

# Práctica 18: Colas en C++

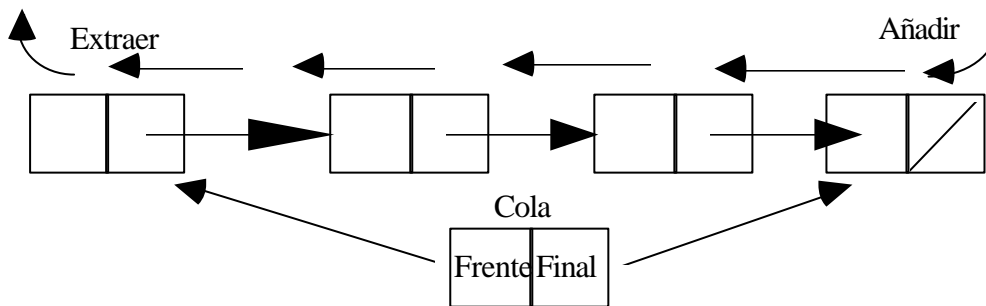
## Teoría.

En muchas aplicaciones se cumple el hecho de que el primer proceso que solicita un servicio o recurso es el primero en ser servido. Para retirar elementos (servicios, recursos) en el mismo orden en que fueron solicitados, se necesitan unas estructuras de datos abstractas que mantengan una secuencia de valores y permitan añadir nuevos elementos por un extremo y retirarlos por el otro. Esta estructura recibe el nombre de **cola**.

En las colas el elemento que entró el primero sale también el primero, por ello se le conoce también como listas **FIFO** (*first-in, first-out*; primero en entrar, primero en salir.)

Se define la cola, como una estructura de datos homogénea de tamaño variable que soporta el siguiente modo de acceso: *inserción* por un extremo y *extracción* por el opuesto. Operaciones posibles sobre la cola son la *creación* de una cola vacía, la *determinación* de si la cola está o no vacía y la *destrucción* de la cola.

De igual modo que se ha visto para la pila, una cola se puede implementar haciendo uso de las estructuras dinámicas de datos. En este caso, se simula mediante una lista enlazada que mantiene dos punteros, uno llamado *frente* por donde se van extrayendo los elementos más antiguos de la cola, y otro llamado *final*, que señala al extremo por el cual se irán añadiendo elementos a la cola. De esta forma, el puntero *final* nos garantiza que cada vez que se tenga que extraer un elemento de la cola, no sea necesario recorrer ésta desde el principio, lo cual sería altamente ineficiente para colas largas. Las operaciones de insertar y extraer de la cola tienen acceso directo a los nodos de la cola en los que se van a realizar estas operaciones. La diferencia con las pilas reside en el modo de entrada y salida de los datos, en las colas las inserciones se realizan al final de la lista, no al principio.



## Práctica

Se desea simular el comportamiento de un planificador de tareas (scheduler) de un ordenador que tiene varios procesadores. Cuando llegue un aviso con  $n$  procesadores libres, las tareas que estén esperando serán ejecutadas por orden de llegada hasta completar los procesadores vacíos o no haber esperando más trabajos.

El comportamiento del scheduler es idéntico a una estructura de Cola. Por lo que se debe implementar una Cola (a partir de la definición que se adjunta **Y SIN MODIFICAR LA DEFINICIÓN DE LA MISMA**) y un programa cliente con las siguientes opciones de menú:

Nombre: (Apellidos, Nombre)	Curso:
Especialidad: (Gestión/Sistemas)	Grupo: (A/B/C/D)
Puesto: número de ordenador	Fecha: dd/mm/aa
Práctica 18: Colas en C++	
A. Llegar Trabajo Scheduler	
B. Llegar Aviso Procesadores Libres	
C. Primer Trabajo en Espera	
D. Pintar Trabajos en Espera	
E. Leer Trabajos Fichero Texto	
F. Pintar Trabajos Fichero Texto	

- **Llegar Trabajo Scheduler.** Se pedirá por teclado el nombre del trabajo y se meterá en el scheduler.
- **Llegar Aviso Procesadores Libres.** Se pedirá por teclado el número de procesadores libres y se quitarán de la cola (mostrando su nombre por pantalla) tantos trabajos como procesadores libres haya o hasta que la cola esté vacía.
- **Primer Trabajo en Espera.** Mostrará por pantalla el nombre del primer trabajo en espera **SIN MODIFICAR EL CONTENIDO DE LA COLA.**
- **Pintar Trabajos en Espera.** Muestra por orden todos los trabajos que están en la cola **SIN MODIFICAR EL**

**CONTENIDO DE LA COLA.**

- **Leer Trabajos en Fichero de Texto.** Lee los trabajos desde un fichero de texto cuyo nombre se solicita al usuario y los inserta en la cola.
- **Pintar Trabajos en Fichero de Texto.** Escribe en un fichero cuyo nombre se pide al usuario, por orden, todos los trabajos que están en la cola **SIN MODIFICAR EL CONTENIDO DE LA COLA.**
- **Salir del Programa.** Se pedirá confirmación de salida.

Después de cada opción se hará una pausa y se limpiará la pantalla antes de volver al menú principal.

**NOTA.** Podrá usarse el módulo de cadenas de la práctica anterior.

```
// Fichero MCola.h
#ifndef _MCola_h_
#define _MCola_h_

#include "MCadena.h"

namespace MCola
{
    using namespace MCadena;

    typedef struct TNode *TLista;
    struct TNode
    {
        TCadena val;
        TLista sig;
    };

    struct TCola
    {
        TLista frente;
        TLista final;
    };

    TCola CrearCola();
    void DestruirCola(TCola &c);
    void MeterCola(TCola &c, TCadena s, bool &llena);
    void SacarCola(TCola &c, TCadena &s, bool &vacía);
    bool ColaLlena(TCola c);
    bool ColaVacía(TCola c);
}

#endif
```