



# Algoritmos Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología  
Universidad Nacional de Tucumán  
2023

# FILA-QUEUE



# FILA

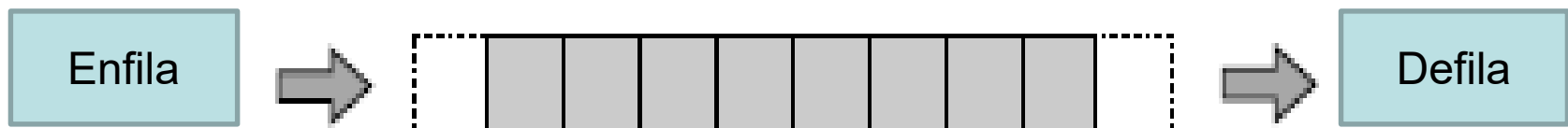
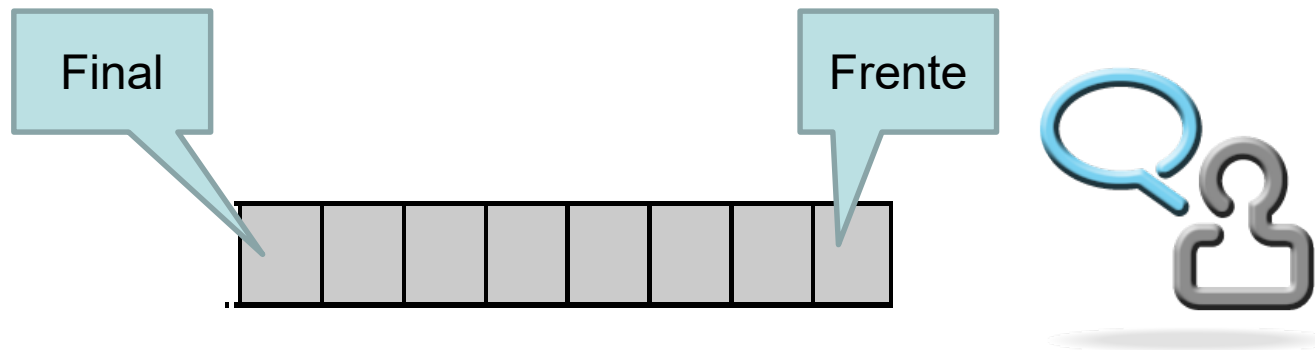
Una fila es una lista ordenada de cero o mas objetos de ***un mismo tipo*** que puede crecer por un extremo y decrecer por el otro de de sus extremos. Los extremos se llaman *final* y *frente* respectivamente.

Una fila se llama también una estructura de tipo **FIFO** (first in, first out), porque posee la siguiente propiedad: el primer elemento que llega a una fila es el primero en ser sacado.

Las filas tienen un orden lineal que es el “orden de llegada”.

Tienen muchas aplicaciones en sistemas operativos y simulaciones.

# FILA



# **FILA(ITEM)**

## Especificación Algebraica

### **OPERACIONES**

#### **A) Sintaxis:**

**FILAVACIA :  $\rightarrow$  FILA**

**ESFILAVACIA : FILA  $\rightarrow$  BOOL**

**FRENTE : FILA  $\rightarrow$  ITEM U {indefinido}**

**ENFILA : FILA X ITEM  $\rightarrow$  FILA**

**DEFILA : FILA  $\rightarrow$  FILA**

# FILA(ITEM)

## Especificación Algebraica

**B) Semántica:** Para todo  $q \in \text{FILA}$  ,  $\forall i \in \text{ITEM}$

$\text{ESFILAVACIA}(\text{FILAVACIA}) \equiv \text{TRUE}$

$\text{ESFILAVACIA}(\text{ENFILA}(q,i)) \equiv \text{FALSE}$

$\text{FRENTE}(\text{FILAVACIA}) \equiv \text{indefinido}$

$\text{FRENTE}(\text{ENFILA}(q,i)) \equiv \text{si } \text{ESFILAVACIA}(q) \text{ entonces } i$   
 $\text{sino } \text{FRENTE}(q)$

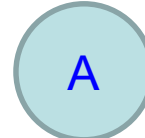
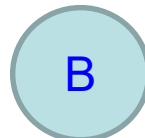
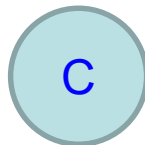
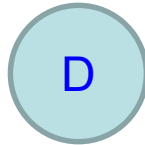
$\text{DEFILA}(\text{FILAVACIA}) \equiv \text{FILAVACIA}$

$\text{DEFILA}(\text{ENFILA}(q,i)) \equiv \text{si } \text{ESFILAVACIA}(q) \text{ entonces}$   
 $\text{FILAVACIA}$   
 $\text{sino } \text{ENFILA}(\text{DEFILA}(q),i)$

# FILA

## Construir una Fila

**FILA f :**



**FILAVACIA (f)**

**ENFILA(f, A )**

**ENFILA(f, B )**

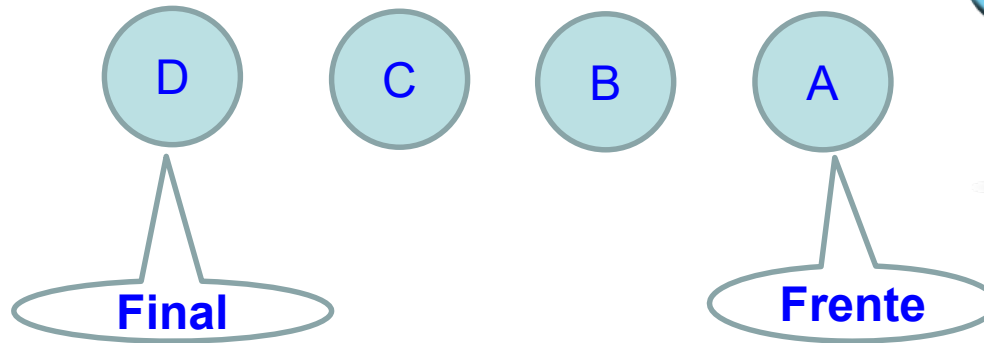
**ENFILA(f, C )**

**ENFILA(f, D )**

# FILA

## Construir una Fila

**FILA f :**



**ENFILA(ENFILA(ENFILA(ENFILA(FILAVACIA (f), A ), B ), C ), D )**



# FILA

## Aplicación: longitud de una Fila

### 1) Dentro de la Especificación Algebraica

Sintaxis:

LONGITUD : FILA  $\rightarrow$  entero  $\geq 0$

Semántica:  $\forall q \in \text{FILA}, \forall i \in \text{ITEM}$

LONGITUD ( FILAVACIA )  $\equiv 0$

LONGITUD ( ENFILA(q,i) )  $\equiv 1 + \text{LONGITUD}(q)$

# FILA

## Aplicación: longitud de una Fila

### 2) Como usuario del ADT **FILA**, función recursiva

Funcion LONGITUD (f) :  $FILA \rightarrow \text{entero} \geq 0$

Si ESFILAVACIA (f) entonces

Retorna 0

Sino

Retorna  $1 + \text{LONGITUD}(\text{DEFILA}(f))$

Fin

*NOTA: la función DEFILA modifica la Fila.*

# FILA

## Aplicación: longitud de una Fila

### 3) Como usuario del ADT FILA, función iterativa

Funcion LONGITUD (f) : FILA  $\rightarrow$  entero  $\geq 0$

Auxiliar: h  $\in$  entero

h  $\leftarrow$  0

Mientras NOT ESFILAVACIA(f) hacer

h  $\leftarrow$  h+1

DEFILA(f)

Retorna h

Fin

*NOTA: la función DEFILA modifica la Fila.*

# FILA

Aplicación: pertenece un Item a la Fila

## 1) Dentro de la Especificación Algebraica

Sintaxis:

PERTENECE : FILA x ITEM  $\rightarrow$  BOOL

Semántica:  $\forall q \in \text{FILA}, \forall i, j \in \text{ITEM}$

PERTENECE (FILAVACIA, i)  $\equiv$  FALSO

PERTENECE (ENFILA(q, i), j)  $\equiv$   $i=j$  OR  
PERTENECE(q, j)



*donde = es la operación que me permite comparar 2 objetos del tipo item*

# FILA

Aplicación: pertenece un Item a la Fila

2) Como usuario del ADT **FILA**, función recursiva

Funcion **PERTENECE** (f,x) : FILA x ITEM  $\rightarrow$  BOOL

Si ESFILAVACIA (f) entonces

Retorna FALSO

Sino

Retorna

FRENTE(f)=x OR **PERTENECE** (DEFILA(f),x)

Fin

*NOTA: la función DEFILA modifica la Fila.*

# FILA

Aplicación: pertenece un Item a la Fila

## 3) Como usuario del ADT FILA, función iterativa

Funcion PERTENECE (f,x) : FILA x ITEM  $\rightarrow$  BOOL

Auxiliar: esta  $\in$  BOOL

esta  $\leftarrow$  FALSO

Mientras NOT ESFILAVACIA(f) AND NOT esta hacer

esta  $\leftarrow$  FRENTE(f) = x

DEFILA(f)

Retorna esta

Fin

# FILA

Aplicación: pertenece un Item a la Fila

**4) Como usuario del ADT FILA,** otra función iterativa

Funcion PERTENECE (f,x) : FILA x ITEM  $\rightarrow$  BOOL

Mientras NOT ESFILAVACIA(f) AND NOT FRENTE(f) = x  
hacer

DEFILA(f)

Retorna NOT ESFILAVACIA(f)

Fin

# FILA

## Aplicación: igualdad de Filas

### 1) Dentro de la Especificación Algebraica

Sintaxis:

$\text{IGUALF} : \text{FILA} \times \text{FILA} \rightarrow \text{BOOL}$

Semántica:  $\forall f, q \in \text{FILA}, \forall i, j \in \text{ITEM}$

$\text{IGUALF}(\text{FILAVACIA}, \text{FILAVACIA}) \equiv \text{TRUE}$

$\text{IGUALF}(\text{FILAVACIA}, \text{ENFILA}(f, i)) \equiv \text{FALSE}$

$\text{IGUALF}(\text{ENFILA}(f, i), \text{FILAVACIA}) \equiv \text{FALSE}$

$\text{IGUALF}(\text{ENFILA}(f, i), \text{ENFILA}(q, j)) \equiv i = j \text{ AND } \text{IGUALF}(f, q)$

*donde = es la operación que me permite comparar 2 objetos del tipo item*



# FILA

## Aplicación: igualdad de Filas

**2) Como usuario del ADT FILA**, función recursiva

Funcion IGUALF (f1,f2) : FILA x FILA  $\rightarrow$  BOOL

Si ESFILAVACIA(f1) AND ESFILAVACIA(f2) entonces

Retorna TRUE

Sino

Si ESFILAVACIA(f1) OR ESFILAVACIA(f2) entonces

Retorna FALSE

Sino

Retorna      FRENTE(f1)=FRENTE(f2) AND  
                 IGUALF(DEFILA(f1),DEFILA(f2))

Fin

# FILA

## Aplicación: igualdad de Filas

### 3) Como usuario del ADT FILA, función iterativa

Funcion IGUALF (f1,f2) : FILA x FILA  $\rightarrow$  BOOL

Mientras NOT ESFILAVACIA(f1) AND  
NOT ESFILAVACIA(f2) AND  
FRENTE(f1)=FRENTE(f2) hacer

DEFILA(f1)

DEFILA(f2)

Retorna ESFILAVACIA(f1) AND ESFILAVACIA(f2)

Fin

# FILA

## Aplicación: concatenar Filas

### 1) Dentro de la Especificación Algebraica

Sintaxis:

$\text{CONCAT} : \text{FILA} \times \text{FILA} \rightarrow \text{FILA}$

Semántica:  $\forall p, q \in \text{FILA}, \forall i \in \text{ITEM}$

$\text{CONCAT}(q, \text{FILAVACIA}) \equiv q$

$\text{CONCAT}(q, \text{ENFILA}(p, i)) \equiv \text{ENFILA}(\text{CONCAT}(q, p), i)$

# FILA

## Aplicación: concatenar Filas

### 2) Como usuario del ADT **FILA**, función recursiva

Funcion **CONCAT** (f1,f2) : FILA x FILA  $\rightarrow$  FILA

Si ESFILAVACIA(f2) entonces

Retorna f1

Sino

ENFILA (f1, FRENTE(f2))

DEFILA (f2)

Retorna **CONCAT** (f1,f2)

Fin

# FILA

## Aplicación: concatenar Filas

### 3) Como usuario del ADT **FILA**, función iterativa

Funcion CONCAT (f1,f2) : FILA x FILA  $\rightarrow$  FILA

Mientras NOT ESFILAVACIA(f2) hacer

    ENFILA(f1,FRENTE(f2))

    DEFILA(f2)

Retorna f1

Fin

# FILA

## Aplicación: invertir la Fila



Fila original



Fila invertida

# FILA

## Aplicación: invertir la Fila

### 1) Dentro de la Especificación Algebraica

Sintaxis:

**INVERTIR : FILA  $\rightarrow$  FILA**

Semántica:  $\forall q \in \text{FILA}, \forall i \in \text{ITEM}$

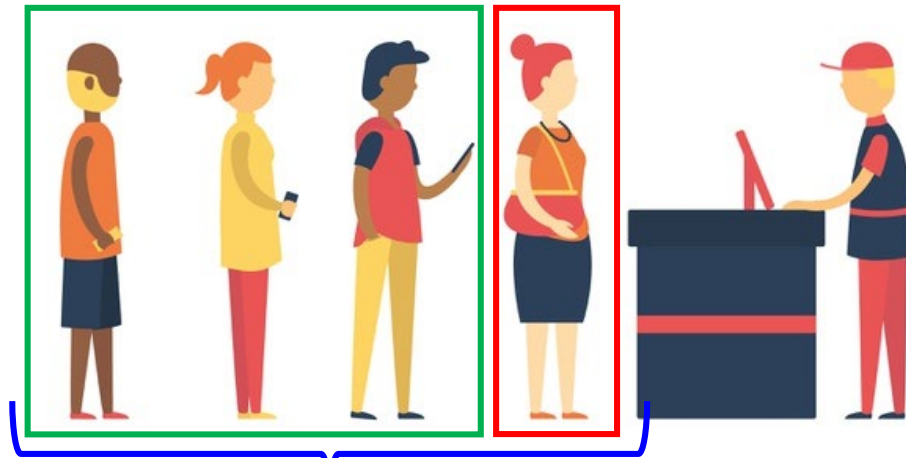
**INVERTIR (FILAVACIA)  $\equiv$  FILAVACIA**

**INVERTIR(ENFILA(q,i))  $\equiv$**

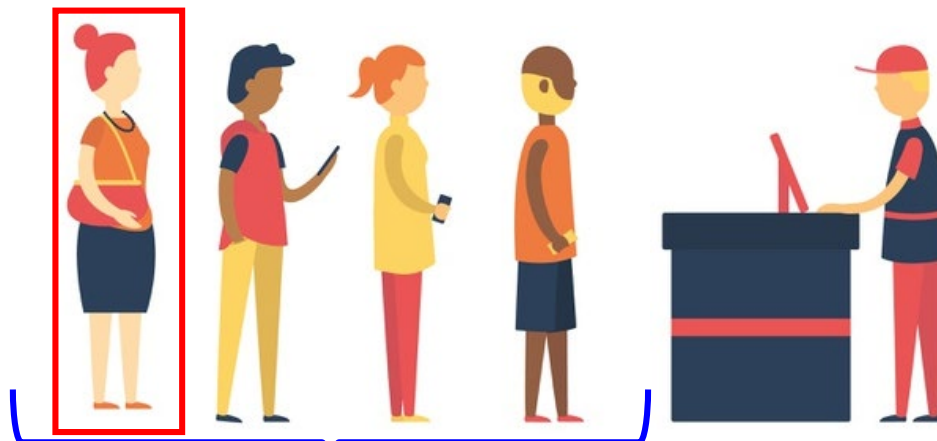
**ENFILA(INVERTIR(DEFILA(ENFILA(q,i)), FRENTE(ENFILA(q,i)))**

# FILA

## Invertir la Fila - algoritmo recursivo



FILA f



FILA finv



# FILA

## Aplicación: invertir la Fila

### 2) Como usuario del ADT FILA, algoritmo recursivo

Algoritmo INVERTIR (f,finv)

ENTRADA :  $f \in \text{FILA}$

SALIDA: finv  $\in \text{FILA}$ , llega inicializada en FILAVACIA

AUXILIAR: itemaux  $\in \text{ITEM}$

Si NOT ESFILAVACIA(f) entonces

itemaux  $\leftarrow$  FRENTE(f)



DEFILA(f)

INVERTIR(f,finv)

ENFILA(finv, itemaux)



Fin

# FILA

## Invertir la Fila - algoritmo iterativo

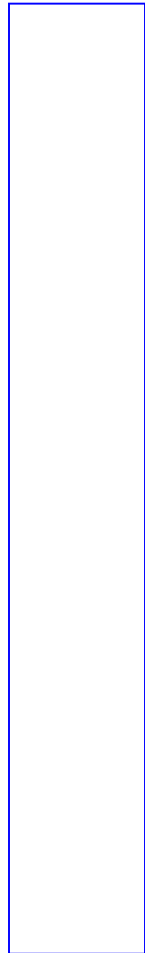


**FILA f**



**FILA finv**

**Pilaux**

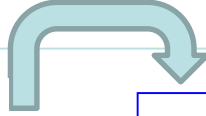


# FILA

## Invertir la Fila - algoritmo iterativo



**FILA f**



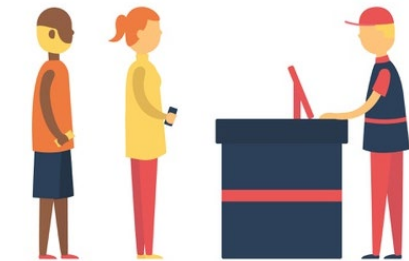
**FILA finv**

**Pilaux**



# FILA

## Invertir la Fila - algoritmo iterativo



**FILA f**



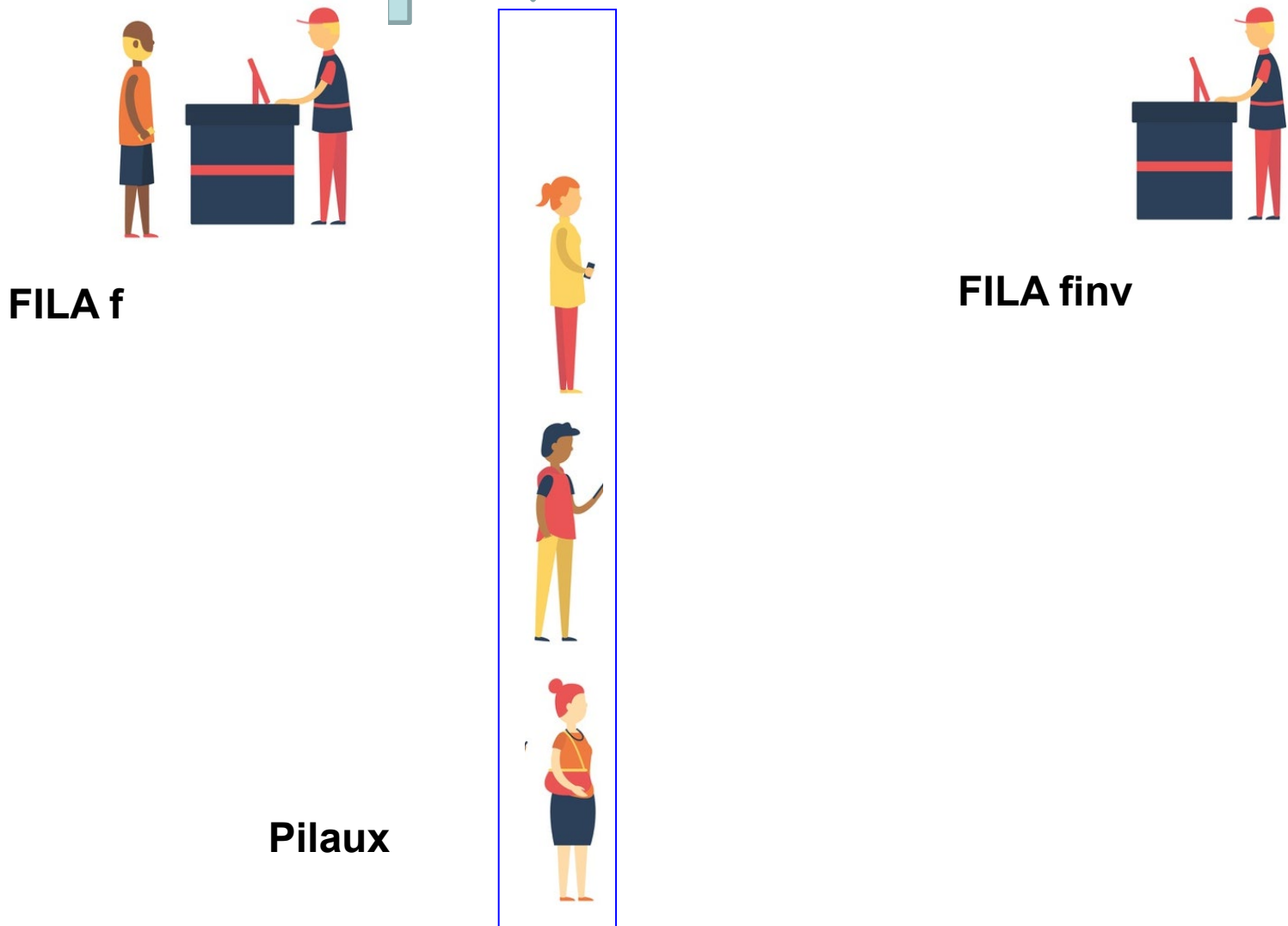
**FILA finv**



**Pilaux**

# FILA

## Invertir la Fila - algoritmo iterativo



# FILA

## Invertir la Fila - algoritmo iterativo

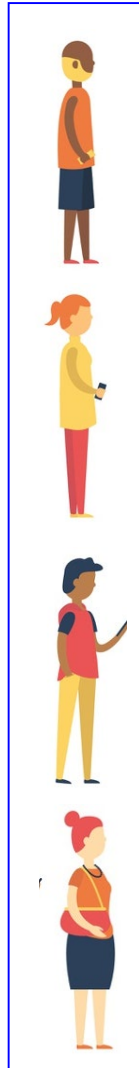
FILA f



FILA finv



Pilaux



# FILA

## Invertir la Fila - algoritmo iterativo

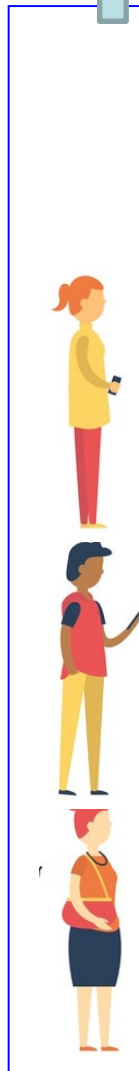
**FILA f**



**FILA finv**



**Pilaux**



# FILA

## Invertir la Fila - algoritmo iterativo

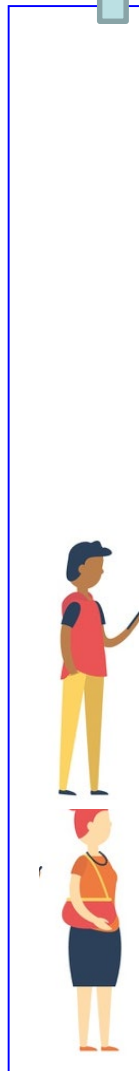
**FILA f**



**FILA finv**



**Pilaux**





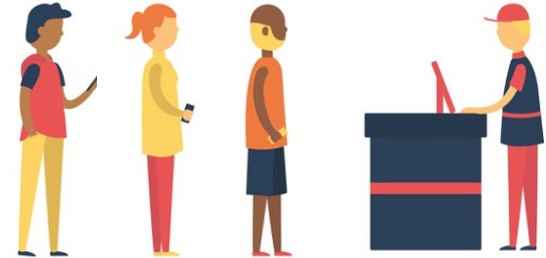
# FILA

## Invertir la Fila - algoritmo iterativo

**FILA f**



**FILA finv**



**Pilaux**



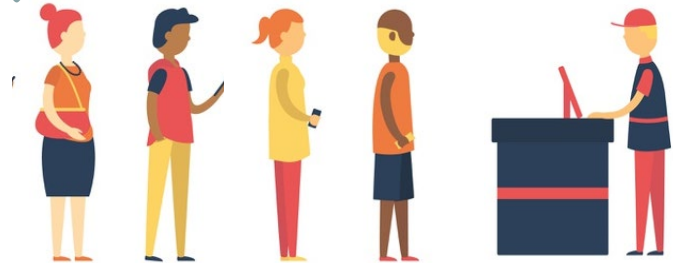
# FILA

## Invertir la Fila - algoritmo iterativo

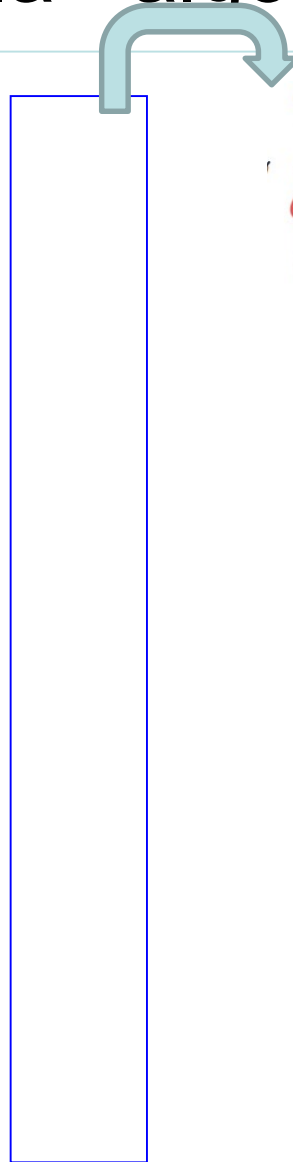
**FILA f**



**FILA finv**



**Pilaux**



# FILA

## Aplicación: invertir la Fila

### 3) Como usuario del ADT FILA, algoritmo iterativo

Algoritmo INVERTIR(f,finv)

ENTRADA :  $f \in \text{FILA}$

SALIDA:  $\text{finv} \in \text{FILA}$

AUXILIAR:  $\text{pilaux} \in \text{PILA}(\text{ITEM})$

PILAVACIA(pilaux)

Mientras NOT ESFILAVACIA(f) hacer

PUSH (pilaux, FRENTE(f))

DEFILA(f)

FILAVACIA(filainv)

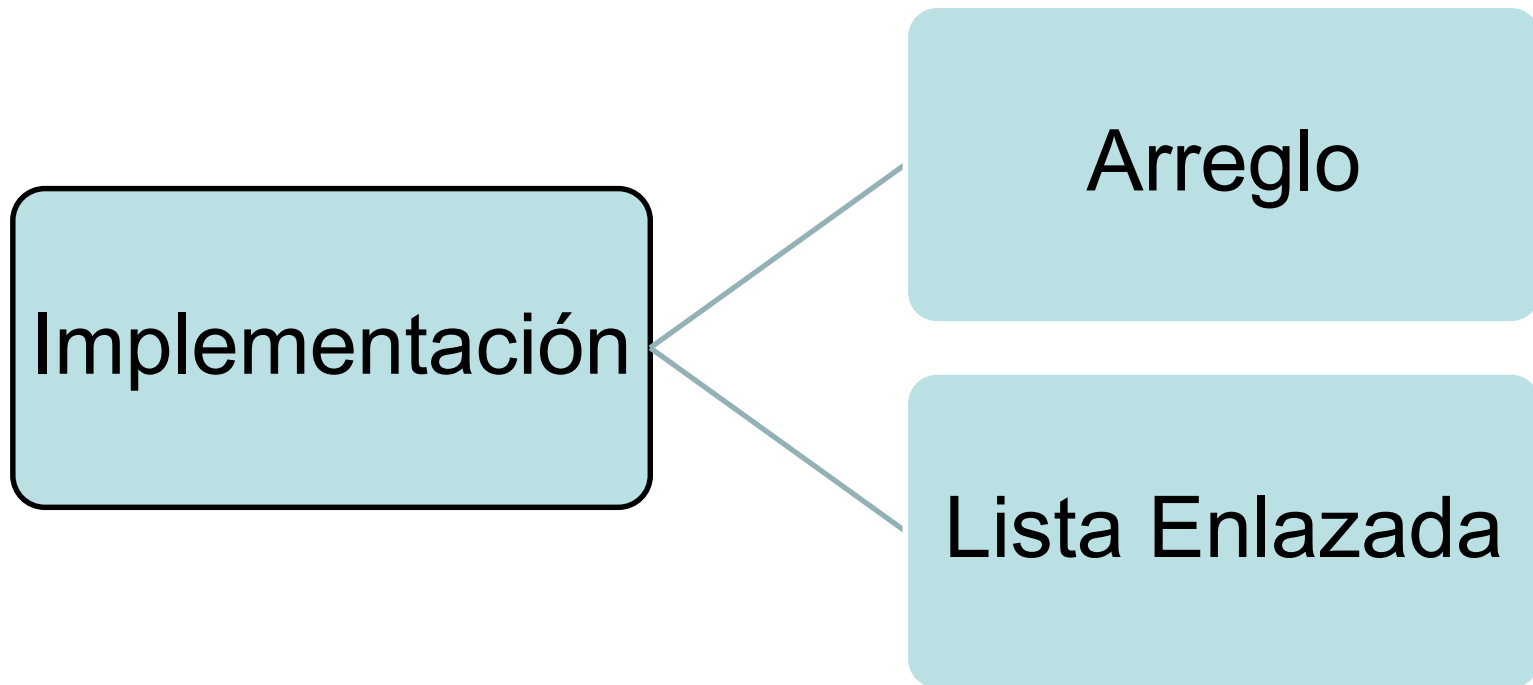
MIENTRAS NOT ESPILAVACIA(pilaux) HACER

ENFILA(filainv, TOP(pilaux))

POP(pilaux)

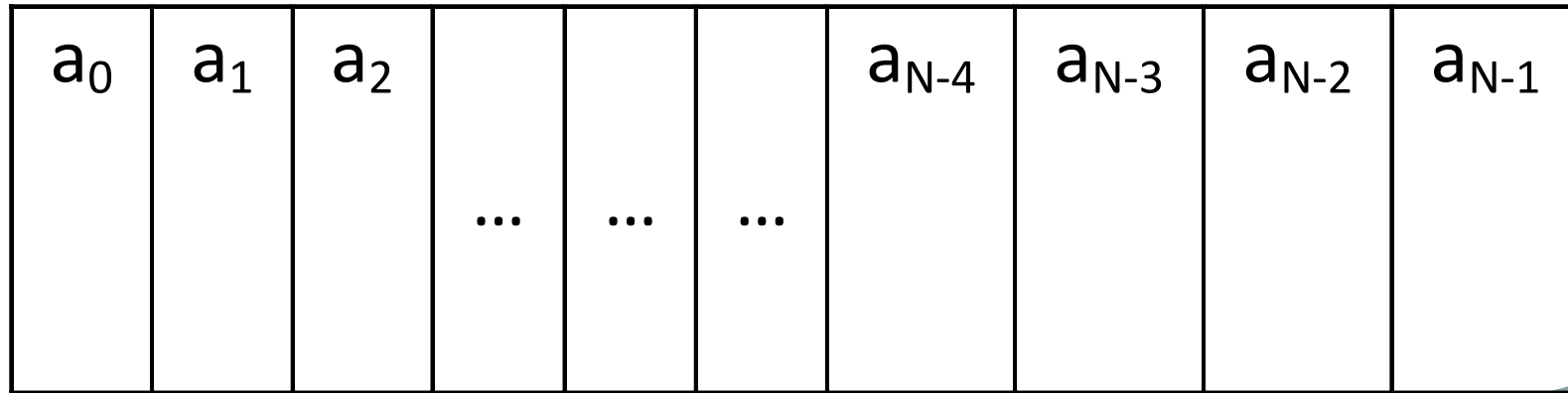
Fin

# FILA



Ventajas y Desventajas

# Implementación Fila con arreglo

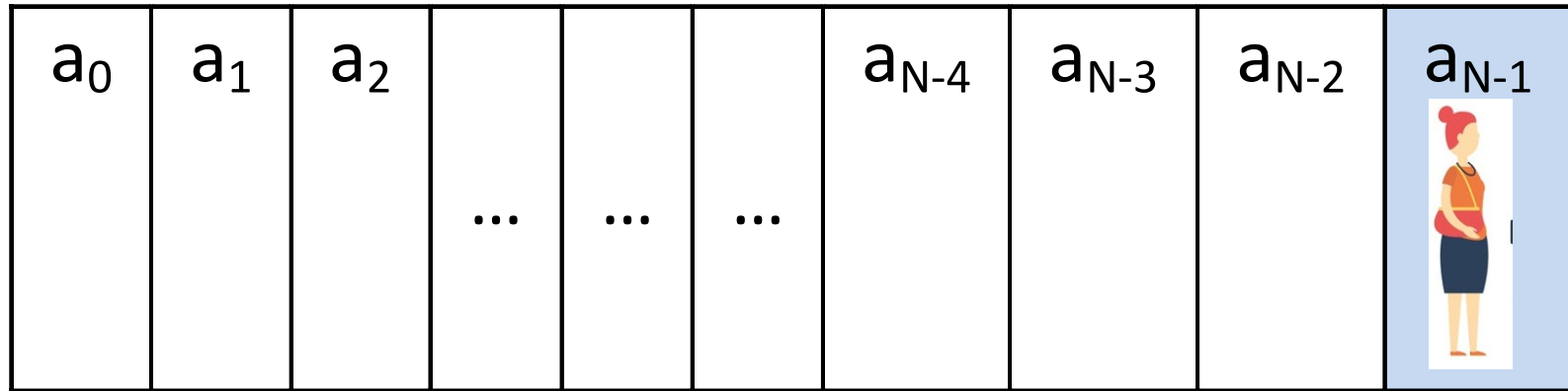


**FILAVACIA**

**Final**

**Frente**

# Implementación Fila con arreglo

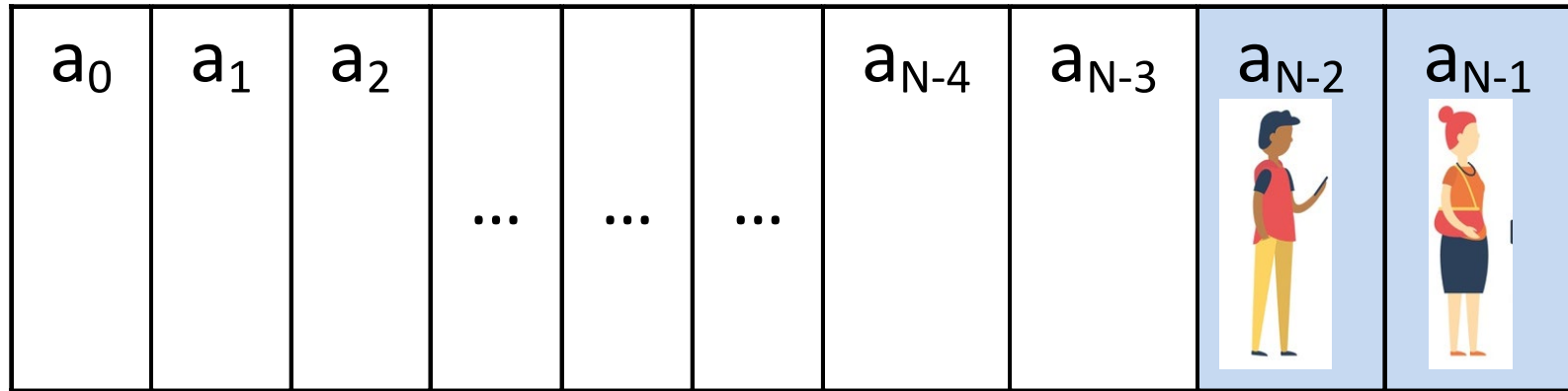


**ENFILA**

**Final**

**Frente**

# Implementación Fila con arreglo

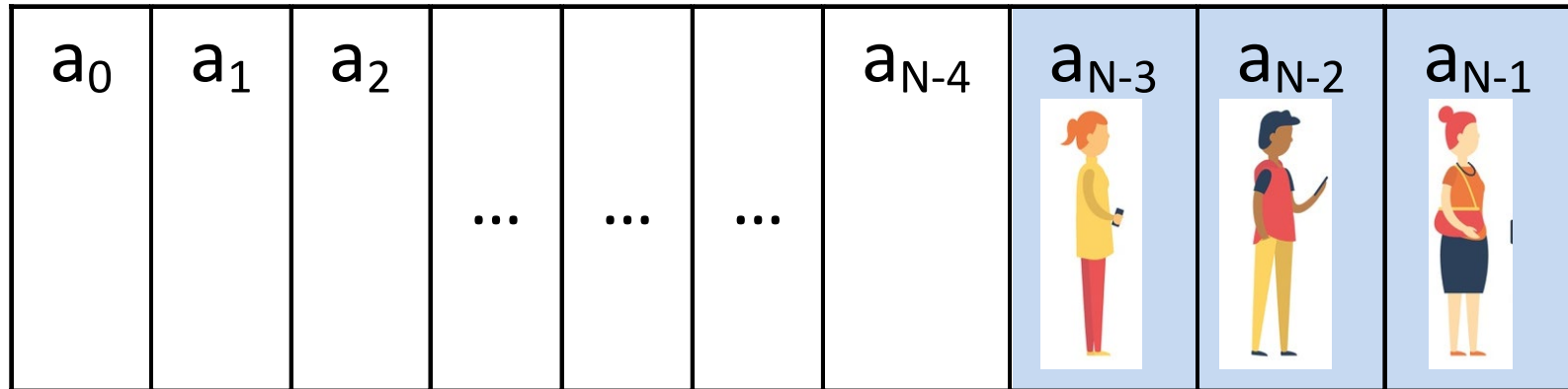


**ENFILA**

**Final**

**Frente**

# Implementación Fila con arreglo



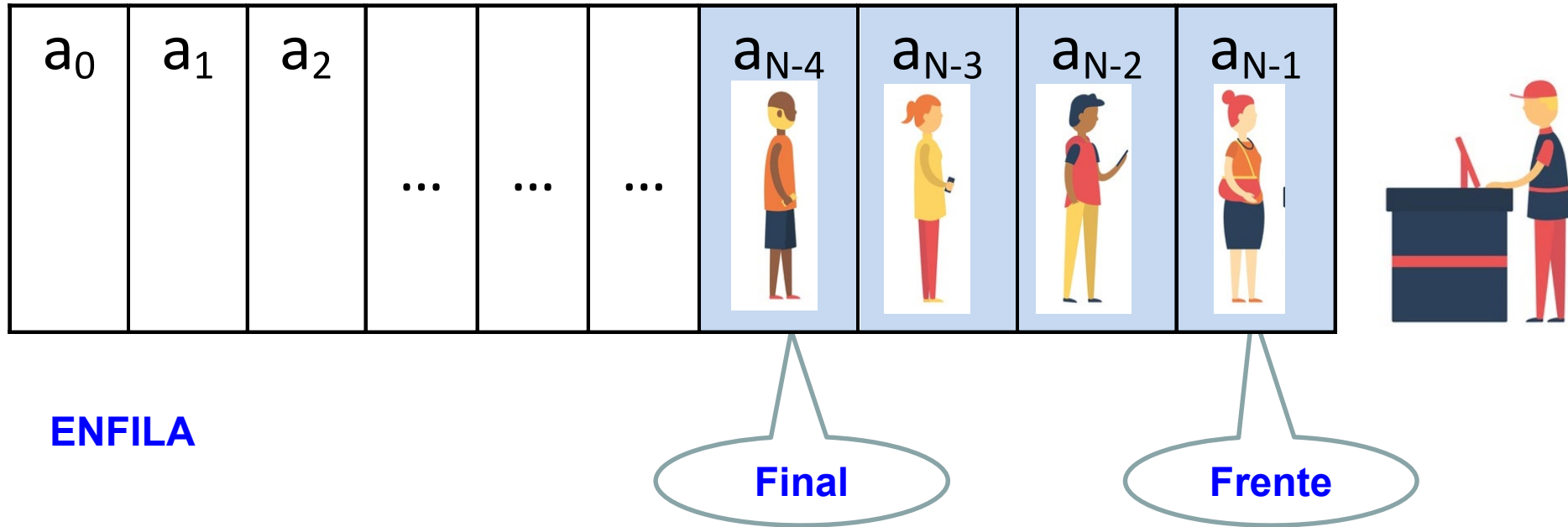
**ENFILA**

**Final**

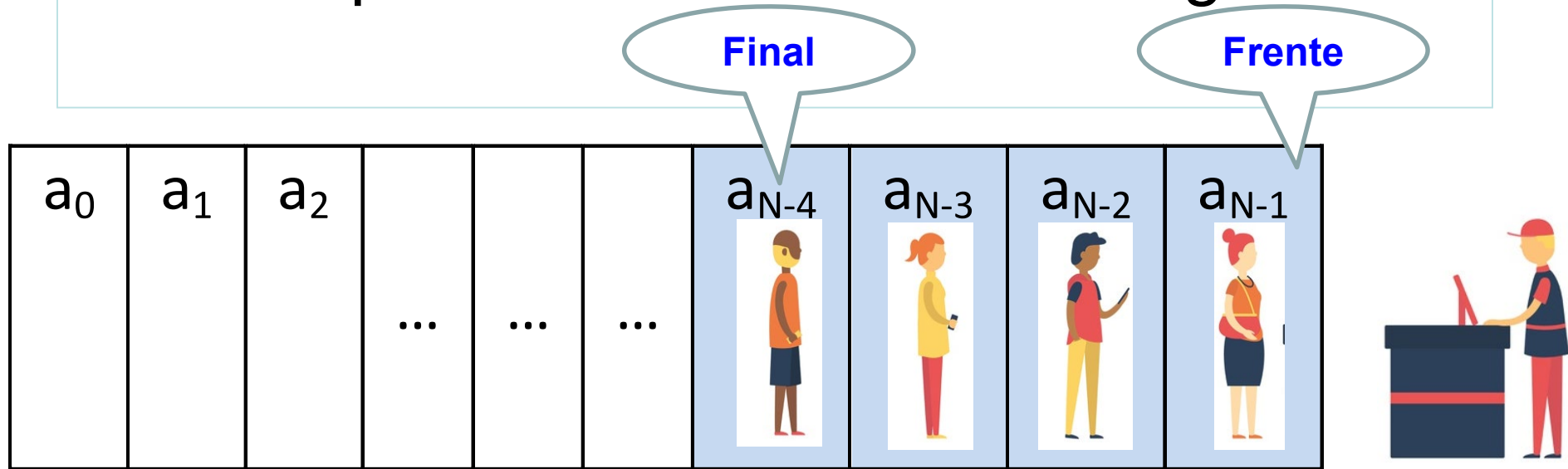
**Frente**



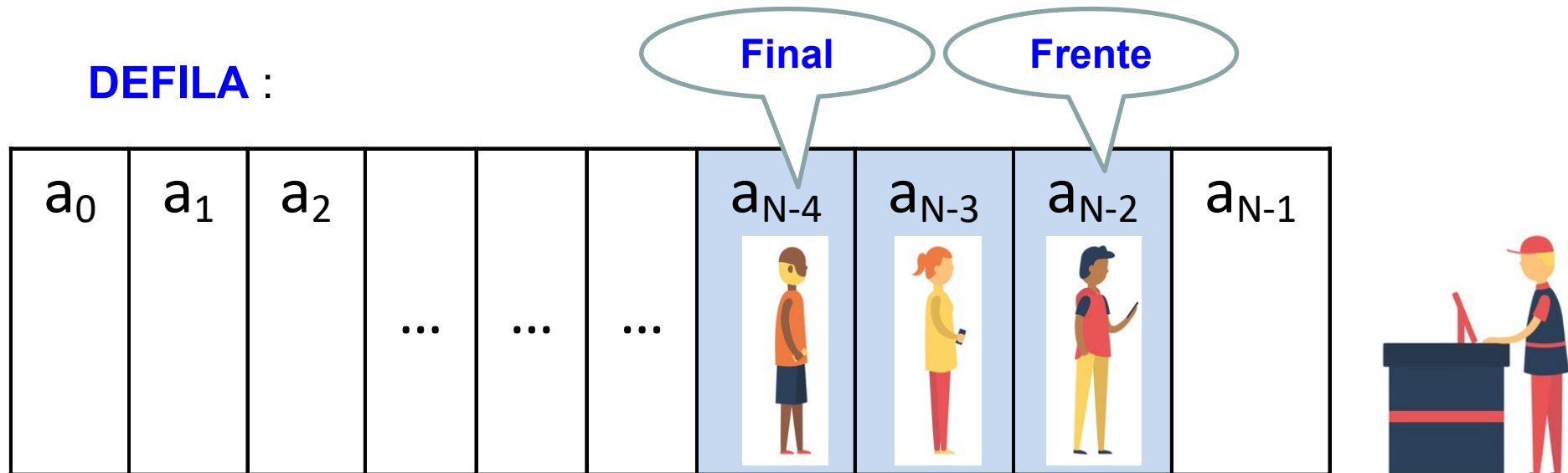
# Implementación Fila con arreglo



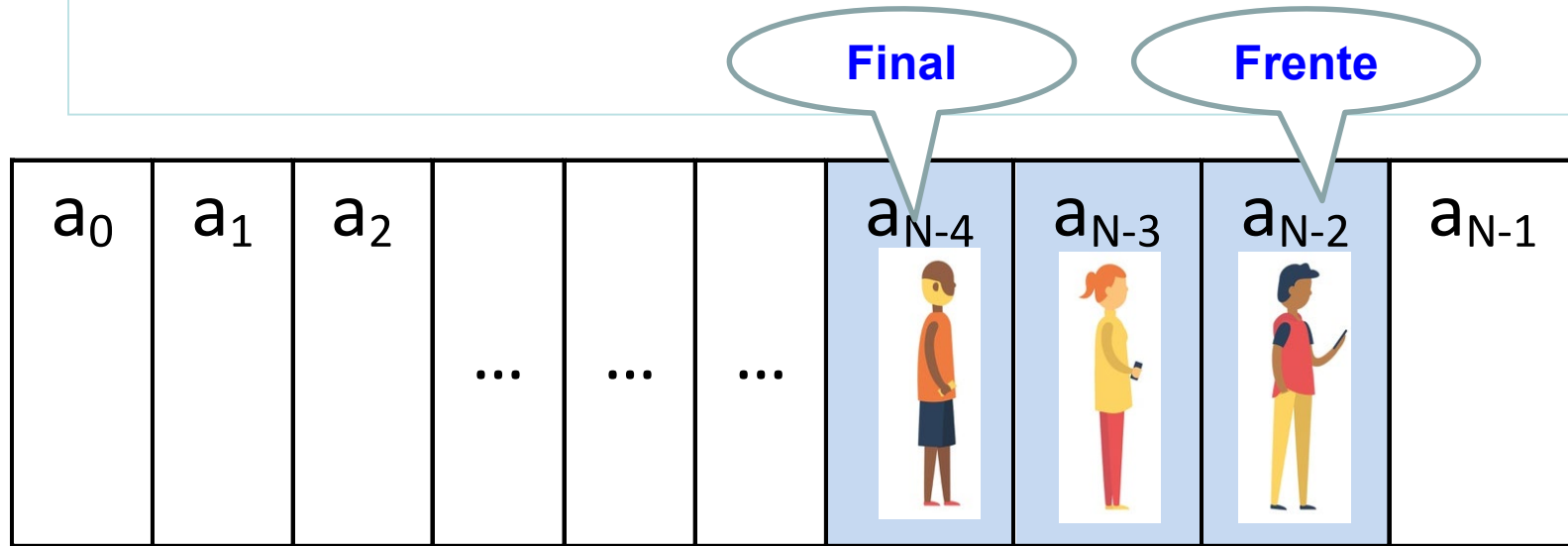
# Implementación Fila con arreglo



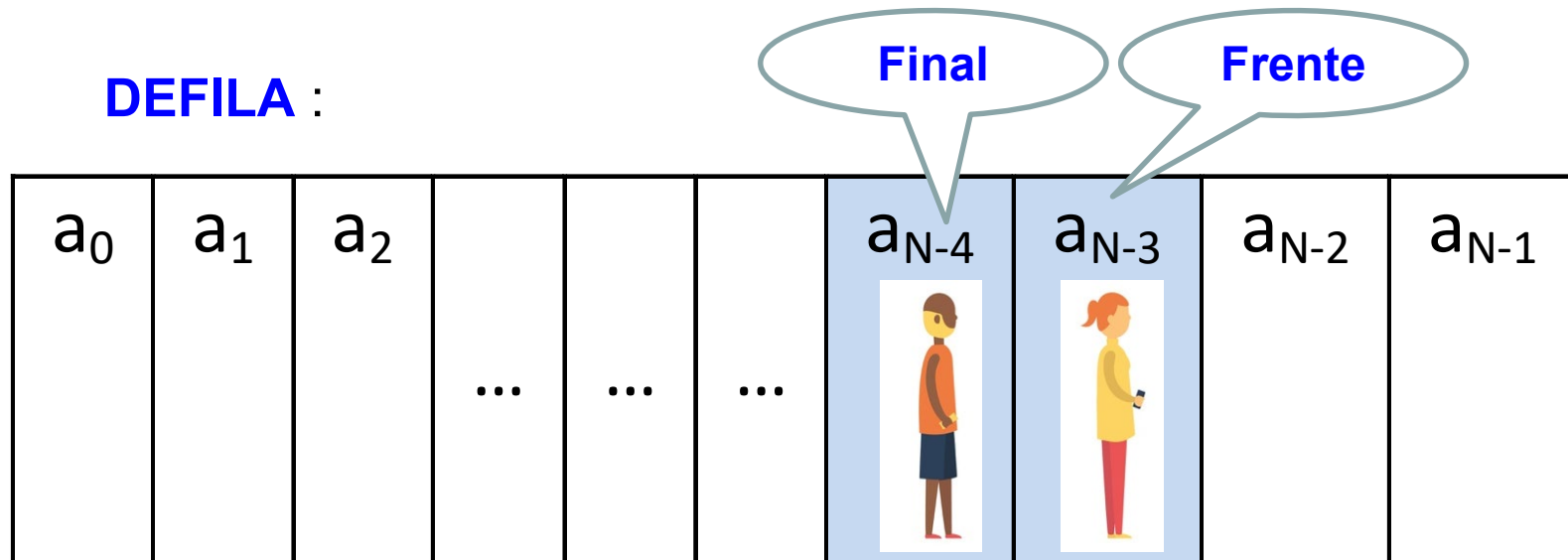
**DEFILA :**



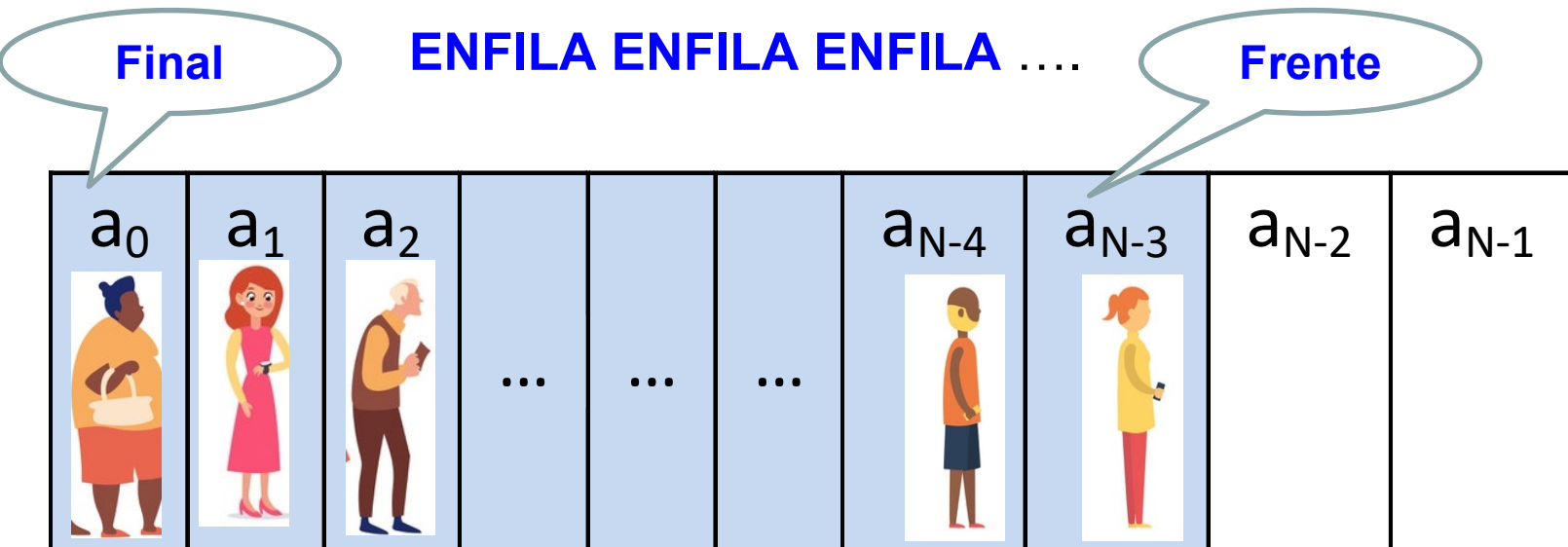
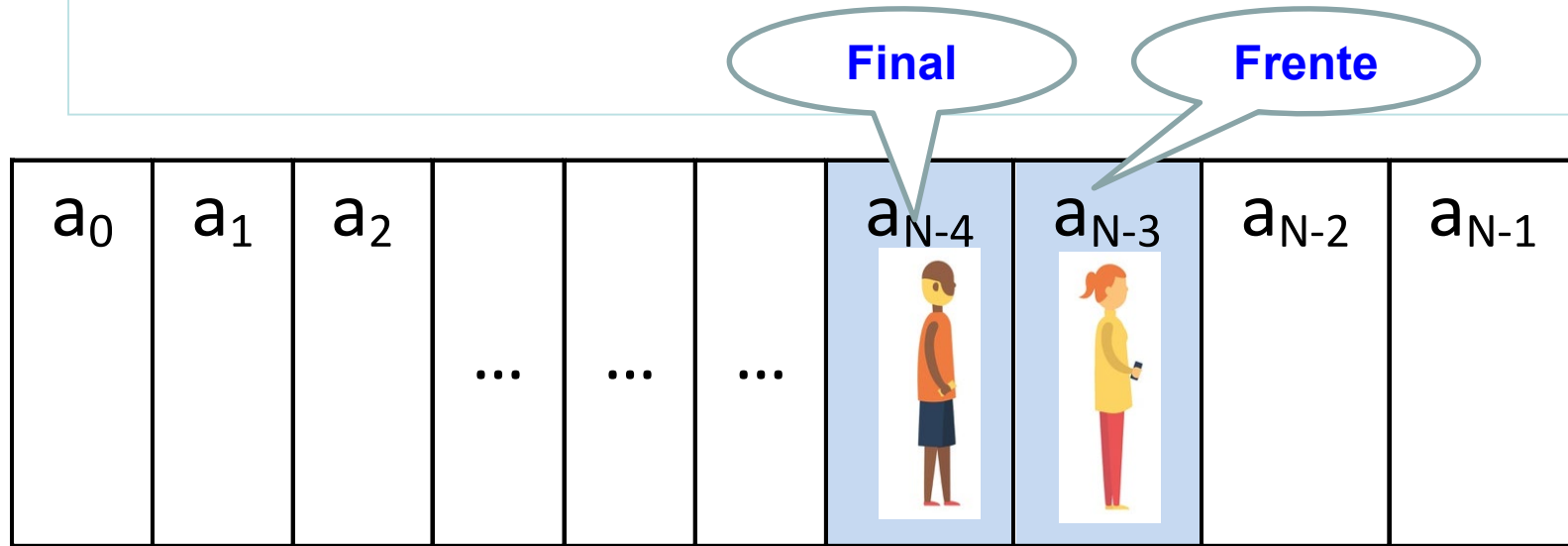
# Implementación Fila con arreglo



**DEFILA :**

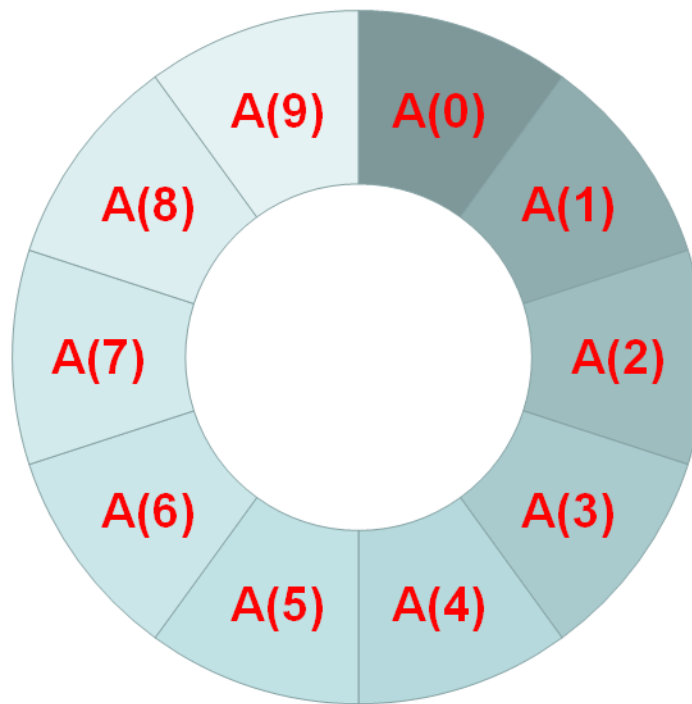


# Implementación Fila con arreglo



# Implementación Fila con Arreglo Circular

Arreglo Circular



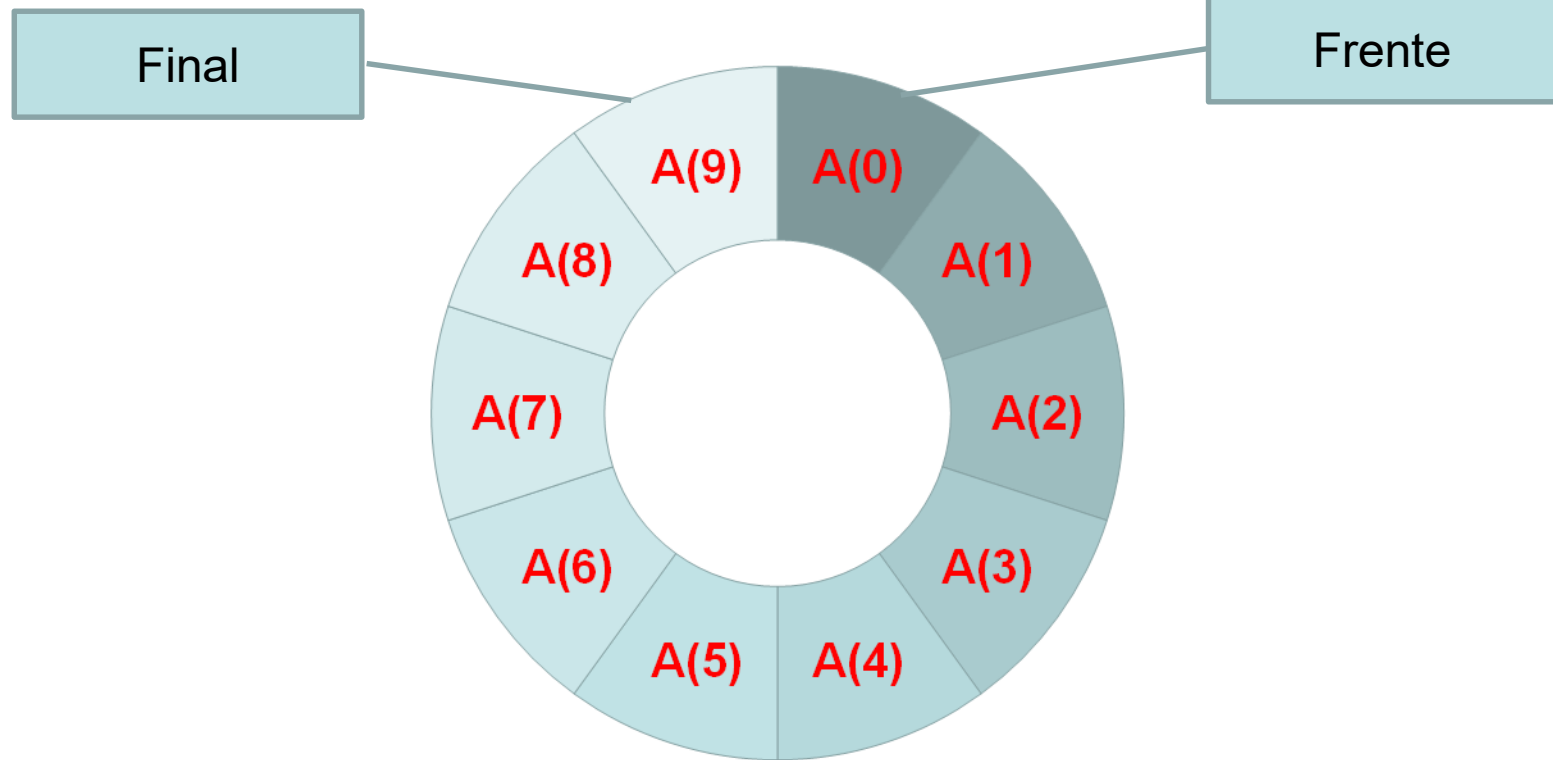
Indices:

Frente

Final

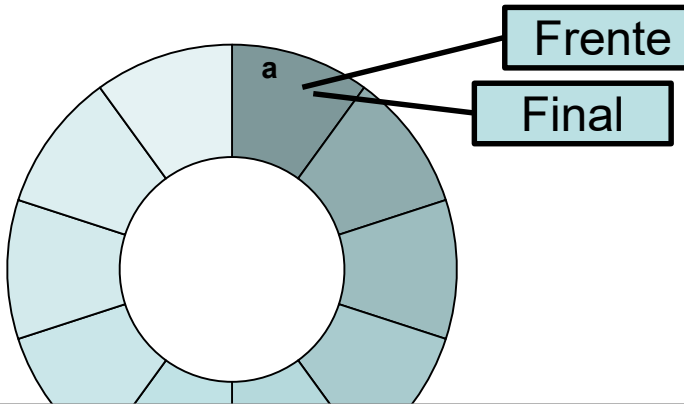
# Implementación Fila con Arreglo Circular

Fila Vacía(F)

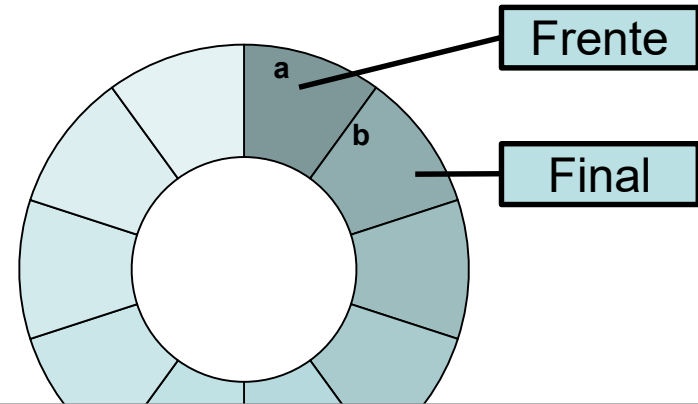


# Implementación Fila con Arreglo Circular

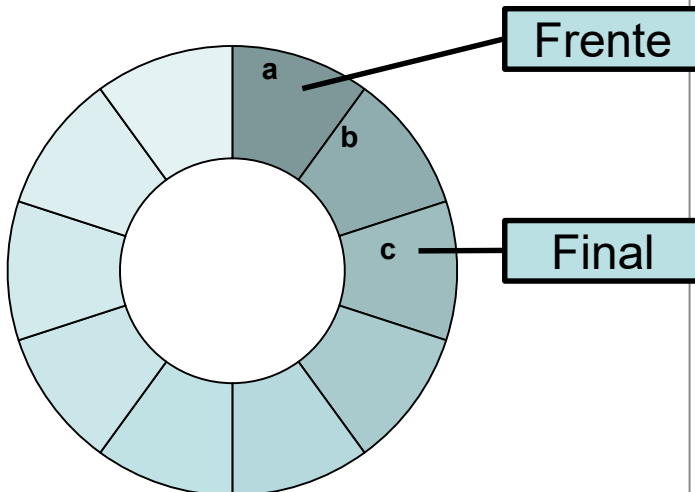
1) Enfila(F,a) →



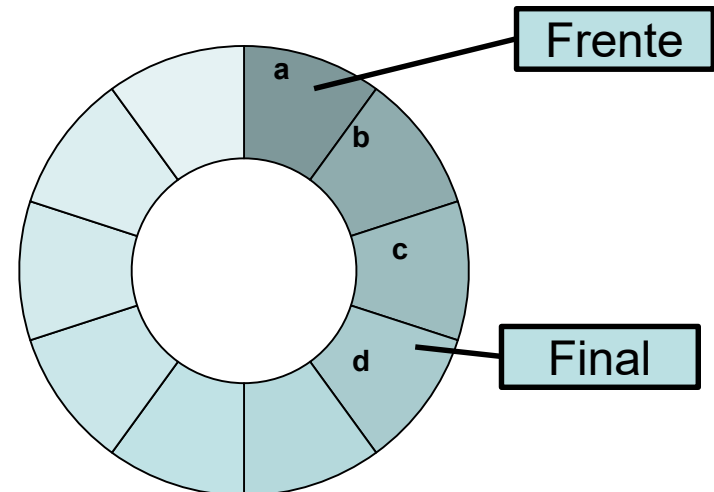
2) Enfila(F,b) →



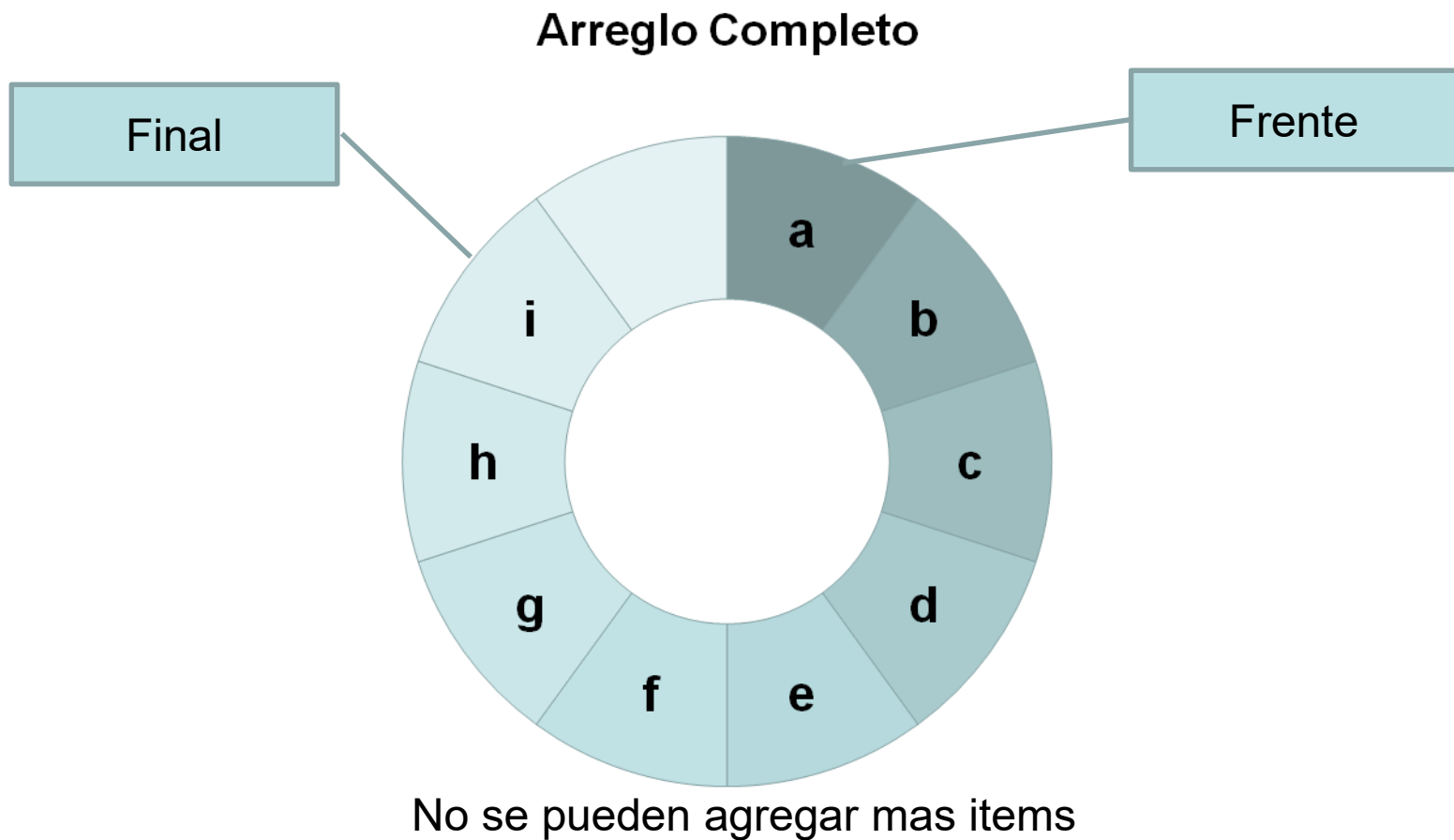
3) Enfila(F,c) →



4) Enfila(F,d) →



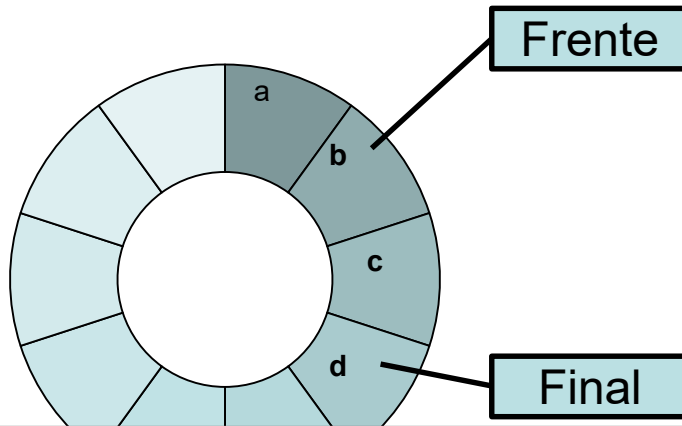
# Implementación Fila con Arreglo Circular



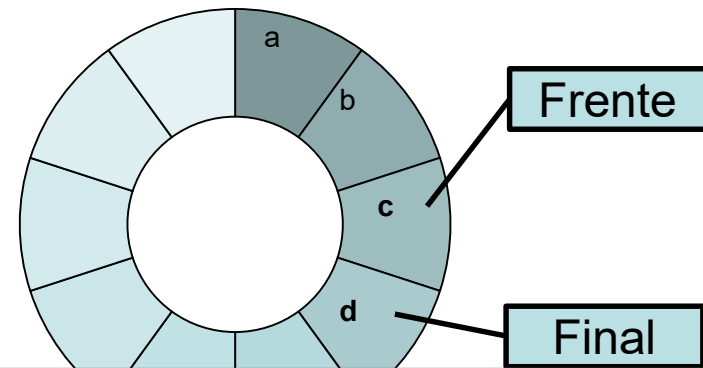


# Implementación Fila con Arreglo Circular

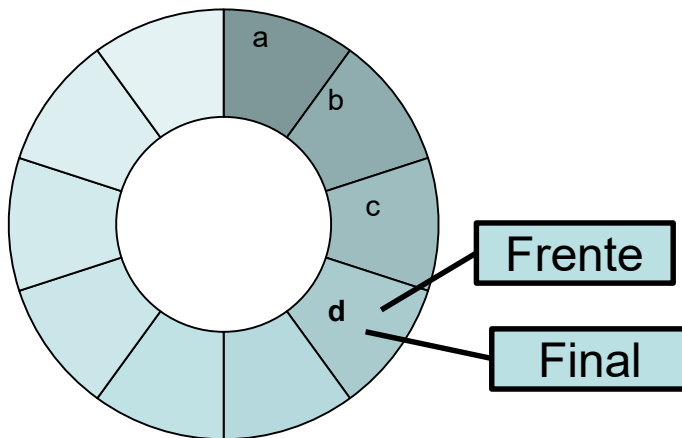
1) Defila(F) →



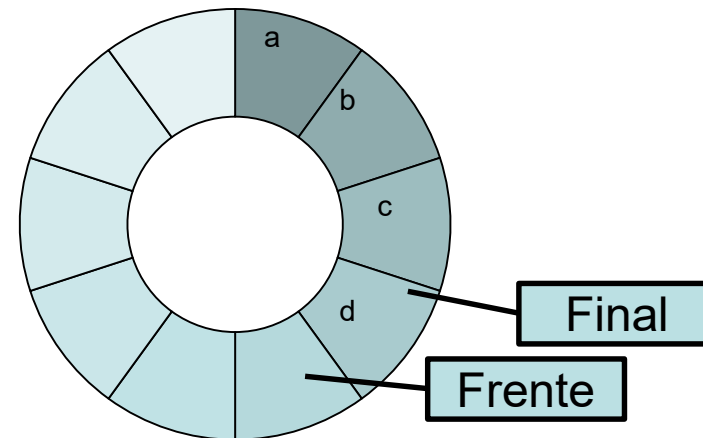
2) Defila(F) →



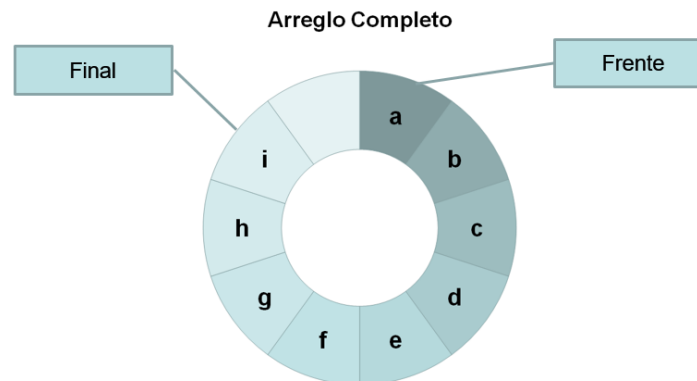
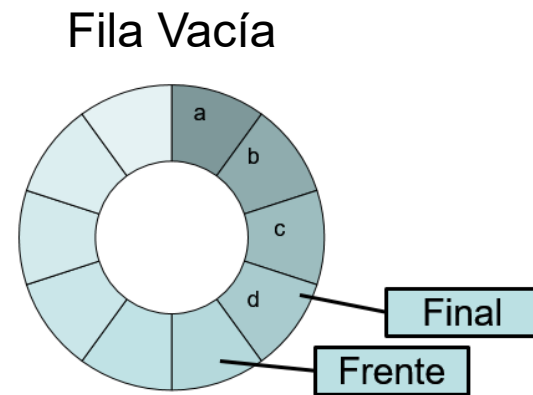
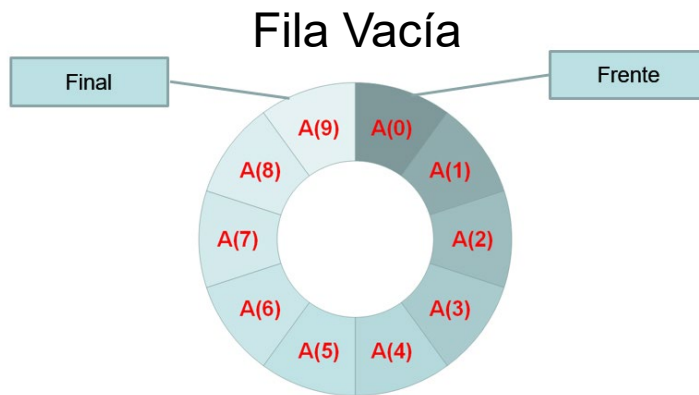
3) Defila(F) →



4) Defila(F) → Fila Vacía



# Implementación Fila con Arreglo Circular



# Una Implementación Fila con Arreglo en C

## // PRIVADO

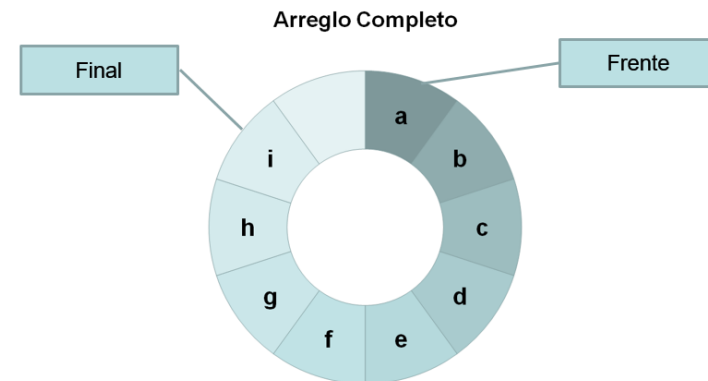
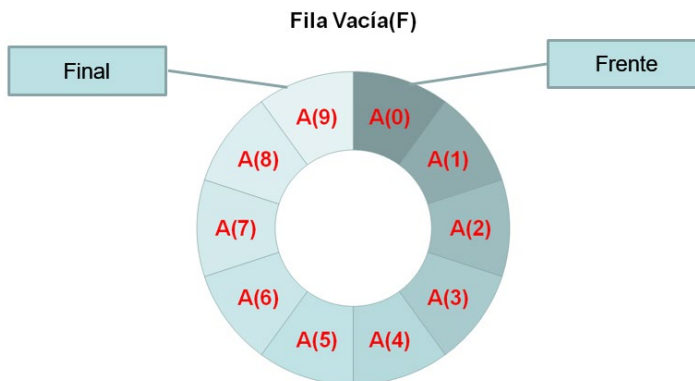
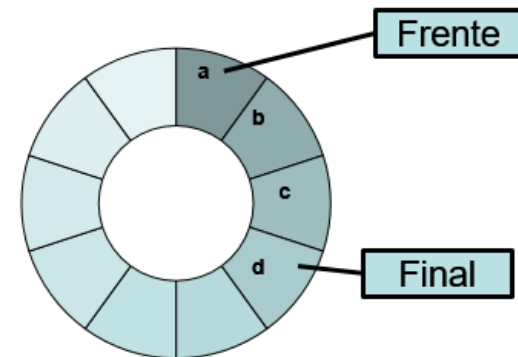
```
#define MAX 10          // o cualquier otro valor
#define indefinido -9999
typedef int item;        // o cualquier otro tipo
typedef struct {
    int frente,final;
    item elementos [MAX];
} tipofila;
int avanzar(int i);
```

## //PUBLICO

```
void filavacia(tipofila *f);
int esfilavacia(tipofila *f);
item frente (tipofila *f);
void defila(tipofila *f);
void enfila(tipofila *f, item x);
```

# Una Implementación Fila con Arreglo en C

```
int avanzar(int i)
{   if (i == MAX-1)
        return(0);
    else
        return(i+1); }
```



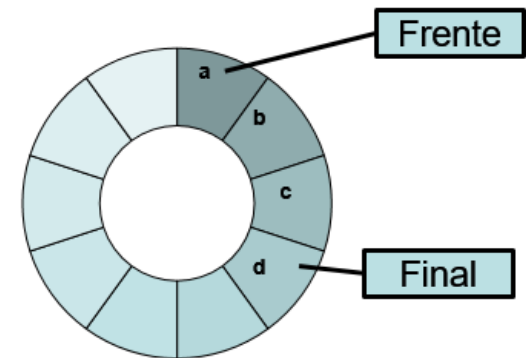
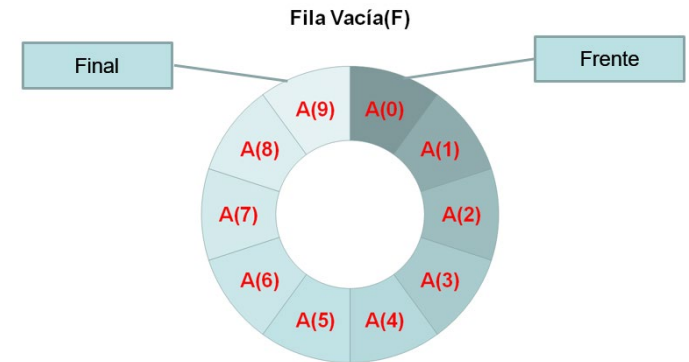
# Una Implementación Fila con Arreglo en C

```
void filavacia (tipofila *f)
{   f->frente = 0;           f->final = MAX-1; }
```

```
int esfilavacia (tipofila *f)
{   return( avanzar(f->final) == f->frente); }
```

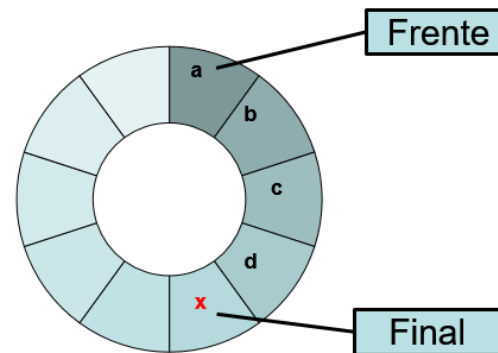
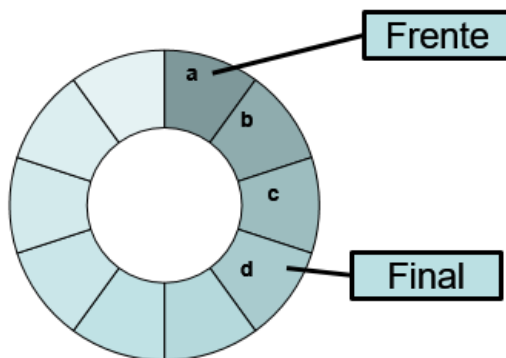
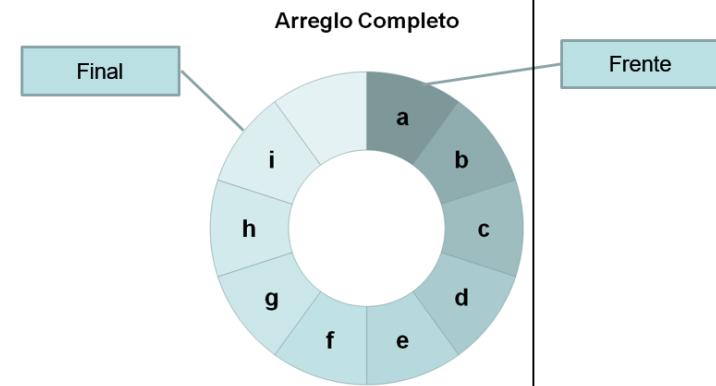
```
item frente (tipofila *f)
{   if( esfilavacia(f) )
        return (indefinido)
    else
        return(f->elementos[f->frente]);}
```

```
void defila (tipofila *f)
{   if ( !esfilavacia(f) ) f->frente = avanzar(f->frente); }
```



# Una Implementación Fila con Arreglo en C

```
void enfila (tipofila *f, item x)
{
    int uno;
    uno = avanzar(f->final);
    if( ! (avanzar(uno) == f->frente))
        { f->final =avanzar(f->final);
          f->elementos[f->final] = x;}
}
```



**// NOTA: todas las operaciones son  $O(1)$  en esta implementación.**