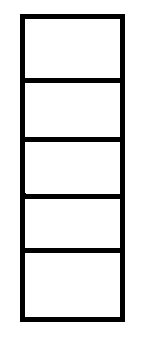
**COLAS**

Una cola es una [estructura de datos](http://es.wikipedia.org/wiki/Estructura_de_datos), caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pop por el otro. También se le llama estructura [FIFO](http://es.wikipedia.org/wiki/FIFO) (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

Las colas se utilizan en sistemas [informáticos](http://es.wikipedia.org/wiki/Inform%C3%A1tica), [transportes](http://es.wikipedia.org/wiki/Transporte) y operaciones de [investigación](http://es.wikipedia.org/wiki/Investigaci%C3%B3n) (entre otros), dónde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento. Este tipo de estructura de datos abstracta se implementa en [lenguajes orientados a objetos](http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) mediante clases, en forma de listas enlazadas.



## Usos concretos de la cola

La particularidad de una estructura de datos de cola es el hecho de que sólo podemos acceder al primer y al último elemento de la estructura. Así mismo, los elementos sólo se pueden eliminar por el principio y sólo se pueden añadir por el final de la cola.

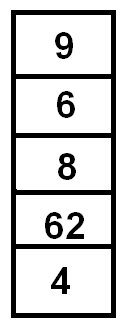
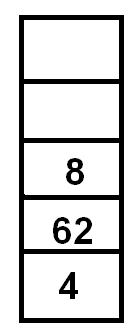
Ejemplos de colas en la vida real serían: personas comprando en un supermercado, esperando para entrar a ver un partido de béisbol, esperando en el cine para ver una película, una pequeña peluquería, etc. La idea esencial es que son todos líneas de espera.

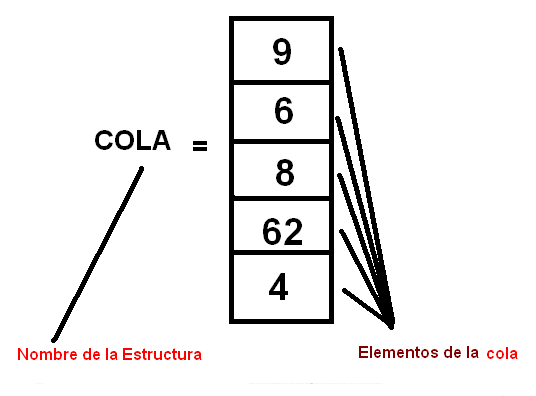
En estos casos, el primer elemento de la lista realiza su función (pagar comida, pagar entrada para el partido o para el cine) y deja la cola. Este movimiento está representado en la cola por la función pop o desencolar. Cada vez que otro elemento se añade a la lista de espera se añaden al final de la cola representando la función *push* o encolar. Hay otras funciones auxiliares para ver el tamaño de la cola (*size*), para ver si está vacía en el caso de que no haya nadie esperando (*empty*) o para ver el primer elemento de la cola (*front*).

**Indexación base-cero (0):** En este modo el primer elemento del vector será la componente cero ('0') del mismo, es decir, tendrá el índice '0'. En consecuencia, si el vector tiene 'n' componentes la última tendrá como índice el valor 'n-1'. Java y el [lenguaje C](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C) es un ejemplo típico que utiliza este modo de indexación.

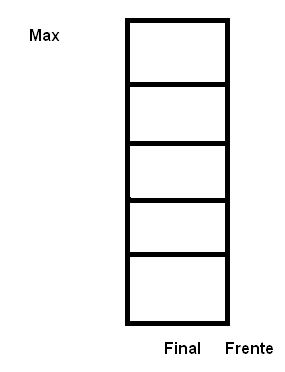
## Operaciones Básicas

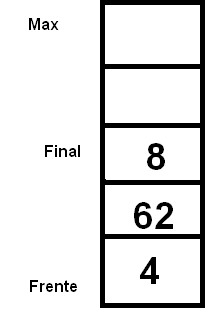
* **Crear**: se crea la cola vacía.
* **Encolar** (añadir, entrar, push): se añade un elemento a la cola. Se añade al final de esta.
* **Desencolar** (sacar, salir, pop): se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.
* **Frente** (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primero elemento que entró.





Nota: No confundir los elementos de la Cola con la posición o índice de la Misma





Para acceder a los elementos de la estructura es necesario hacer referencia al nombre e indicar la posición.

Z= Cola[Frente]

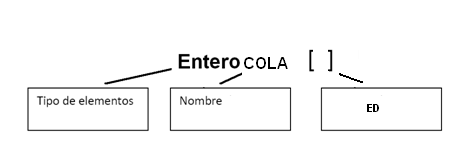
Cola [Final]= 9

Nota: En adelante se hace referencia a las estructuras creadas con base a la utilización de clases, métodos y el nivel de accesibilidad de los mismos.

**I**

**DECLARACION DE LA ESTRUCTURA**

**1.1CREACION DE LA COLA**



**1.2 Definir las variables que manipularán la estructura.**

En esta parte del diseño es necesario definir las variables índices, variables de cálculos y operaciones que regirán los métodos con ámbito global y el nivel de visibilidad. Para darle una correcta documentación al diseño es recomendable comentar el uso que tendrá dicha variable en la estructura que se está diseñando

publico Entero Cola []; // Estructura de la Cola

publico Entero Final, Frente, Max , Elem, pos; // variables para la cola

//Final : El ultimo elemento insertado en la cola

//Frente:El primer elemento de la cola

//Max : Máximo de espacios que tiene la Cola

//Elem : Elemento que se agrega a la cola (Tecleado por el usuario)

// pos : determina la posición de un dato consultado

**II**

**INICIALIZACIÓN DE LA ESTRUCTURA**

**Elaboración de un método constructor para inicializar la estructura.**

2.1 Se toman como referencia para el tamaño de la cola el parámetro contenido en las variables **Espacios**, el dato de este parámetro es obtenido a través de otra clase Que llamaremos MenuPpal donde el usuario determina el tamaño de la Cola.

2.2 Se indica nivel de acceso público a la estructura creada.

2.3 Creación de la Cola se representa como estructura de datos definida por el usuario

* 1. Se inicializan los valores de las variables de la estructura.

**//----- Contructor -------\\**

publico Cola (entero Espacios) // se recibe como parámetro los Espacios de la COLA

Inicio

Cola = new entero [Espacios]; // Asignamos los espacios en la Pila

Max = Espacios - 1; cola.length-1 // Se determina la capacidad máxima de almacenamiento

FInal = -1;

Frente=-1;

Fin

**//------- Fin del constructor ------\\**

**III**

**Métodos**

Los métodos son subrutinas que gobernaran a la estructura de datos creada, los métodos permitirán definir el comportamiento de la estructura durante la ejecución del programa.

Las operaciones cotidianas que realizan las pilas son las siguientes

* Inserción ( Push)
* Eliminación (Pop)
* Búsqueda
* Visualización

Inicialmente definiremos los métodos que indicarán el estado de la estructura en este caso nos interesa saber cuando la pila está llena y cuando esta vacía para lo cual realizaremos métodos booleanos que determinar si es falso o verdadero el estado consultado.

**3.1 Método para determinar si una cola esta vacía.**

Publico booleano ColaVacia()

Inicio

Si (Frente==-1) o (Frente > Final)

Inicio

retorne Verdadero;

sino

retorne falso;

Fin Si

Fin Metodo

**Explicación:**

**3.2 Método para determinar si una Cola está Llena.**

Publico booleano ColaLLena()

Inicio

Si (Final==Max)

Inicio

retorne Verdadero;

sino

returne falso;

Fin Si

Fin Metodo

**Explicación:** La variable Max está almacenado la capacidad máxima de almacenamiento de la cola, por lo tanto se comparan el índice en este caso la variable final con la capacidad en caso tal que sean iguales nos indicaría que la estructura está Llena.

**3.3 Método para insertar un elemento a la COLA.**

Publico Push()

Inicio

Si(!colaLlena())

Elem = Leer(" Digite numero ");

Final = final +1;

Cola[Final] = Elem;

si (ColaLlena())

Escriba (" La Cola ha quedado llena !! ");

Fin si

sino

Escriba(" Pila llena !! Imposible introducir un valor a la Cola");

Fin Si

Fin Metodo

**Explicación:** Se verifica si se pueden insertar elementos invocando la negación del método Colallena, si es posible se incrementa el final se inserta el elemento capturado en la cola.

**3.4 Método para Eliminar un elemento de la Pila.**

Publico Pop ()

Inicio

Si(!colaVacia())

Escriba("Se borro el Dato : " + cola[Frente]);

Frente = Frente+1;

sino

Escriba ("Cola Vacia... Imposible Eliminar");

Fin Si

Fin Metodo

**Explicación:** Se verifica si la cola esta vacía invocando la negación del método Cola Vacía, si hay elementos se incrementa el frente o null según sea el caso para indicar la eliminación del elemento.