FACULTA DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS

TDA COLA

UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA

Definición:

Una cola es una agrupación de elementos de determinada naturaleza o tipo (personas, números, procesos informáticos, automóviles, etc) entre los que existe definida una relación de orden (**estructura de datos**). En función del tiempo, elementos de dicha naturaleza pueden llegar a la cola o salir de ella.

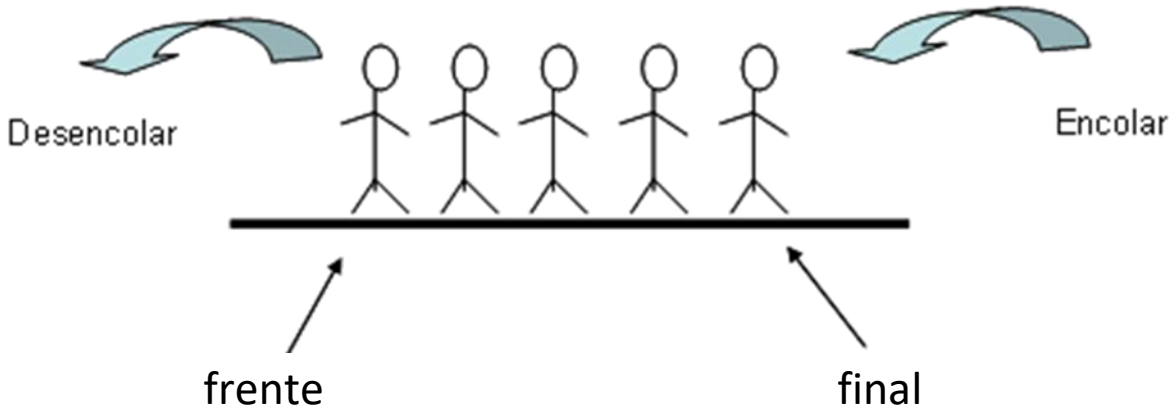
Ejemplos:

- La cola en un banco.
- La cola en un teléfono público.

Definición

En cualquiera de estos ejemplos, los elementos se retiran por un extremo y se añaden por otro extremo.

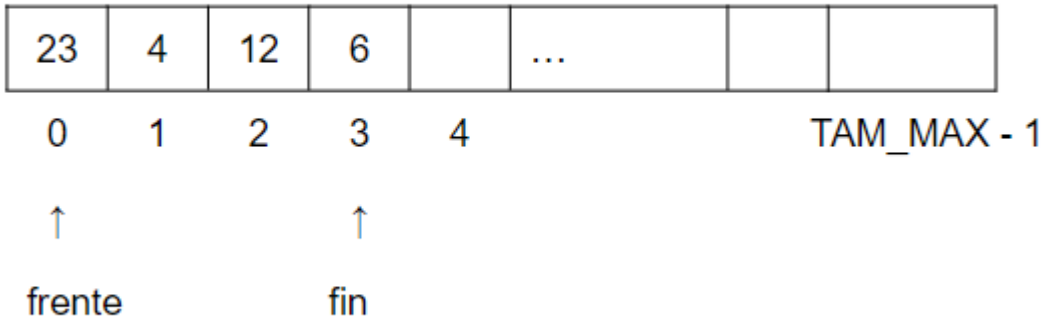
Las colas siguen la lógica FIFO (First In First Out), porque su característica principal es que el Primero en Entrar es el Primero en Salir.



UNIVERSIDAD DE LIMA

Representación en memoria

Utilizaremos un arreglo llamado **C** de tamaño **TAM_MAX** y dos variables **Frente** y **Final** que indican las posiciones del elemento que se encuentra en el frente y final de la cola C.



La cola C almacena números enteros.

UNIVERSIDAD DE LIMA

OPERACIONES

- ❑ crearCola
- ❑ estaVacia
- ❑ estaLLena
- ❑ encolar
- ❑ desencolar

UNIVERSIDAD DE LIMA

(1) Operación crearCola

Especificaciones:

Objetivo: crea la cola C vacía
Entrada: ninguna
Precondición: ninguna
Salida: ninguna
Postcondición: Cola C está vacía

```
public Queue(){
        C = new int[MAX_SIZE];
        front = -1;
        rear = -1;
}
```

UNIVERSIDAD DE LIMA

(2) Operación estaVacia

Especificaciones:

Objetivo: verifica si la cola está vacía.

Entrada: ninguna

Precondición: ninguna

Salida: estado (lógico)

Postcondición: estado verdadero si la cola está vacía, falso en caso contrario.

```
public boolean isEmpty(){
    boolean res = false;
    if(front == -1 && rear == -1){
        res = true;
    }
    return res;
}
```

UNIVERSIDAD DE LIMA

(3) Operación estaLlena

Especificaciones:

Objetivo: verifica si la cola está llena.

Entrada: ninguna

Precondición: ninguna

Salida: estado (lógico)

Postcondición: estado verdadero si la cola está llena, falso en caso contrario.

```
public Boolean isFull(){
    boolean res = false;
    if((front == 0 && rear == MAX_SIZE- 1) || front == rear + 1){
        res = true;
    }
    return res;
}
```

UNIVERSIDAD DE LIMA

(4) Operación Encolar

Especificaciones:

Objetivo: añadir un nuevo elemento a la cola.
Entrada: item
Precondición: ninguna
Salida: ninguna
Postcondición: Cola con un nuevo elemento encolado.

```
public void enqueue(Integer item){
    if(!this.isFull()){
        if(this.isEmpty()){
            front = 0;
            rear = 0;
        }else if(rear == MAX_SIZE - 1){
            rear = 0;
        }else{
            rear++;
        }
        C[rear] = item;
    }
}
```

(5) Operación desencolar

Especificaciones:

Objetivo: devolver el elemento que está al frente de la cola y eliminarlo.
Entrada: Cola c
Precondición: La cola no está vacía
Salida: elemento
Postcondición: Cola con un elemento menos.

```
public Integer dequeue(){
    Integer element = null;
    if(front != -1 && rear != -1){
        element = C[front];
        if(front == rear){
            front = -1;
            rear = -1;
        }else{
            if(front == MAX_SIZE - 1){
                front = 0;
            }else{
                front++;
            }
        }
    }
    return element;
}
```