Algoritmos
Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán
2024

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

FILA-QUEUE

FILA

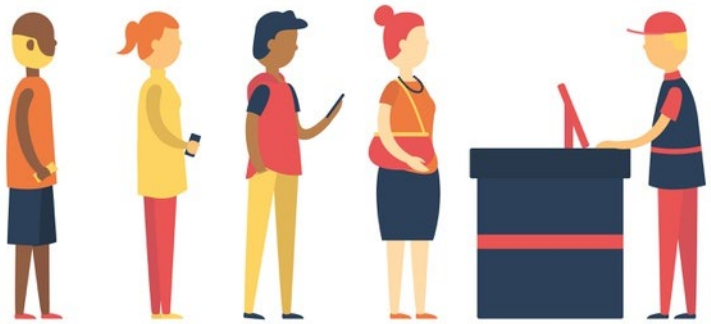
Una fila es una lista ordenada de cero o mas objetos de ***un mismo tipo*** que puede crecer por un extremo y decrecer por el otro de de sus extremos. Los extremos se llaman *final* y *frente* respectivamente.

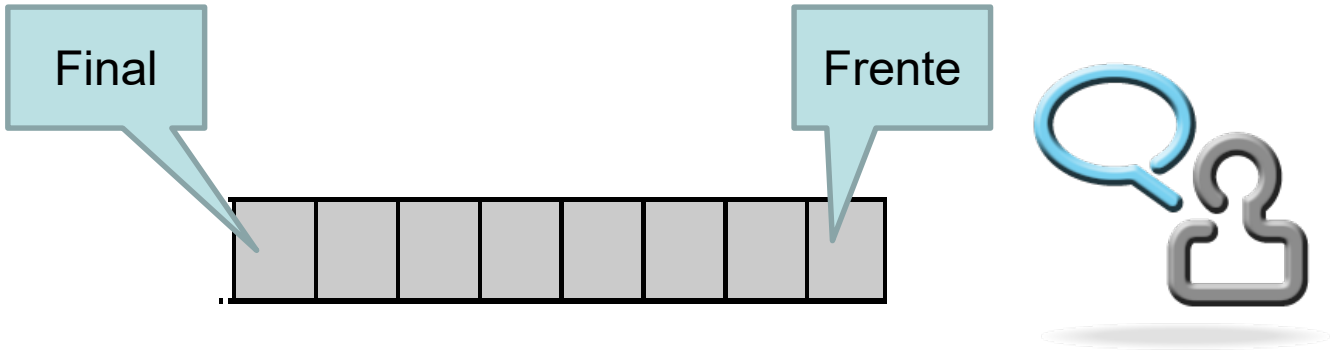
Una fila se llama también una estructura de tipo **FIFO** (*first in, first out*), porque posee la siguiente propiedad: el primer elemento que llega a una fila es el primero en ser sacado.

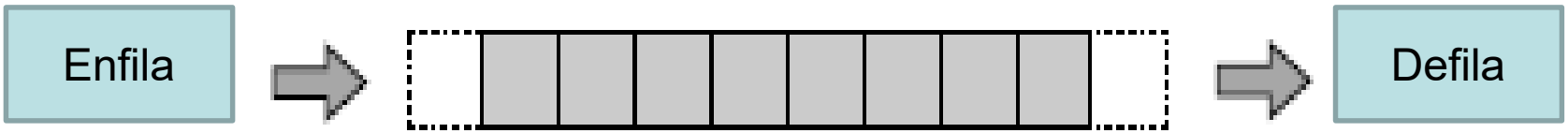
Las filas tienen un orden lineal que es el "*orden de llegada*".

Tienen muchas aplicaciones en sistemas operativos y simulaciones.

FILA







FILA(ITEM)
Especificación Algebraica

OPERACIONES

A) Sintaxis:

FILAVACIA **:** → FILA
ESFILAVACIA **:** FILA → BOOL
FRENTE **:** FILA → ITEM U {indefinido}
ENFILA **:** FILA X ITEM → FILA
DEFILA **:** FILA → FILA

FILA(ITEM)
Especificación Algebraica

B) Semántica: Para todo q $\in$ FILA , $\forall$ i $\in$ ITEM

ESFILAVACIA(FILAVACIA) $\equiv$ TRUE

ESFILAVACIA(ENFILA(q,i)) $\equiv$ FALSE

FRENTE(FILAVACIA) $\equiv$ indefinido

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

DEFILA(FILAVACIA) $\equiv$ FILAVACIA

DEFILA(ENFILA(q,i)) $\equiv$ si ESFILAVACIA(q) entonces
FILAVAVACIA
sino ENFILA(DEFILA(q),i)

FILA

Construir una Fila

FILA f : D C B A

FILAVACIA (f)

ENFILA(f, A **)**

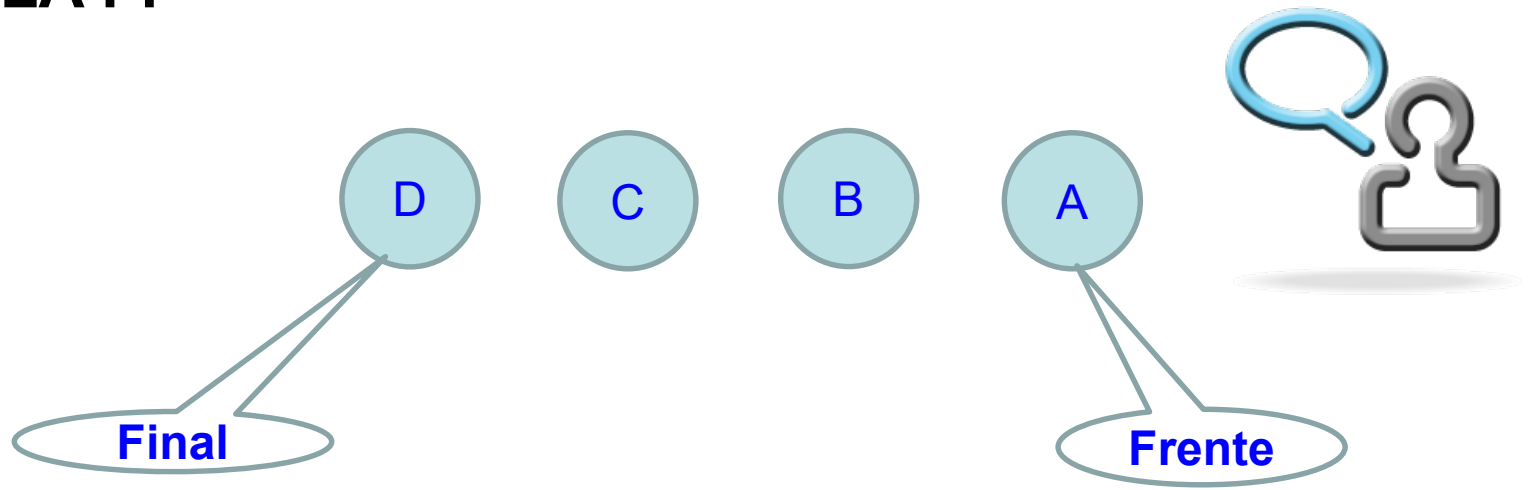
ENFILA(f, B **)**

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 → entero ≥ 0

Semántica: $\forall$ q $\in$ FILA, $\forall$ i $\in$ ITEM

LONGITUD (FILAVACIA) $\equiv$ 0

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

FILA
Aplicación: longitud de una Fila

2) Como usuario del ADT FILA, función recursiva

```
Funcion LONGITUD (f) :  FILA → entero ≥ 0
        Si ESFILAVACIA (f) entonces
                Retorna 0
        Sino
                Retorna 1 + LONGITUD(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 → entero ≥ 0
Auxiliar: h ∈ entero
h ← 0
Mientras NOT ESFILAVACIA(f) hacer
h ← 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→BOOL

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

PERTENECE (FILAVACIA,i) $\equiv$ FALSO

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

donde = es el operador 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,i) :  FILA x ITEM → BOOL
        Si ESFILAVACIA (f) entonces
           Retorna FALSO
        Sino
            Retorna
                 FRENTE(f)=i OR PERTENECE (DEFILA(f),i)
 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,i) **:** FILA x ITEM → BOOL
Auxiliar: esta ∈ BOOL

```
esta←FALSO
Mientras NOT ESFILAVACIA(f) AND NOT esta hacer
        esta ← FRENTE(f) = i
        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,i) **:** FILA x ITEM → BOOL

Mientras NOT ESFILAVACIA(f) AND NOT FRENTE(f) **=** i
hacer

DEFILA(f)

Retorna NOT ESFILAVACIA(f)

Fin

FILA
Aplicación: igualdad de Filas

1) Dentro de la Especificación Algebraica

Sintaxis:

IGUALF : FILA x FILA → BOOL

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

IGUALF(FILAVACIA, FILAVACIA) $\equiv$ TRUE

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

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

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

donde = es el operador 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 → 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 → 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 (4 axiomas)

Sintaxis:

CONCAT : FILA x FILA → FILA

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

CONCAT(FILAVACIA,FILAVACIA) $\equiv$ FILAVACIA

CONCAT(ENFILA(q,j), FILAVACIA) $\equiv$ ENFILA(q,j)

CONCAT(FILAVACIA, ENFILA(p,i)) $\equiv$ ENFILA (p,i)

CONCAT(ENFILA(q,j), ENFILA(p,i)) $\equiv$

ENFILA (CONCAT(ENFILA(q,j),p),i)

FILA
Aplicación: concatenar Filas

2) Dentro de la Especificación Algebraica (2 axiomas)

Sintaxis:

CONCAT : FILA x FILA → FILA

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

CONCAT(q, FILAVACIA) $\equiv$ q

CONCAT(q, ENFILA(p,i)) $\equiv$ ENFILA (CONCAT(q,p),i)

FILA
Aplicación: concatenar Filas

3) Como usuario del ADT FILA, función recursiva

Funcion **CONCAT** (f1,f2) : FILA x FILA → FILA

```
    Si ESFILAVACIA(f2) entonces
        Retorna f1
    Sino
        ENFILA (f1, FRENTE(f2))
        DEFILA (f2)
        Retorna CONCAT (f1,f2)
Fin
```

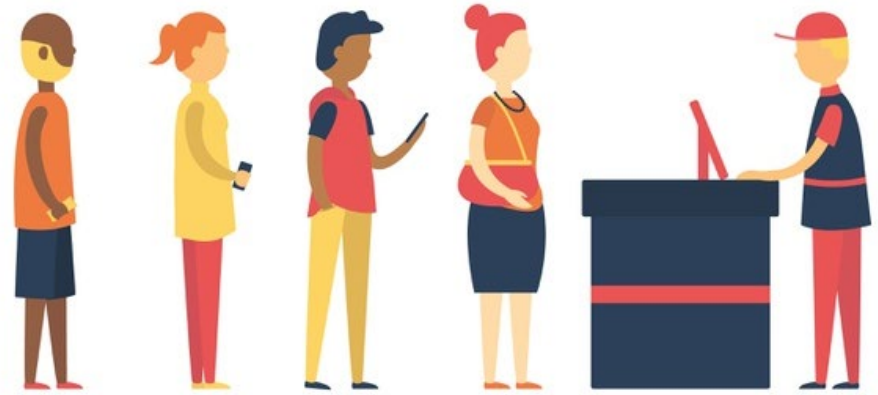
FILA
Aplicación: concatenar Filas

4) Como usuario del ADT FILA, función iterativa

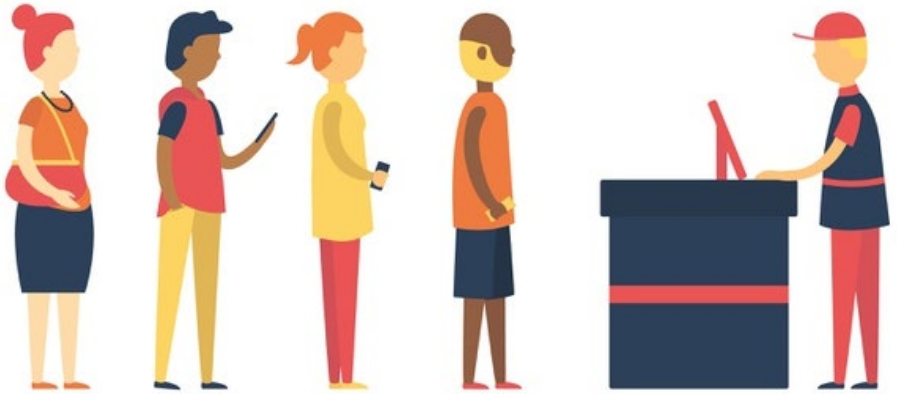
```
Funcion CONCAT (f1,f2) : FILA x FILA → FILA
    Mientras NOT ESFILAVACIA(f2)  hacer
                        ENFILA(f1,FRENTE(f2))
                        DEFILA(f2)
    Retorna f1
Fin
```

FILA

Aplicación: invertir la Fila



Fila original



Fila invertida

FILA
Aplicación: invertir la Fila

1) Dentro de la Especificación Algebraica

Sintaxis:

INVERTIR **:** FILA→FILA

Semántica: $\forall$ q $\in$ FILA, $\forall$ i $\in$ ITEM

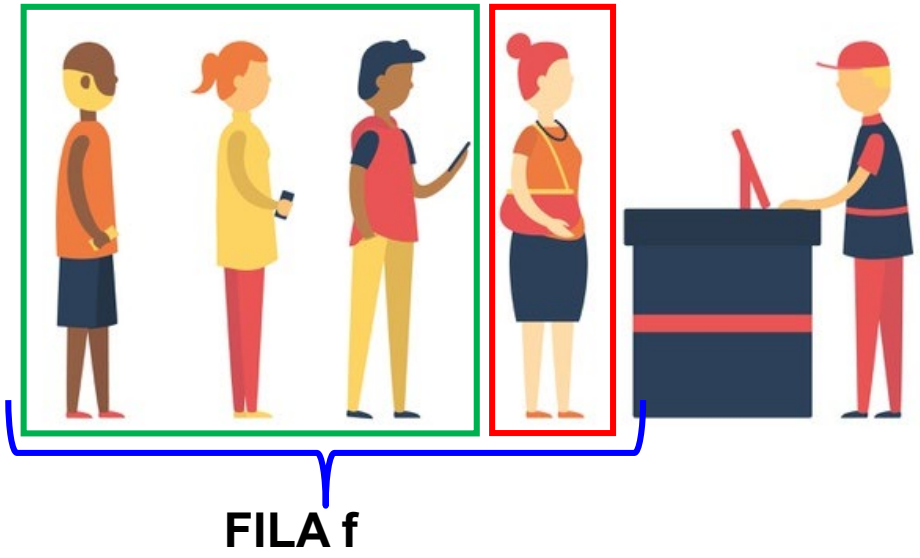
INVERTIR (FILAVACIA) $\equiv$ FILAVACIA

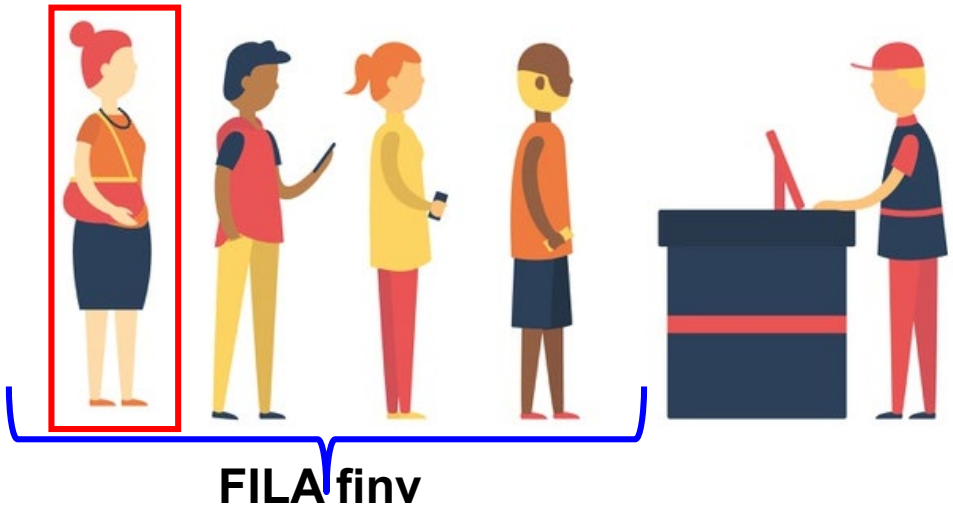
INVERTIR(ENFILA(q,i)) $\equiv$

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

FILA

Invertir la Fila - algoritmo recursivo





FILA
Aplicación: invertir la Fila

2) Como usuario del ADT FILA, algoritmo recursivo

Algoritmo INVERTIR (f,finv)

ENTRADA : f ∈ FILA

SALIDA: finv ∈ FILA, llega inicializada en FILAVACIA

AUXILIAR: itemaux ∈ ITEM

Si NOT ESFILAVACIA(f) entonces

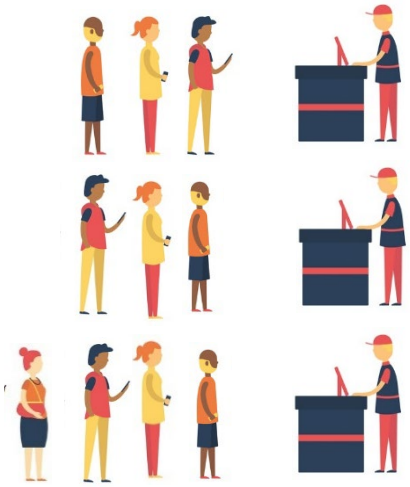
itemaux ← FRENTE(f)

DEFILA(f)

INVERTIR(f,finv)

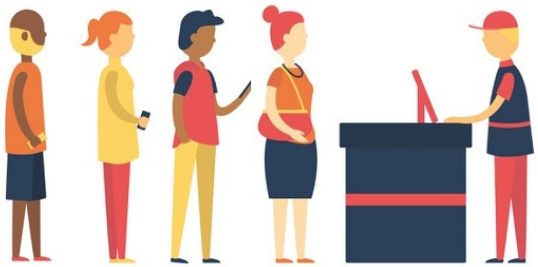
ENFILA(finv, itemaux)

Fin



FILA

Invertir la Fila - algoritmo iterativo



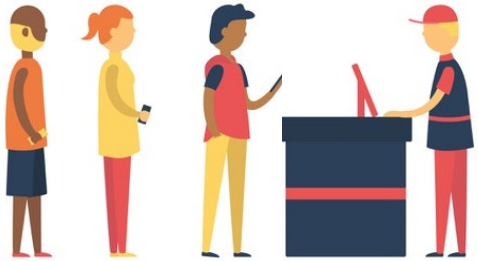
FILA f

FILA finv

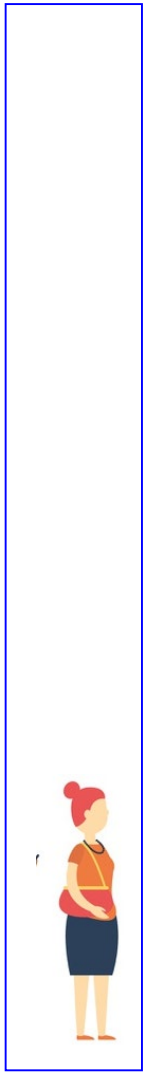
Pilaux

FILA

Invertir la Fila - algoritmo iterativo



FILA f

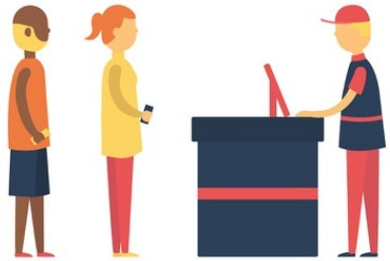


FILA finv

Pilaux

FILA

Invertir la Fila - algoritmo iterativo



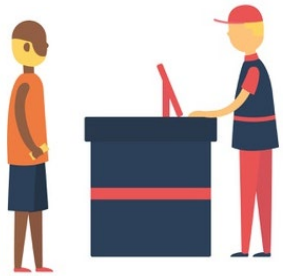
FILA f

FILA finv

Pilaux

FILA

Invertir la Fila - algoritmo iterativo



FILA f

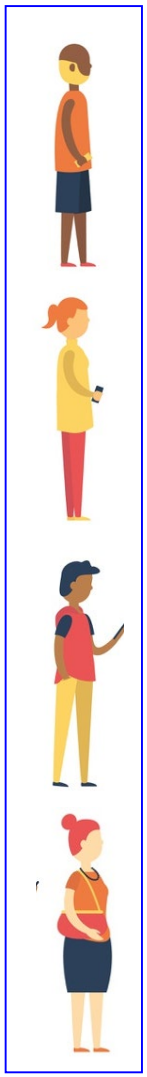
Pilaux

FILA finv

FILA

Invertir la Fila - algoritmo iterativo

FILA f



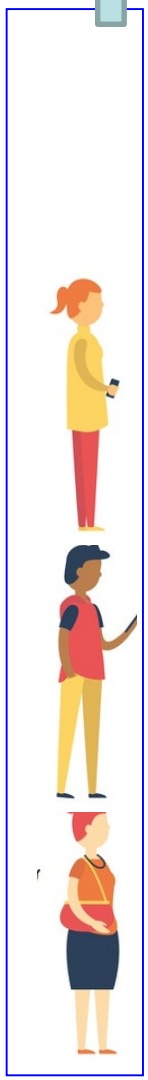
Pilaux

FILA finv

FILA

Invertir la Fila - algoritmo iterativo

FILA f



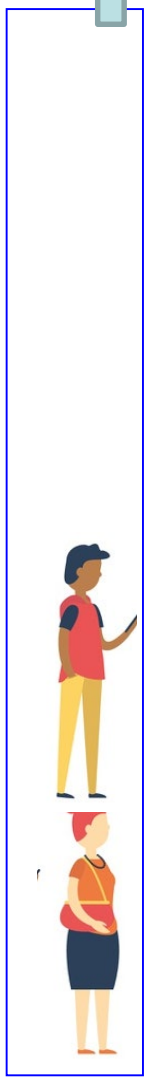
Pilaux

FILA finv

FILA

Invertir la Fila - algoritmo iterativo

FILA f



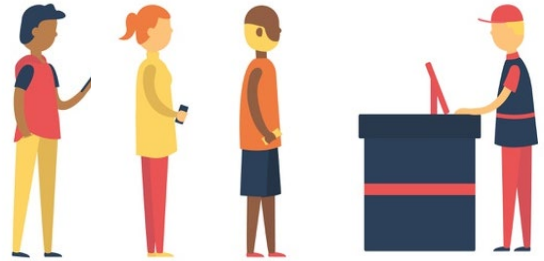
FILA finv

Pilaux

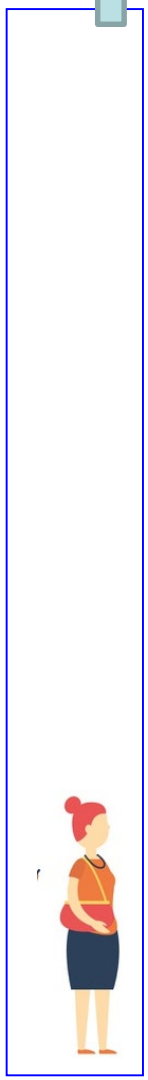
FILA

Invertir la Fila - algoritmo iterativo

FILA f



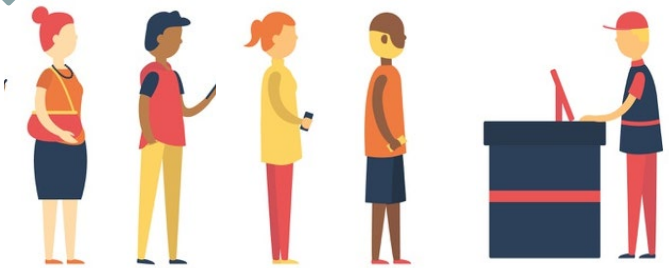
FILA finv



Pilaux

FILA

Invertir la Fila - algoritmo iterativo



FILA f

FILA finv

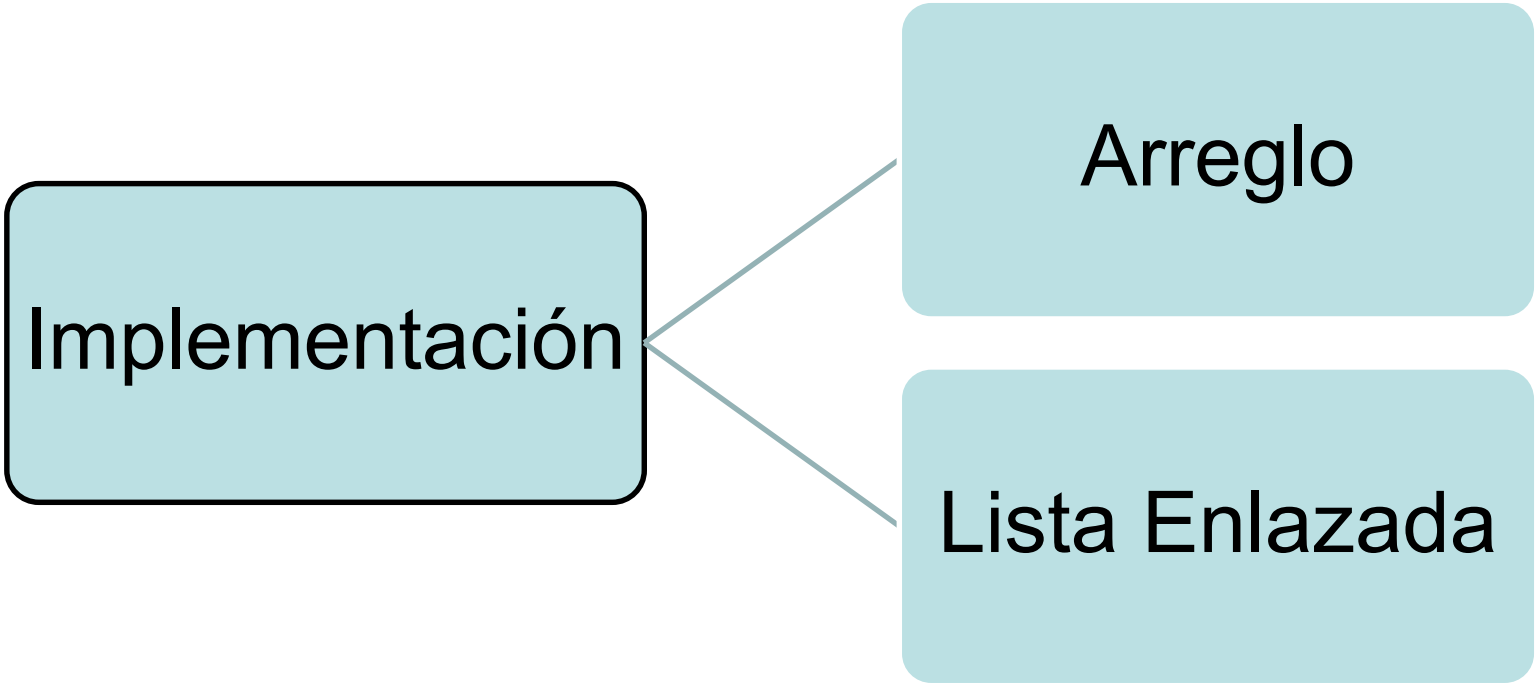
Pilaux

FILA
Aplicación: invertir la Fila

3) Como usuario del ADT FILA, algoritmo iterativo

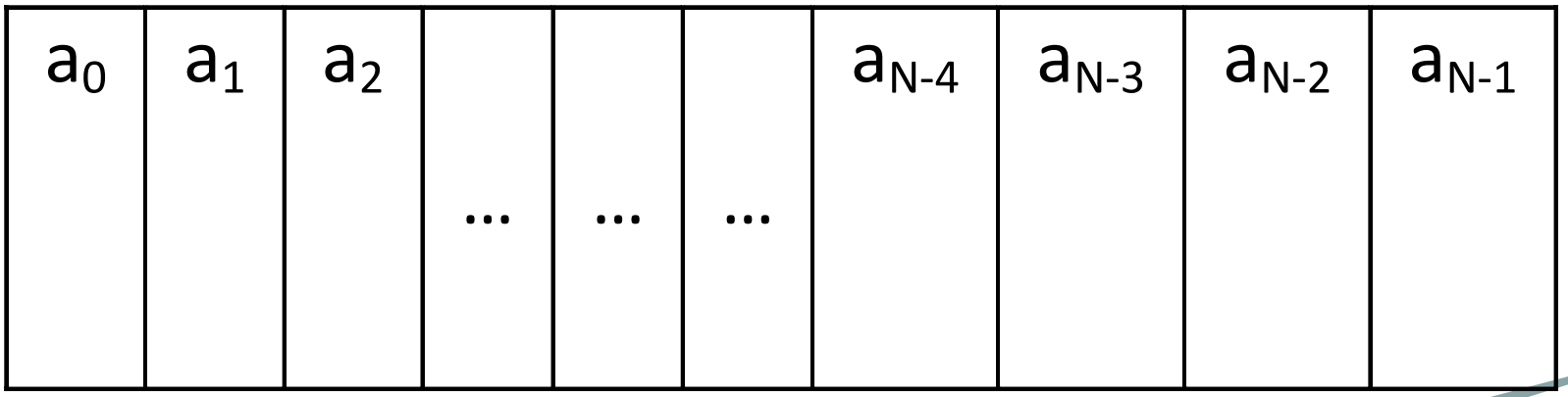
```
Algoritmo INVERTIR(f,finv)
ENTRADA : f ∈ FILA
SALIDA: finv ∈ FILA
AUXILIAR: pilaux ∈ PILA(ITEM)
    PILAVACIA(pilaux)
    Mientras NOT ESFILAVACIA(f) hacer
            PUSH (pilaux, FRENTE(f))
            DEFILA(f)
    FILAVACIA(finv)
    MIENTRAS NOT ESPILAVACIA(pilaux) HACER
            ENFILA(finv, TOP(pilaux))
            POP(pilaux)
 Fin
```

FILA



Ventajas y Desventajas

Implementación Fila con arreglo

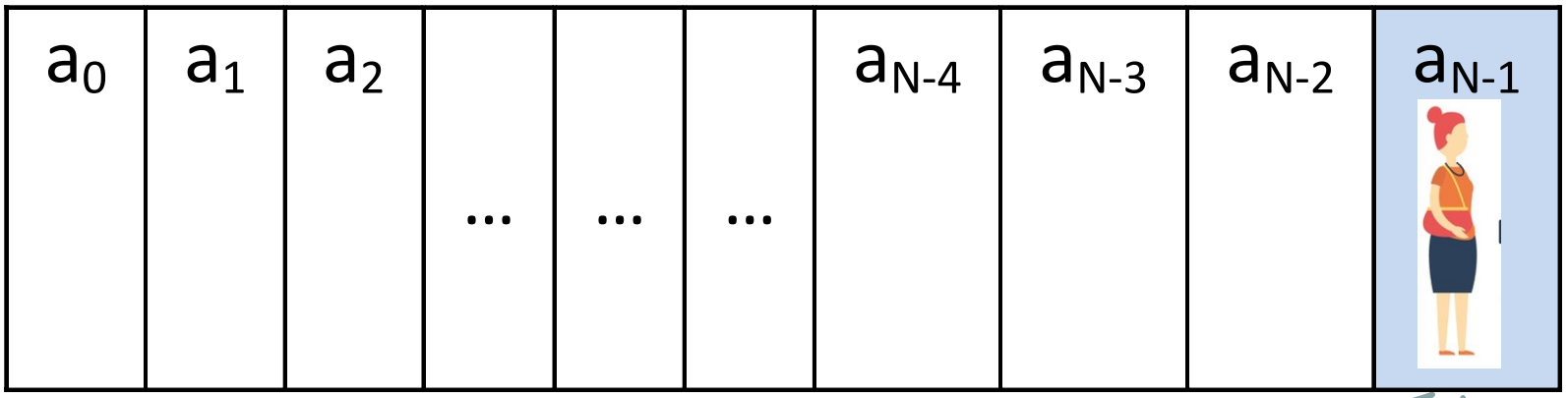


FILAVACIA

Final

Frente

Implementación Fila con arreglo

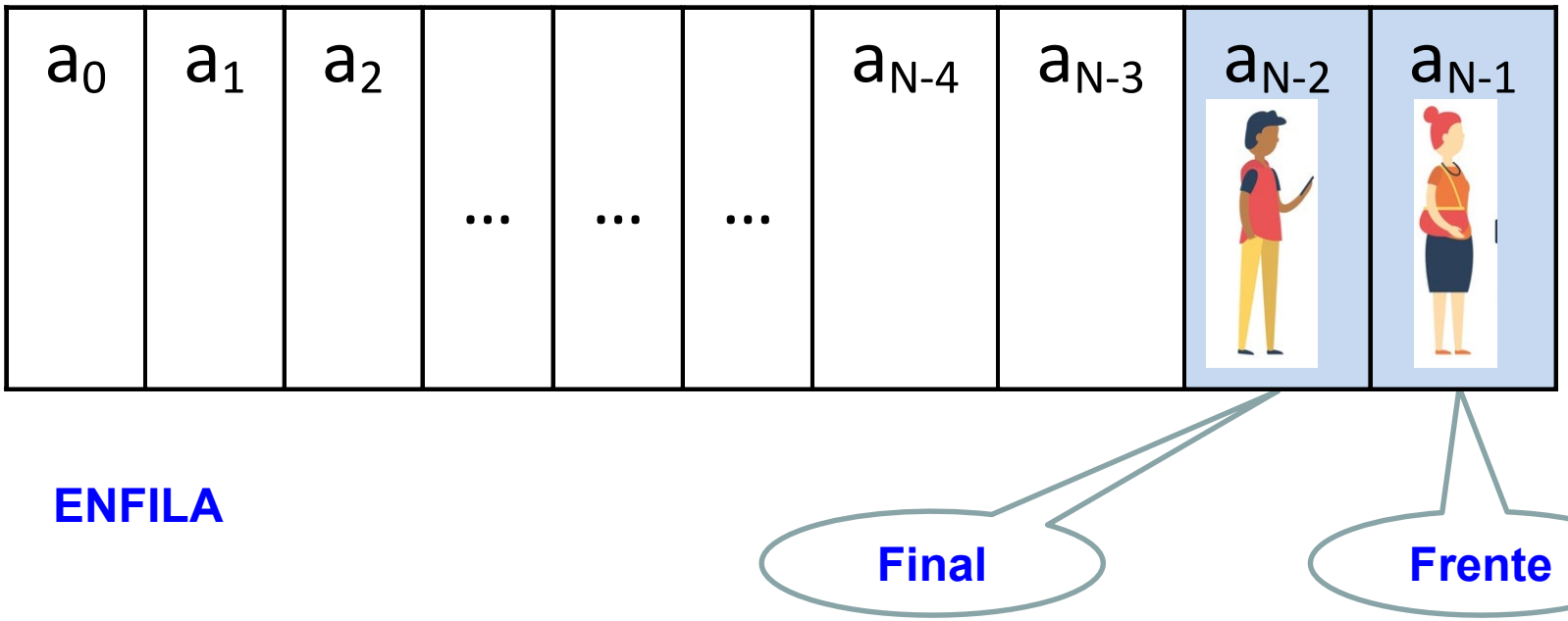


ENFILA

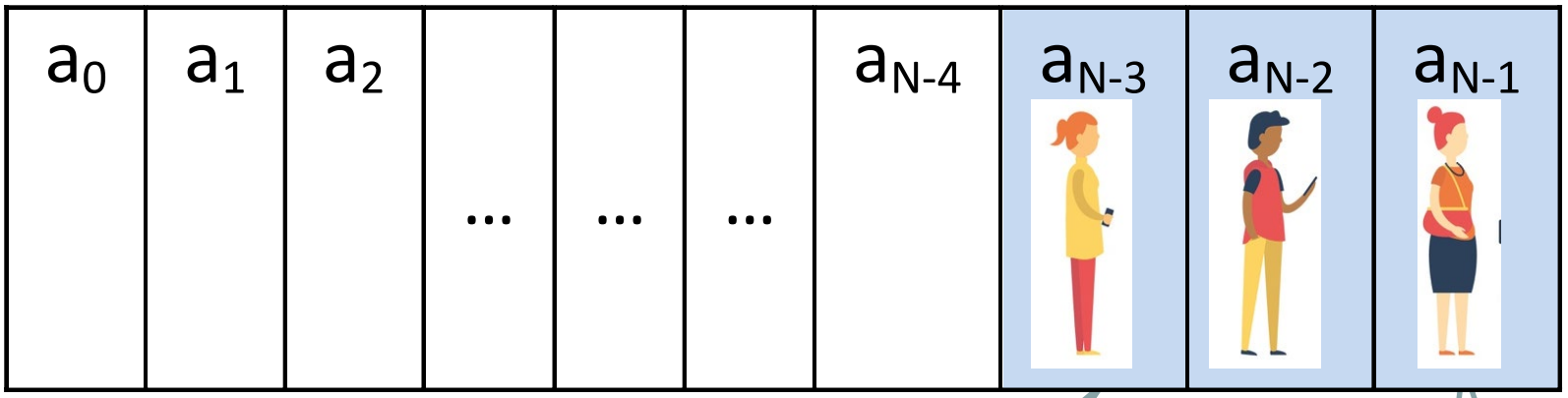
Final

Frente

Implementación Fila con arreglo



Implementación Fila con arreglo

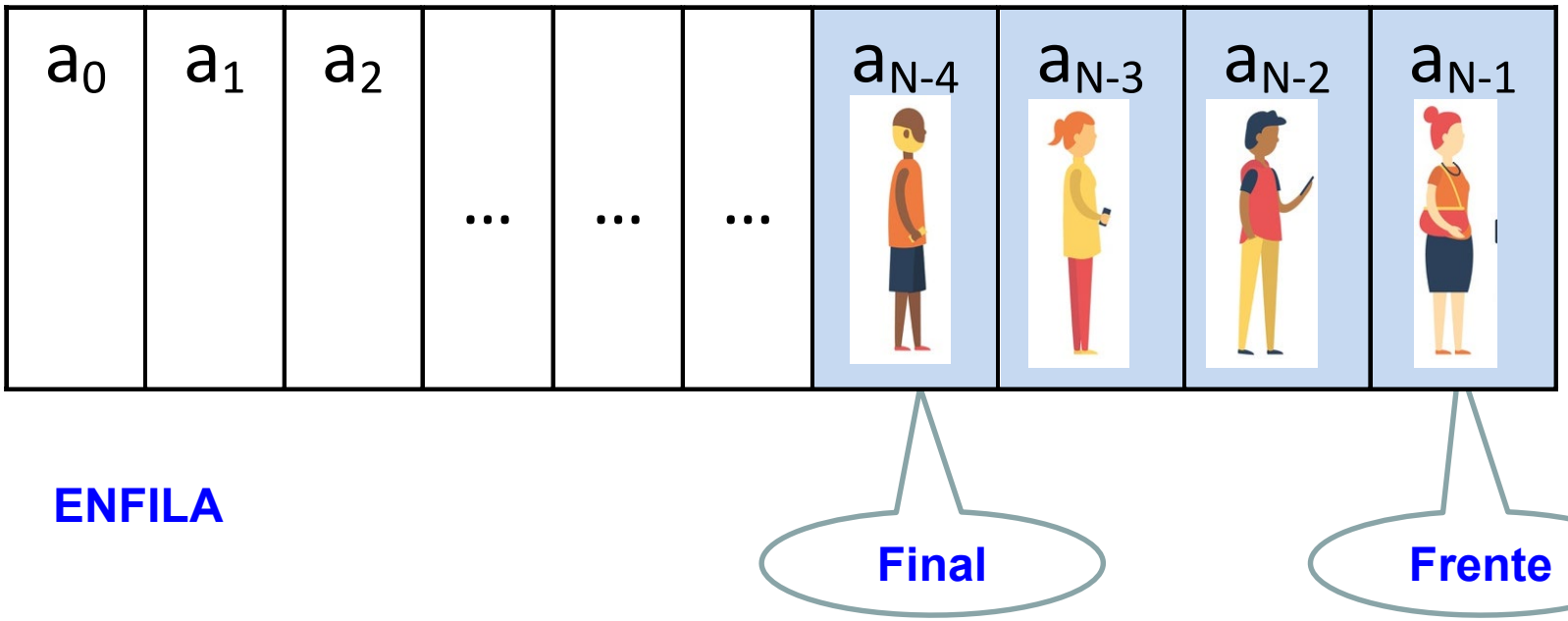


ENFILA

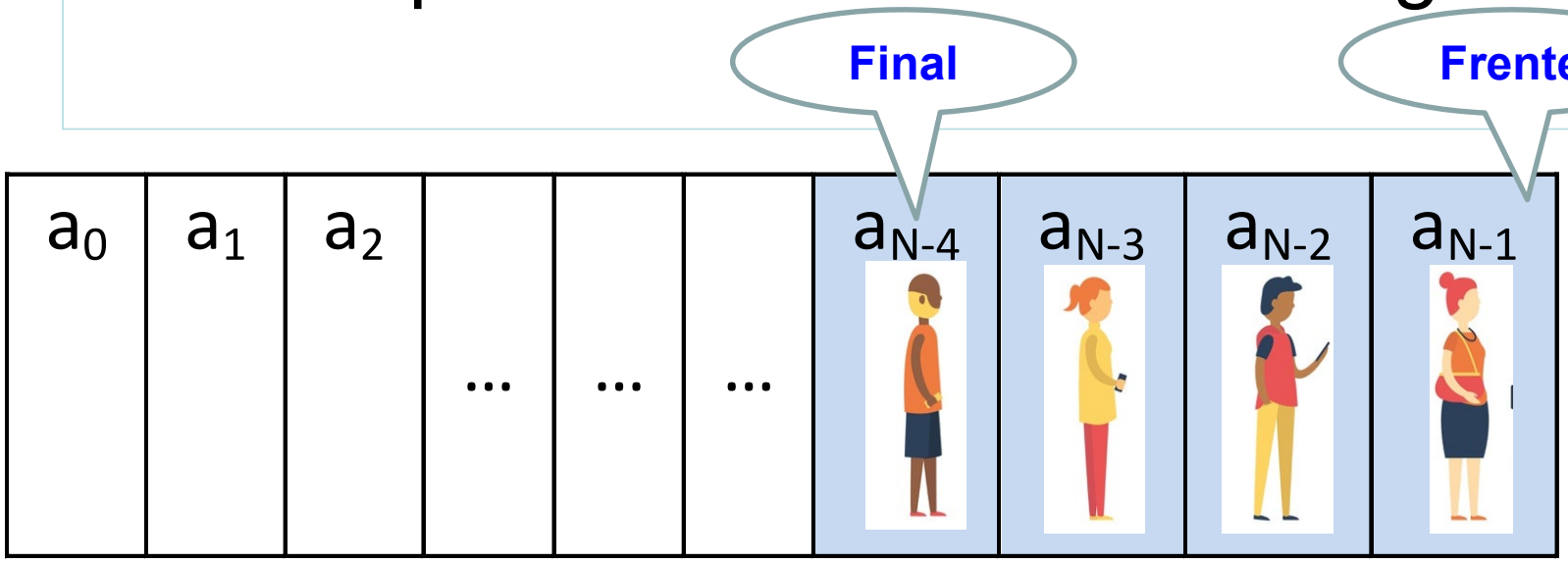
Final

Frente

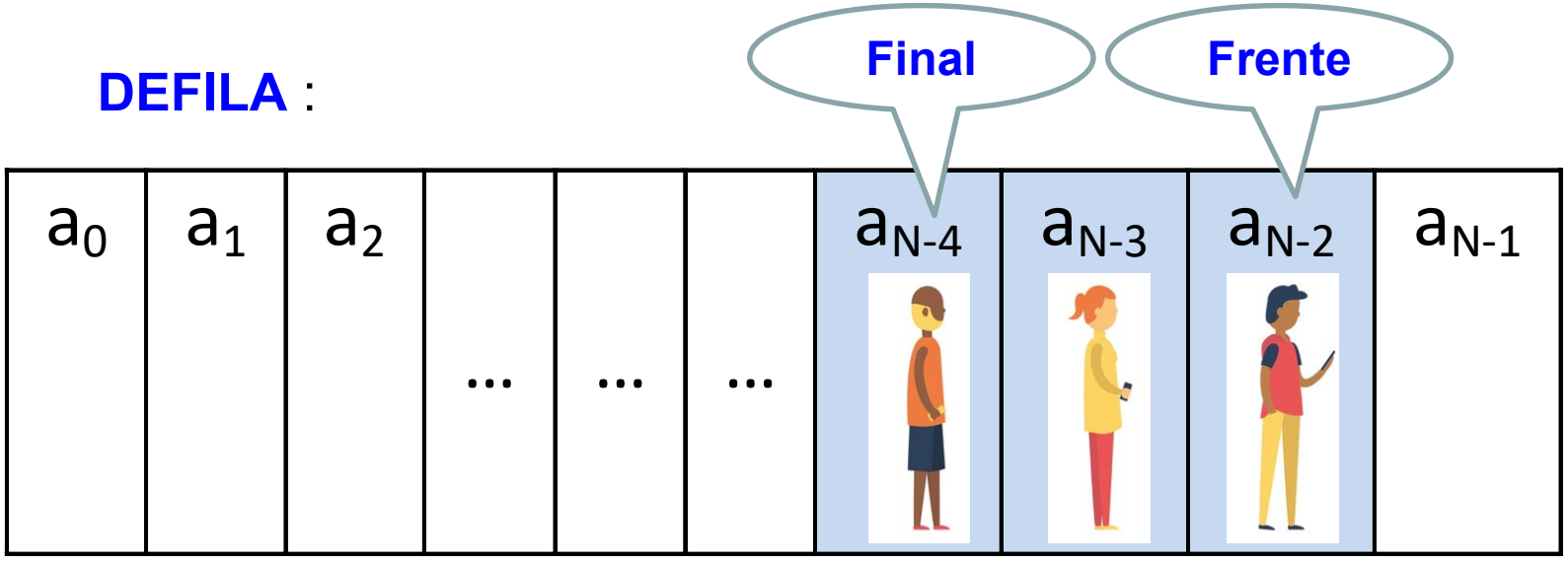
Implementación Fila con arreglo



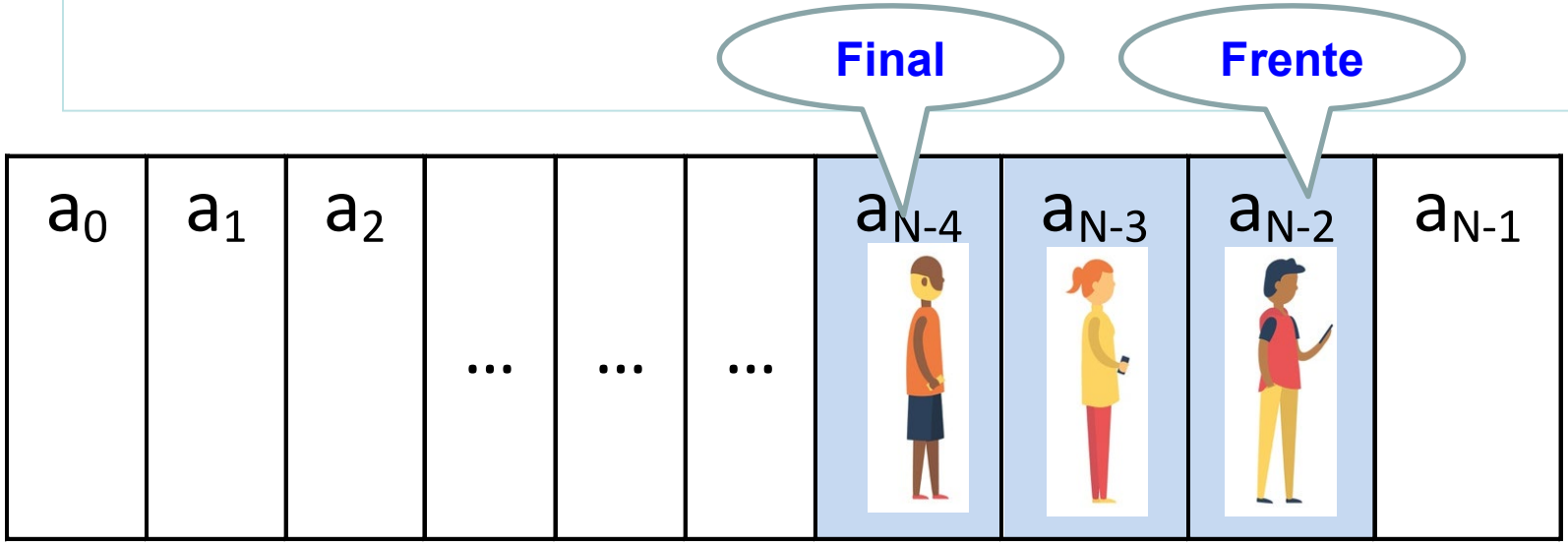
Implementación Fila con arreglo



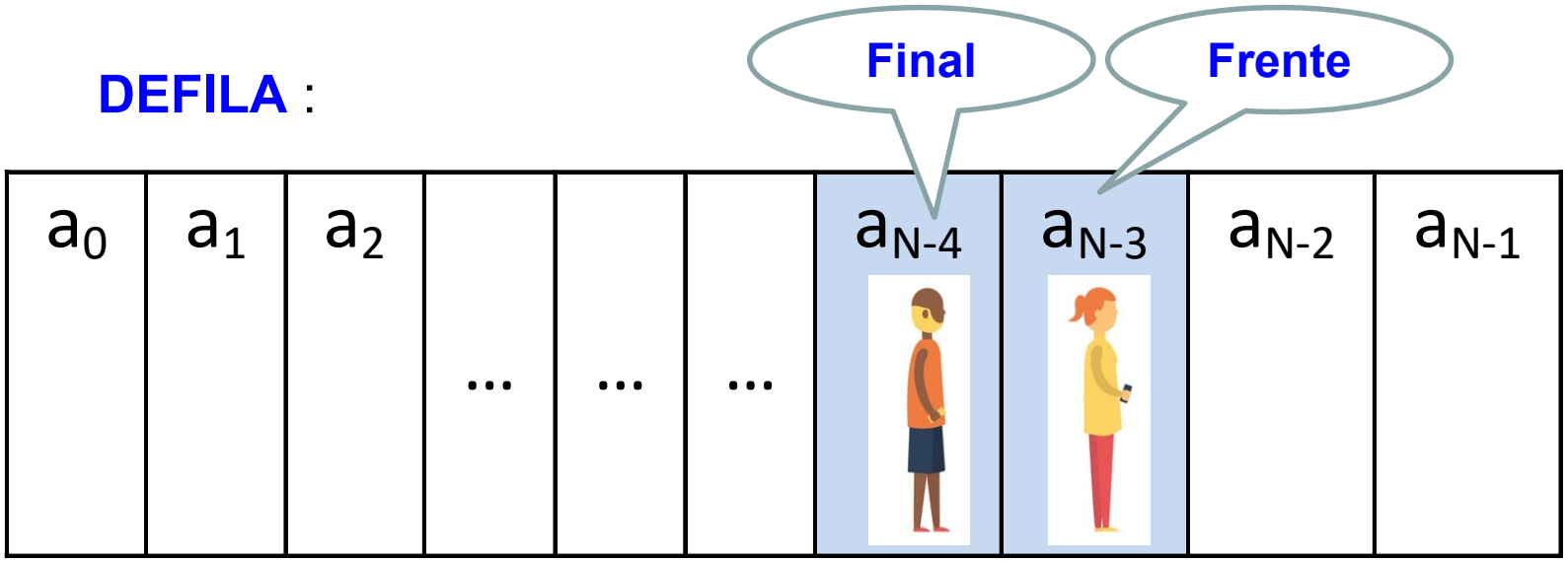
DEFILA :



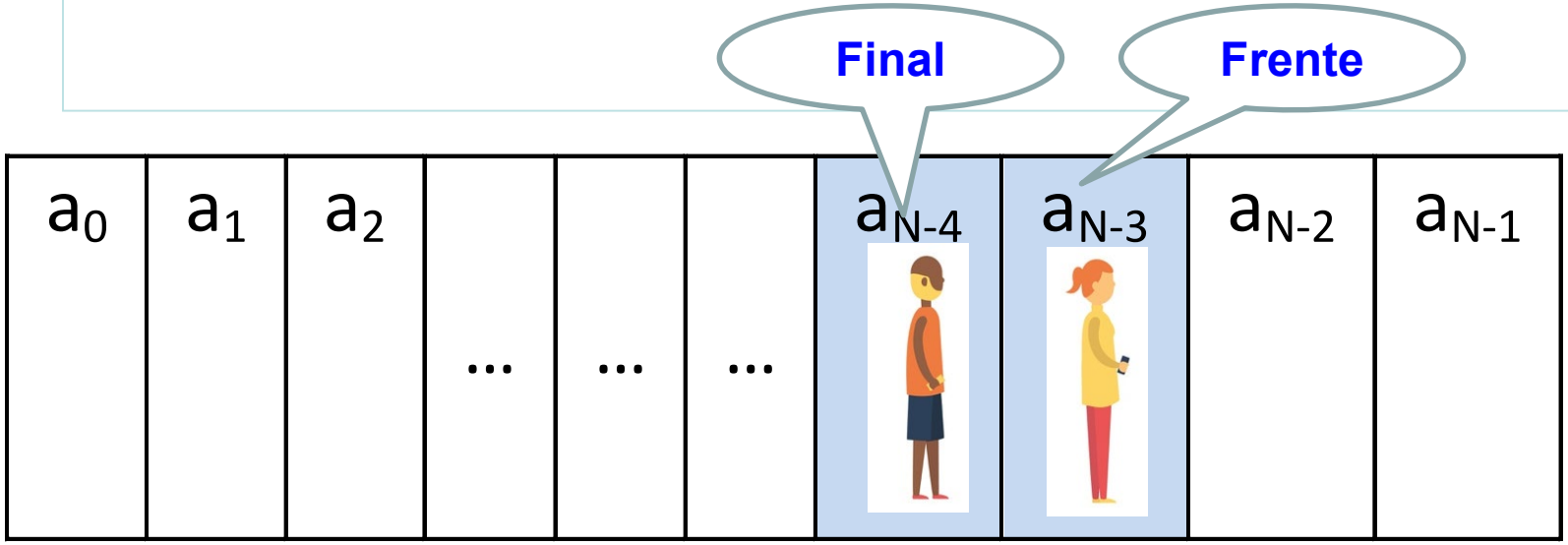
Implementación Fila con arreglo



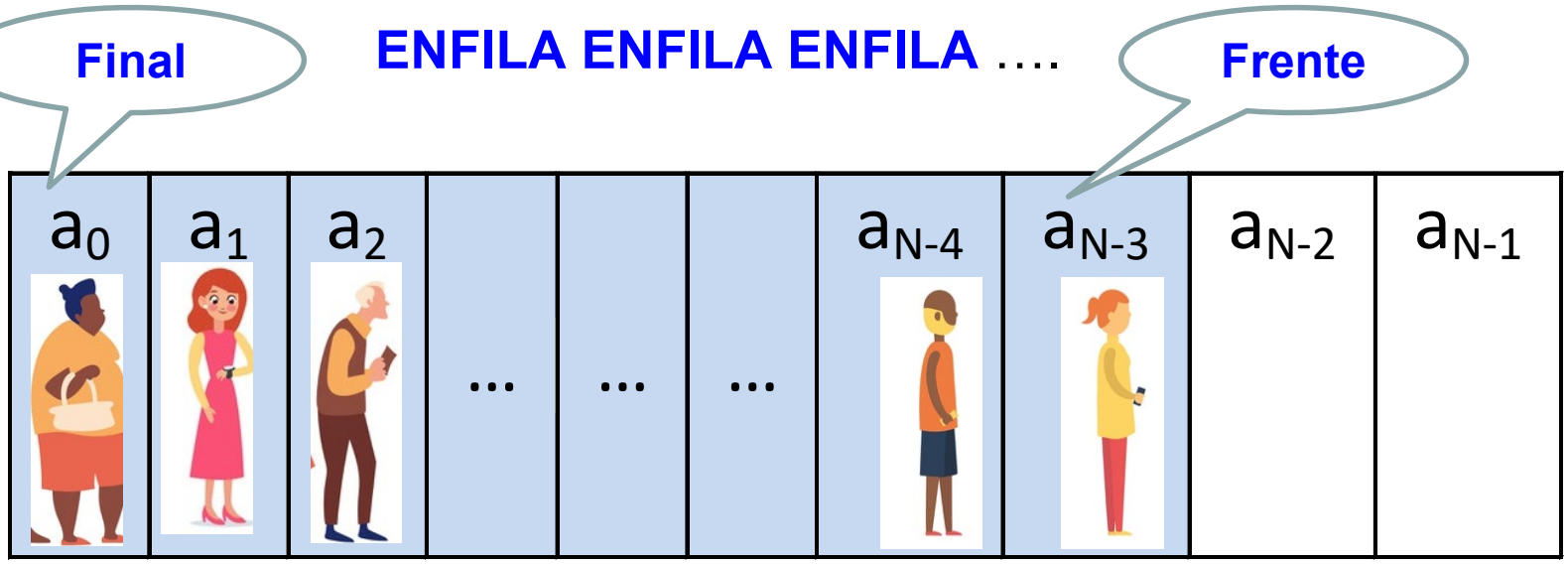
DEFILA :



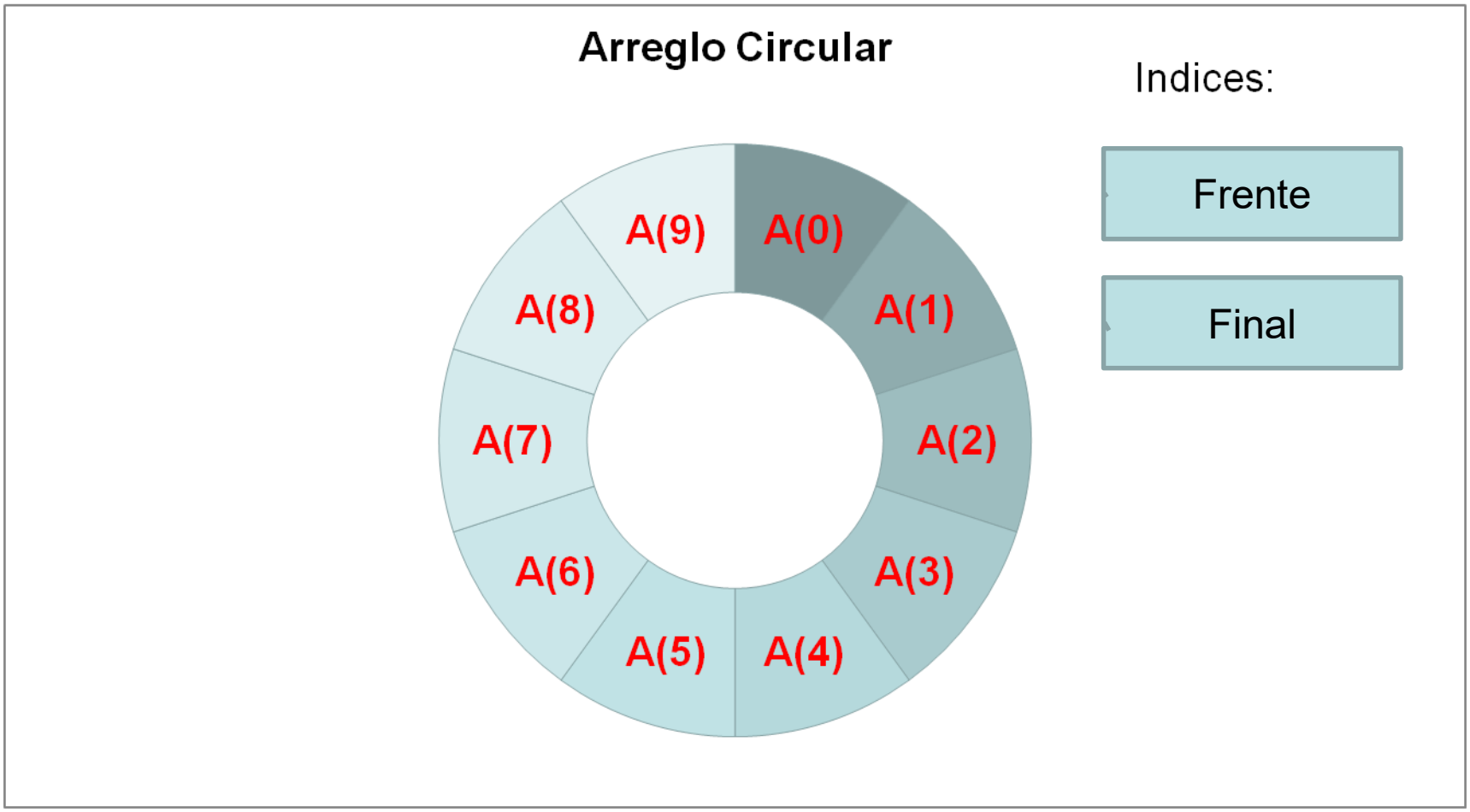
Implementación Fila con arreglo



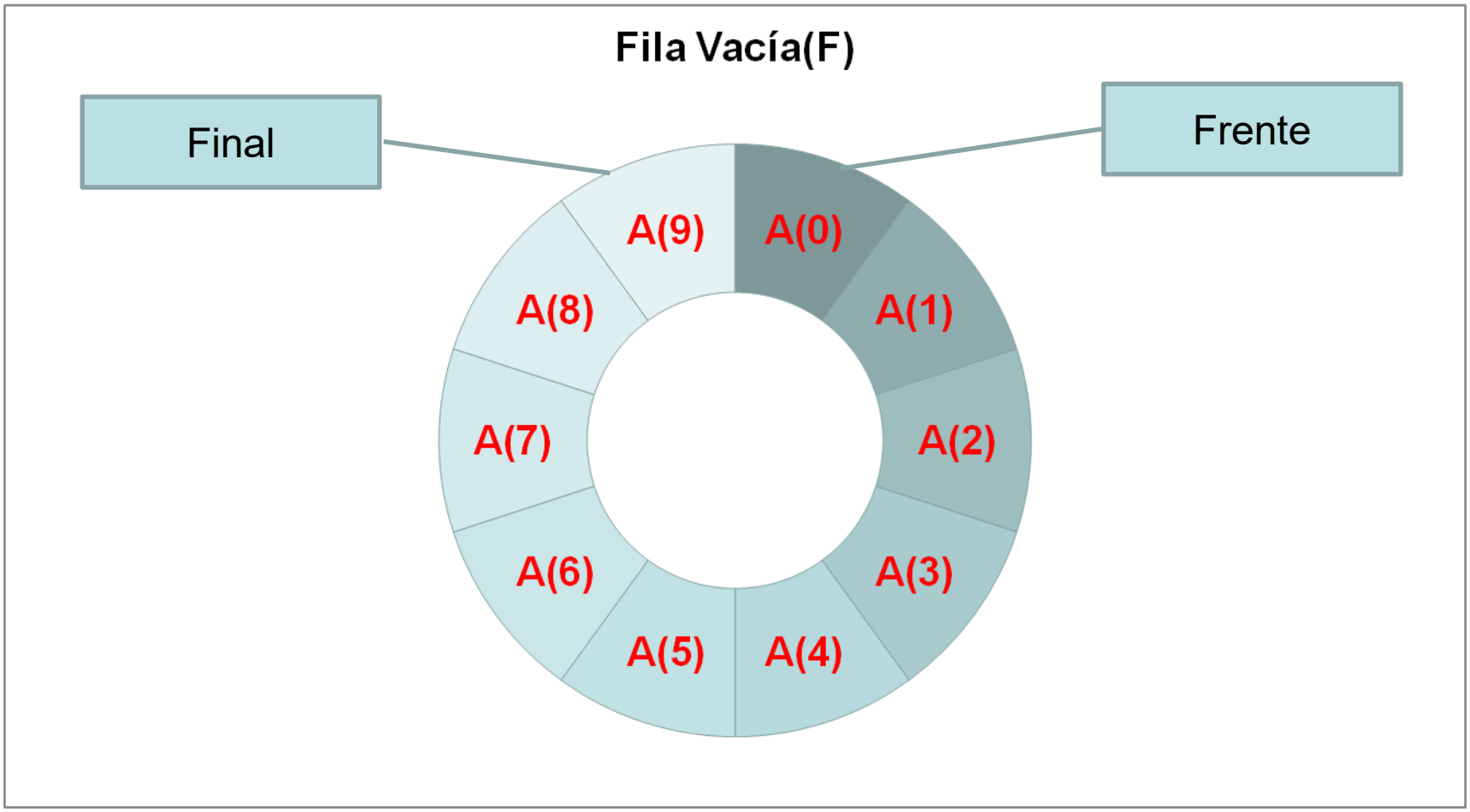
ENFILA ENFILA ENFILA ….



Implementación Fila con Arreglo Circular

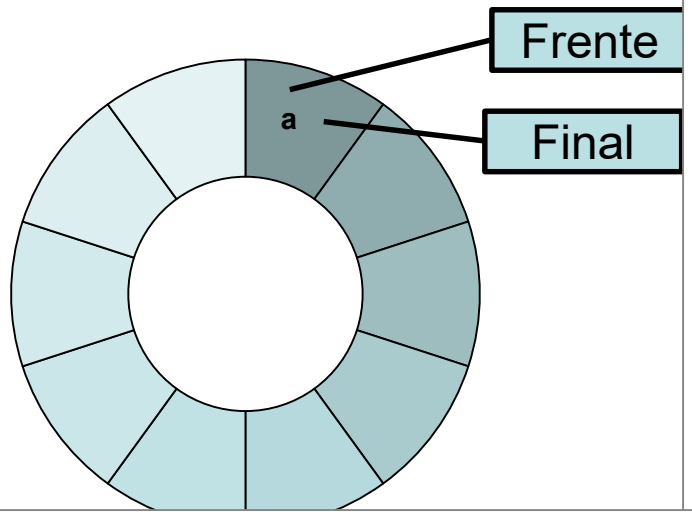


Implementación Fila con Arreglo Circular

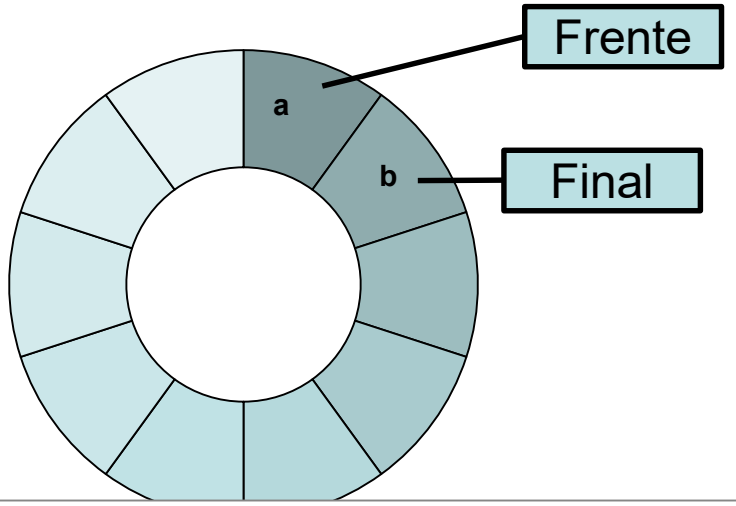


Implementación Fila con Arreglo Circular

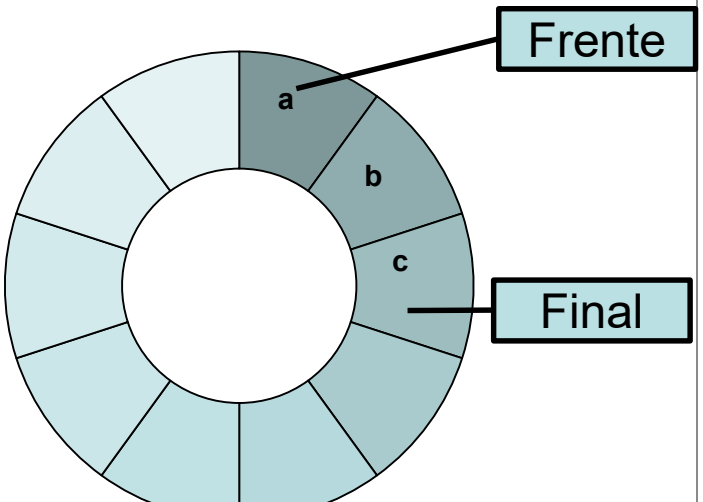
1)Enfila(F,a) ➔



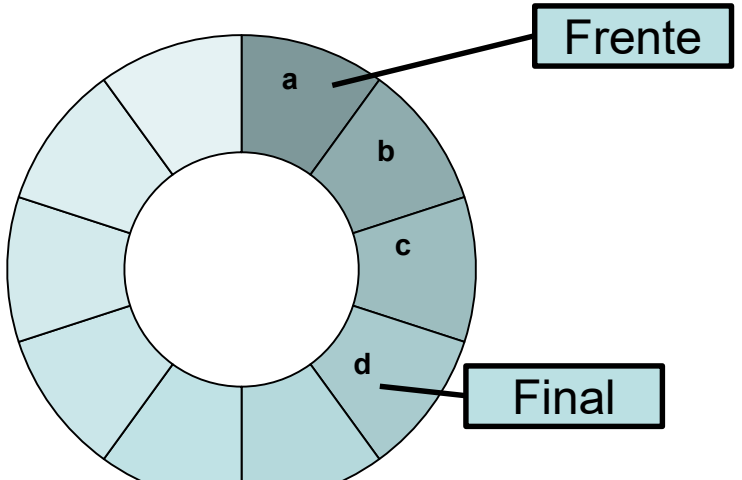
2)Enfila(F,b) ➔



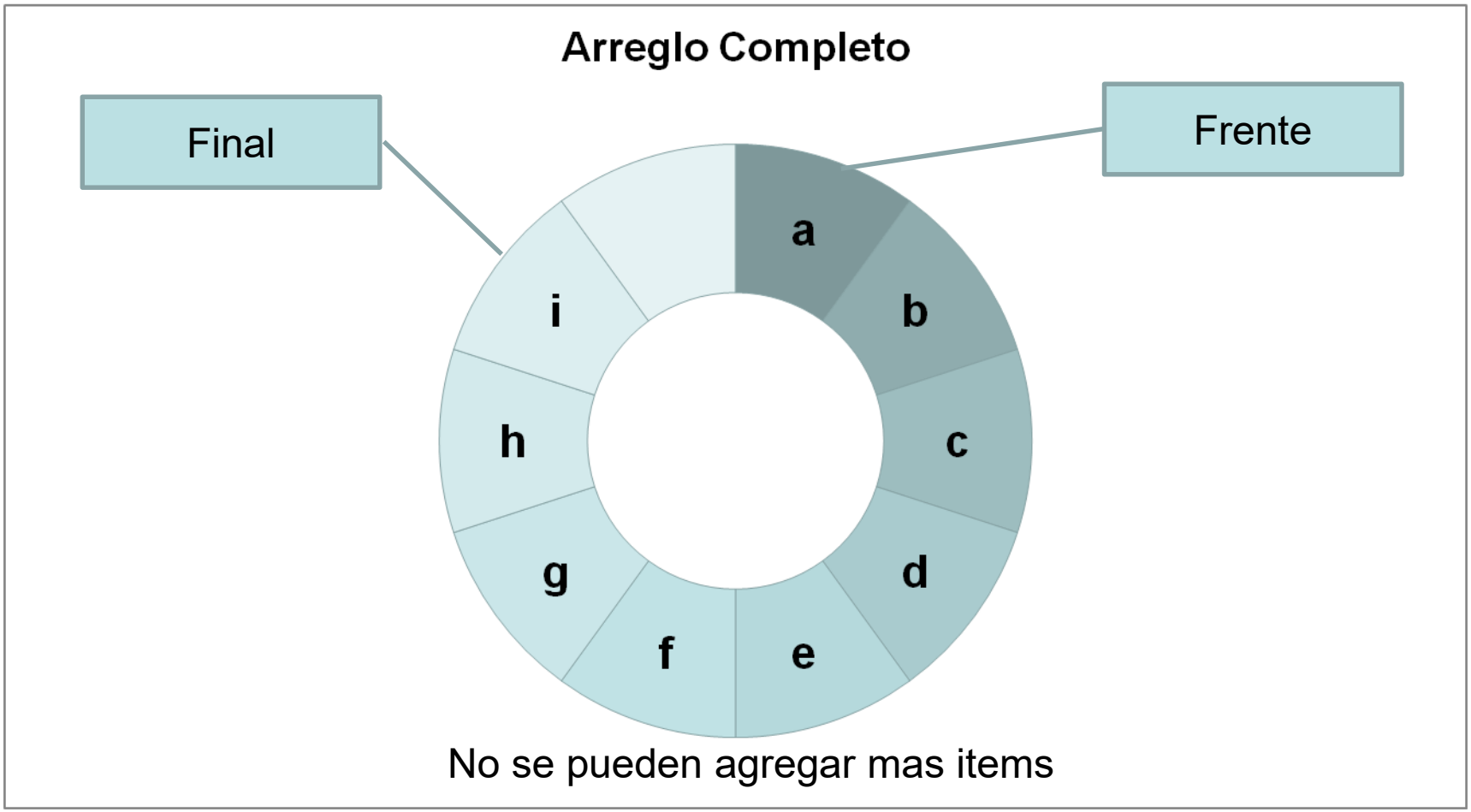
3)Enfila(F,c) ➔



4)Enfila(F,d) ➔

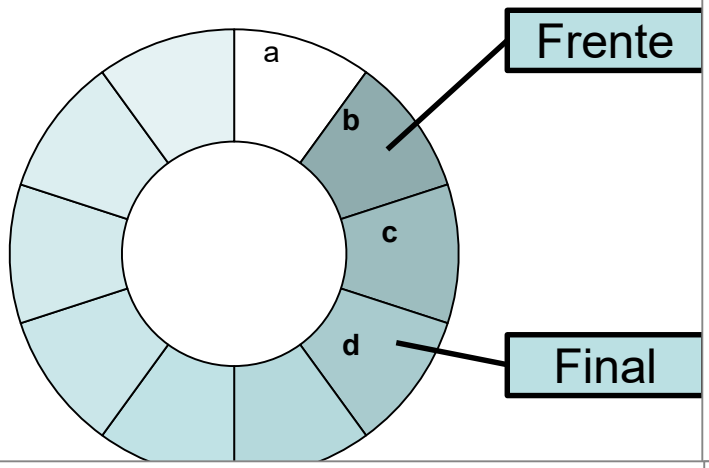


Implementación Fila con Arreglo Circular

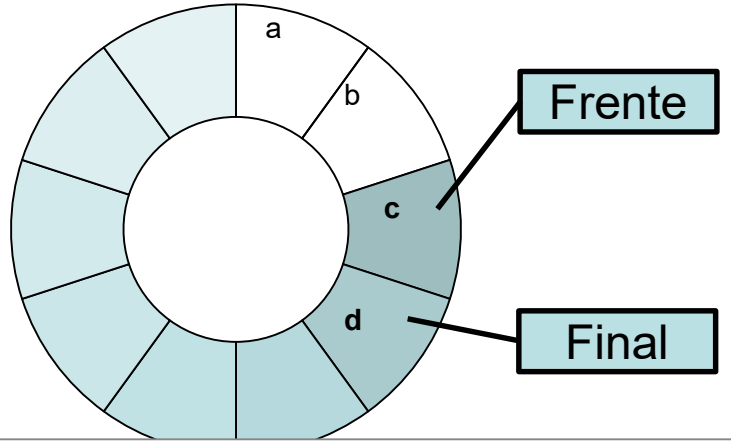


Implementación Fila con Arreglo Circular

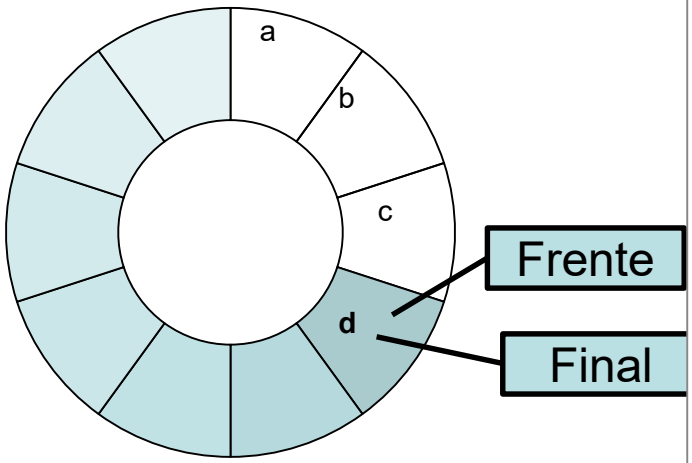
1)Defila(F) ➔



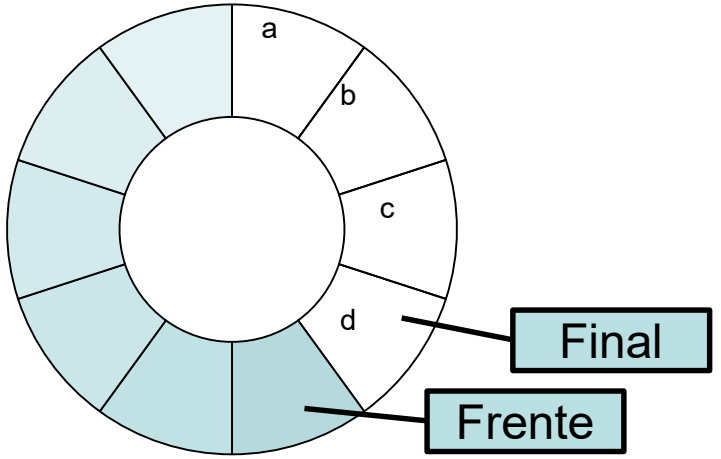
2)Defila(F) ➔



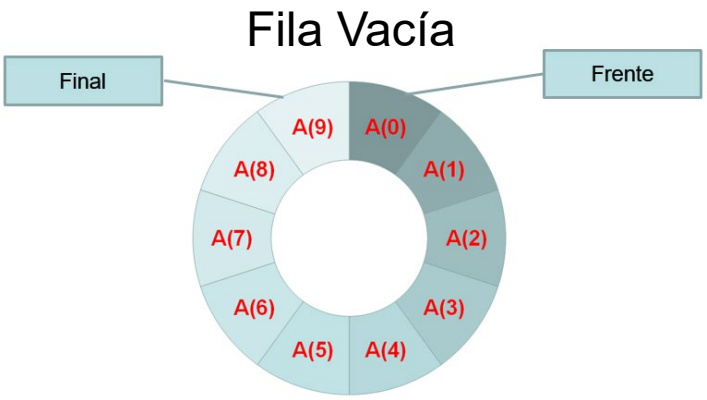
3)Defila(F) ➔

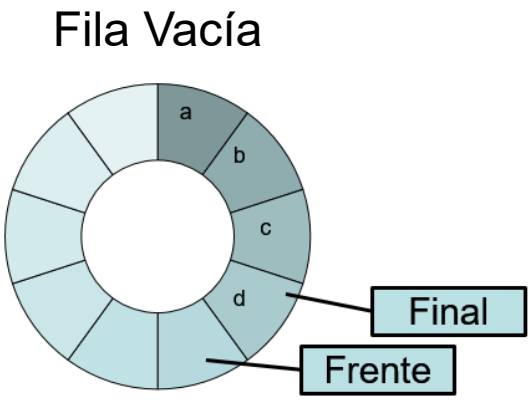


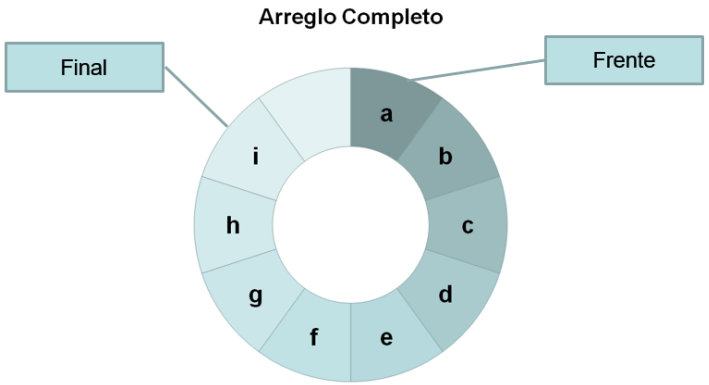
4)Defila(F) ➔ Fila Vacia



Implementación Fila con Arreglo Circular





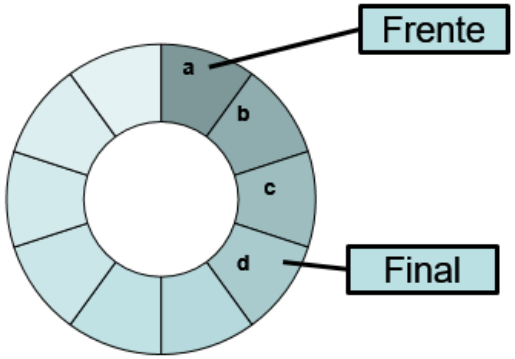


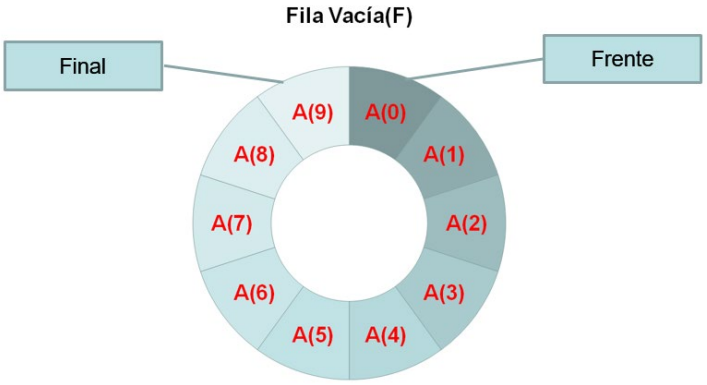
Una Implementación Fila con Arreglo en C

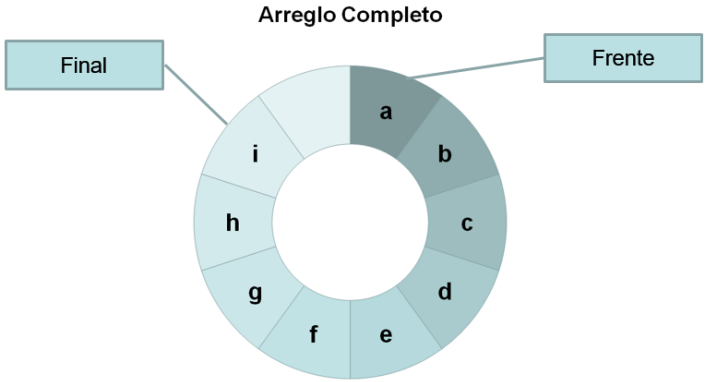
```
// PRIVADO
#define  MAX 10          // o cualquier otro valor
#define indefinido -9999
typedef int item;        // o cualquier otro tipo
typedef struct {
      int frente,final;
      item elementos [MAX];              } tipofila;
int avanzar(int i);

//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); }
```

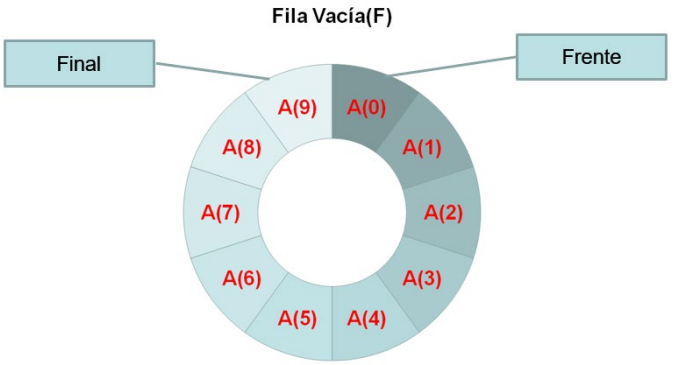






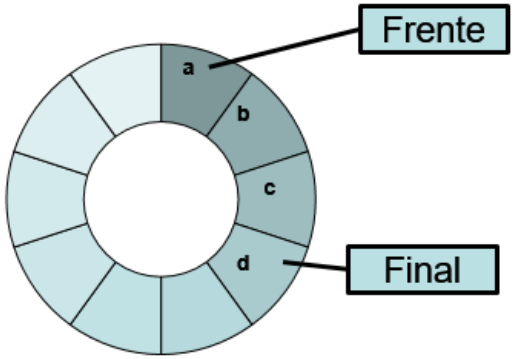
Una Implementación Fila con Arreglo en C

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




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

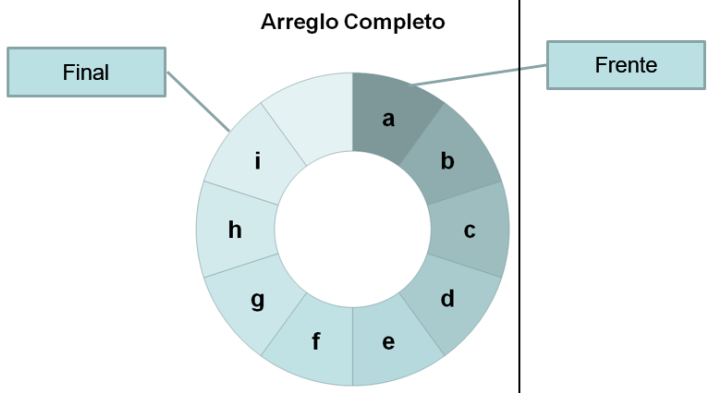
```
item frente (tipofila *f)
{    if( esfilavacia(f) )
            return (indefinido)
     else
            return(f->elementos[f->frente]);}
```

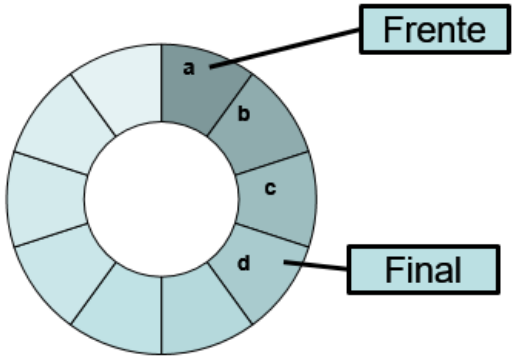


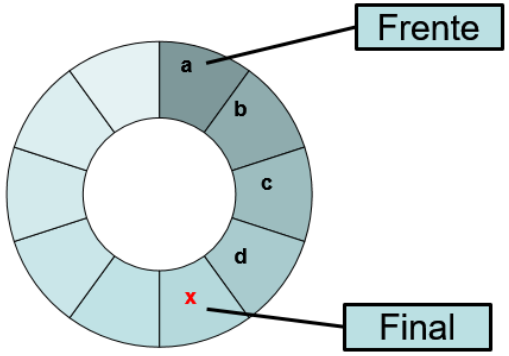

```
void defila (tipofila *f)
{    if ( !esfilavacia(f) ) f->frente = avanzar(f->frente);    }
```

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.