**Tipos de Colas**

# **Introducción**

Previamente en el curso habíamos visto un concepto básico de colas, en este entendimos que se trata de un tipo de estructura de datos FIFO puesto a que el primer dato ingresado también será el primer dato en salir, sin embargo, estas también se clasifican dependiendo de ciertas características, en este capítulo se nombraran algunas de ellas, junto con una breve descripción gráfica y un código representativo de estas. Se explicará de la manera más detallada posible para una mejor comprensión del lector.

1. **Colas de Prioridades**

Las colas de prioridad son estructuras de datos que gestionan elementos según su prioridad. A diferencia de una cola tradicional, donde el primer elemento en entrar es el primero en salir (FIFO), en una cola de prioridad, los elementos se eliminan de acuerdo con su nivel de prioridad. Es decir, el elemento con la mayor (o en algunos casos, menor) prioridad es el próximo en ser eliminado.

Características clave de las colas de prioridad:

* **Prioridad Asociada:**

Cada elemento de la cola de prioridad tiene asociado un valor de prioridad que determina su orden de eliminación.

* **Acceso al Elemento de Máxima Prioridad:**

La principal operación en una cola de prioridad es acceder al elemento con la máxima (o mínima) prioridad sin eliminarlo.

* **Operaciones de Inserción y Eliminación:**

Se pueden insertar nuevos elementos en la cola de prioridad, y cuando se elimina un elemento, generalmente es el que tiene la mayor (o menor) prioridad.

* **Implementación con Estructuras de Datos:**

Las colas de prioridad se pueden implementar utilizando diversas estructuras de datos, como montículos (heaps), árboles binarios de búsqueda balanceados, o arreglos con técnicas de ordenamiento específicas.

Casos de Uso:

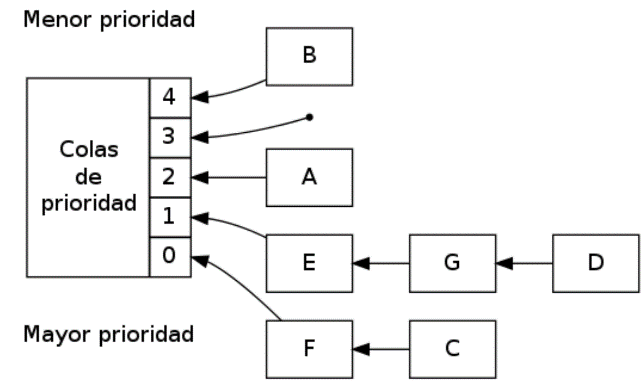
Son útiles en situaciones donde es necesario manejar elementos en función de su urgencia, importancia o algún otro criterio de prioridad. Se utilizan en algoritmos y aplicaciones donde la planificación o el procesamiento deben realizarse en un orden específico.

**Tipos de Colas de Prioridad:**

Existen colas de prioridad de máxima y de mínima. En una cola de prioridad de máxima, el elemento con la mayor prioridad se elimina primero. En una cola de prioridad de mínima, el elemento con la menor prioridad se elimina primero.

Ejemplos de aplicaciones de colas de prioridad incluyen la planificación de tareas, algoritmos de búsqueda de caminos mínimos, gestión de procesos en sistemas operativos, entre otros. La flexibilidad y eficiencia en la gestión de prioridades hacen que las colas de prioridad sean una herramienta valiosa en diversas áreas de la informática y la ingeniería de software.

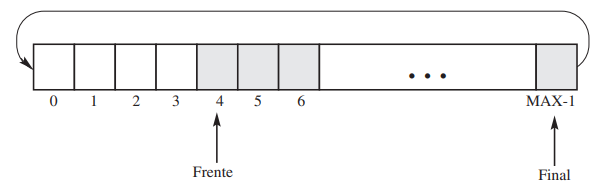
.



***Imagen 1:*** *Cola de prioridad.*

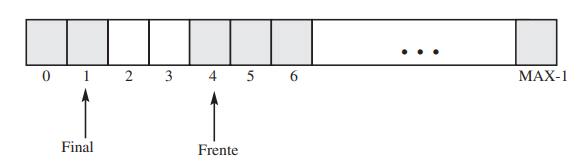
1. **Colas circulares**

Una cola circular es aquella en la cual el sucesor del último elemento es el primero. Por lo tanto, el manejo de las colas como estructuras circulares permite un mejor uso del espacio de memoria reservado para la implementación de las mismas. La Imagen 2 corresponde a la representación gráfica de una cola circular. Observe que el siguiente elemento del último es el primero. (Silvia Guardati, 2017, p 231).



**Imagen 2.** *Estructura de una cola circular.*

La Imagen 3 presenta el esquema correspondiente a una cola circular, en la cual el final se movió hacia el inicio de la cola, teniendo un valor menor al frente. En este ejemplo, (Silvia Guardati, 2017) la cola tiene las posiciones 4 a MAX-1 y 0 a 1 ocupadas, siendo el primero elemento a salir el que está en la posición 4 y el último insertado el que está en la posición 1.



**Imagen 3** *Cola circular*

Los algoritmos correspondientes a las operaciones de inserción y eliminación varían al tratarse de colas circulares, es lo referente a la actualización de los punteros. Asimismo, (Silvia Guardati, 2017) la condición para determinar si la cola está llena debe considerar todos los casos que puedan presentarse, que son:

1. El frente en la posición 0 y el final en la posición (MAX -1),0.
2. El (Final+1) es igual al Frente.

Ambos se evalúan por medio de la expresión: (Final +1) % MAX == Frente.

* **Estructura Circular:**

En una cola circular, los elementos están organizados en una estructura circular, lo que significa que el último elemento está conectado al primer elemento, formando un bucle.

* **Frente y Final:**

A diferencia de las colas lineales tradicionales, las colas circulares no tienen un "final" o una "cima" claramente definidos. El frente y el final de la cola están conectados en la estructura circular.

* **Uso Eficiente de Espacio:**

Las colas circulares pueden ser más eficientes en términos de uso de espacio en comparación con las colas lineales, ya que no hay una posición fija al final de la cola que debe permanecer vacía para permitir la expansión.

* **Desbordamiento o Subdesbordamiento:**

Al usar una implementación basada en arreglos, es importante gestionar adecuadamente el desbordamiento (cuando la cola está llena) y el subdesbordamiento (cuando la cola está vacía). Esto se logra utilizando la aritmética modular para realizar operaciones circulares.

* **Operaciones de Enqueue (Encolar) y Dequeue (Desencolar):**

Se pueden realizar operaciones de encolar y desencolar en una cola circular. Al encolar elementos, se insertan al final de la cola, y al desencolar elementos, se eliminan desde el frente.

* **Uso Común en Sistemas Embebidos y Ciclos Continuos:**

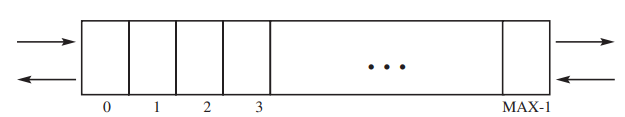
Debido a su eficiencia y estructura cíclica, las colas circulares son comunes en sistemas embebidos y situaciones donde las operaciones de encolar y desencolar son frecuentes y deben ocurrir de manera continua.

* **Aplicaciones en Algoritmos de Manejo de Datos:**

Se utilizan en algoritmos y estructuras de datos donde se necesita un acceso cíclico a los elementos, como en ciertos tipos de algoritmos de manejo de datos o en situaciones donde la cola debe mantener un estado constante de actividad.

1. **Colas dobles**

Otra variante de las estructuras tipo cola son las colas dobles. Como su nombre lo indica, estas estructuras permiten realizar las operaciones de la inserción y eliminación por cualquiera de sus extremos. (Silvia Guardati, 2017, p 232). Gráficamente una cola doble se representa de la siguiente manera Imagen 4:



**Imagen 4.** *Cola doble*

Debido a que este tipo de estructura es una generalización del tipo cola no se presentan aquí las operaciones, (Silvia Guardati, 2017). Al respecto, sólo se menciona que será necesario definir métodos que permitan insertar por el frente y por el final, así como métodos que permitan eliminar por ambos extremos. Las condiciones para determinar el estado de la cola no varían.

Por último, es importante señalar que una doble cola también puede ser circular. En dicho caso, será necesario que los métodos de la inserción y eliminación (sobre cualquier de los extremos) consideren el movimiento adecuado de los punteros. (Silvia Guardati, 2017).

* **Operaciones de Inserción y Eliminación en Ambos Extremos:**

Una característica fundamental es la capacidad de realizar operaciones de inserción (**push**) y eliminación (**pop**) tanto en el frente como en el final de la cola. Esto proporciona flexibilidad en la manipulación de los elementos.

* **Estructura Lineal:**

Aunque el término "cola doble" sugiere que tiene dos extremos, la estructura subyacente suele ser lineal. La diferencia clave es que se pueden acceder y manipular ambos extremos.

* **Acceso Eficiente a Ambos Extremos:**

A diferencia de otras estructuras de datos, como las listas enlazadas, las colas dobles permiten un acceso eficiente tanto al frente como al final. Esto es beneficioso para operaciones que involucran inserciones o eliminaciones en ambos extremos.

* **Implementación con Arreglos o Listas Enlazadas:**

Las colas dobles pueden implementarse utilizando arreglos o listas enlazadas. La elección de la implementación dependerá de los requisitos específicos de la aplicación y las operaciones que se realizarán con mayor frecuencia.

* **Casos de Uso Versátiles:**

Debido a su capacidad para realizar operaciones en ambos extremos, las colas dobles son versátiles y se utilizan en situaciones donde se requiere flexibilidad en la manipulación de elementos, como en algoritmos de procesamiento de datos en tiempo real o sistemas de gestión de eventos.

1. **Ejemplo de cola circular**

A continuación, plantearemos un ejemplo en C++ de una cola circula funcional y adjunto a esto el código del mismo.

**colasCirculares.h**

#include <iostream>

**class** **CircularQueue** {

**private:**

**int** \*array;

**int** frente, final, capacidad;

**public:**

CircularQueue(**int** size);

~CircularQueue();

**bool** **estaVacia**();

**bool** **estaLlena**();

**void** **encolar**(**int** elemento);

**void** **desencolar**();

**void** **mostrarContenido**();

};

**ColasCirculares.cpp**

#include "colasCirculares.h"

CircularQueue::CircularQueue(**int** size) {

capacidad = size + **1**; // El tamaño real es uno más para diferenciar entre frente y final.

array = **new** **int**[capacidad];

frente = final = **0**;

}

CircularQueue::~CircularQueue() {

**delete**[] array;

}

**bool** CircularQueue::estaVacia() {

**return** frente == final;

}

**bool** CircularQueue::estaLlena() {

**return** (final + **1**) % capacidad == frente;

}

**void** CircularQueue::encolar(**int** elemento) {

**if** (!estaLlena()) {

array[final] = elemento;

final = (final + **1**) % capacidad;

std::cout << "Elemento " << elemento << " encolado.**\n**";

} **else** {

std::cout << "La cola está llena. No se puede encolar.**\n**";

}

}

**void** CircularQueue::desencolar() {

**if** (!estaVacia()) {

**int** elemento = array[frente];

frente = (frente + **1**) % capacidad;

std::cout << "Elemento " << elemento << " desencolado.**\n**";

} **else** {

std::cout << "La cola está vacía. No se puede desencolar.**\n**";

}

}

**void** CircularQueue::mostrarContenido() {

**if** (!estaVacia()) {

std::cout << "Contenido de la cola: ";

**int** i = frente;

**while** (i != final) {

std::cout << array[i] << " ";

i = (i + **1**) % capacidad;

}

std::cout << "**\n**";

} **else** {

std::cout << "La cola esta vacía.**\n**";

}

}

main.cpp

#include <iostream>

#include "ColasCirculares.cpp"

**int** **main**() {

**int** tamano;

std::cout << "Ingrese el tamaño de la cola circular: ";

std::cin >> tamano;

CircularQueue colaCircular(tamano);

**int** opcion;

**do** {

std::cout << "**\n**\*\*\* Menu de Operaciones \*\*\***\n**";

std::cout << "1. Encolar**\n**";

std::cout << "2. Desencolar**\n**";

std::cout << "3. Mostrar Contenido**\n**";

std::cout << "0. Salir**\n**";

std::cout << "Ingrese su opcion: ";

std::cin >> opcion;

**switch** (opcion) {

**case** **1**:

**int** elemento;

std::cout << "Ingrese el elemento a encolar: ";

std::cin >> elemento;

colaCircular.encolar(elemento);

**break**;

**case** **2**:

colaCircular.desencolar();

**break**;

**case** **3**:

colaCircular.mostrarContenido();

**break**;

**case** **0**:

std::cout << "Saliendo del programa.**\n**";

**break**;

**default:**

std::cout << "Opción no válida. Inténtelo de nuevo.**\n**";

**break**;

}

} **while** (opcion != **0**);

**return** **0**;

}

1. **Conclusiones**

* Las Colas Tradicionales Siguen el principio FIFO (Primero en entrar, primero en salir) y son útiles en situaciones donde se requiere un orden estricto de procesamiento.
* Las Colas de Prioridad: Permiten gestionar elementos según su prioridad, facilitando la atención de los elementos más importantes antes que los menos importantes.
* Las Colas Dobles: Proporcionan flexibilidad al permitir operaciones de inserción y eliminación tanto al frente como al final, siendo ideales para escenarios donde se necesita acceso eficiente a ambos extremos.
* Las Colas Circulares Presentan una estructura cíclica que permite un uso eficiente del espacio y son ideales para aplicaciones que requieren operaciones continuas de encolar y desencolar, como en sistemas embebidos o situaciones con ciclos de operación repetitivos.

**Referencias**

* Guardati, S. G. (2017). ESTRUCTURA DE DATOS ORIENTADA A OBJETOS Algoritmos con C+ 1ra, PEARSON Educación, Castillo, Bernardino Gutiérrez Hernández Supervisor de producción: Rodrigo Romero Villalobo.
* Cueva, V, Gonzáles Luis (2020). Estructura de datos, algoritmos fundamentales 1ra, Editorial Digital tecnológico de Monterrey
* Hernández, M, Baquero, L. (2022). ESTRUCTURA DE DATOS, fundamentación práctica 1ra, edU Ra-Ma, Sánchez, Israel Sebastián Técnico editorial.