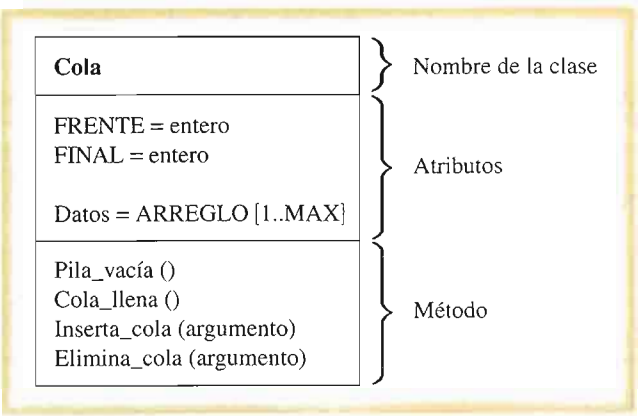


FIGURA 3.25
Clase *Cola*.