FILAS (Queues)

- Estrutura linear de acesso seqüencial;

- Ordena pela seqüência cronológica FIFO (First In First Out) o 1º a chegar será o 1º a sair da fila;

- Manipulação pelas duas extremidades, aqui utilizaremos como *frente* e *cauda* (outras nomenclaturas podem ser encontradas na literatura);

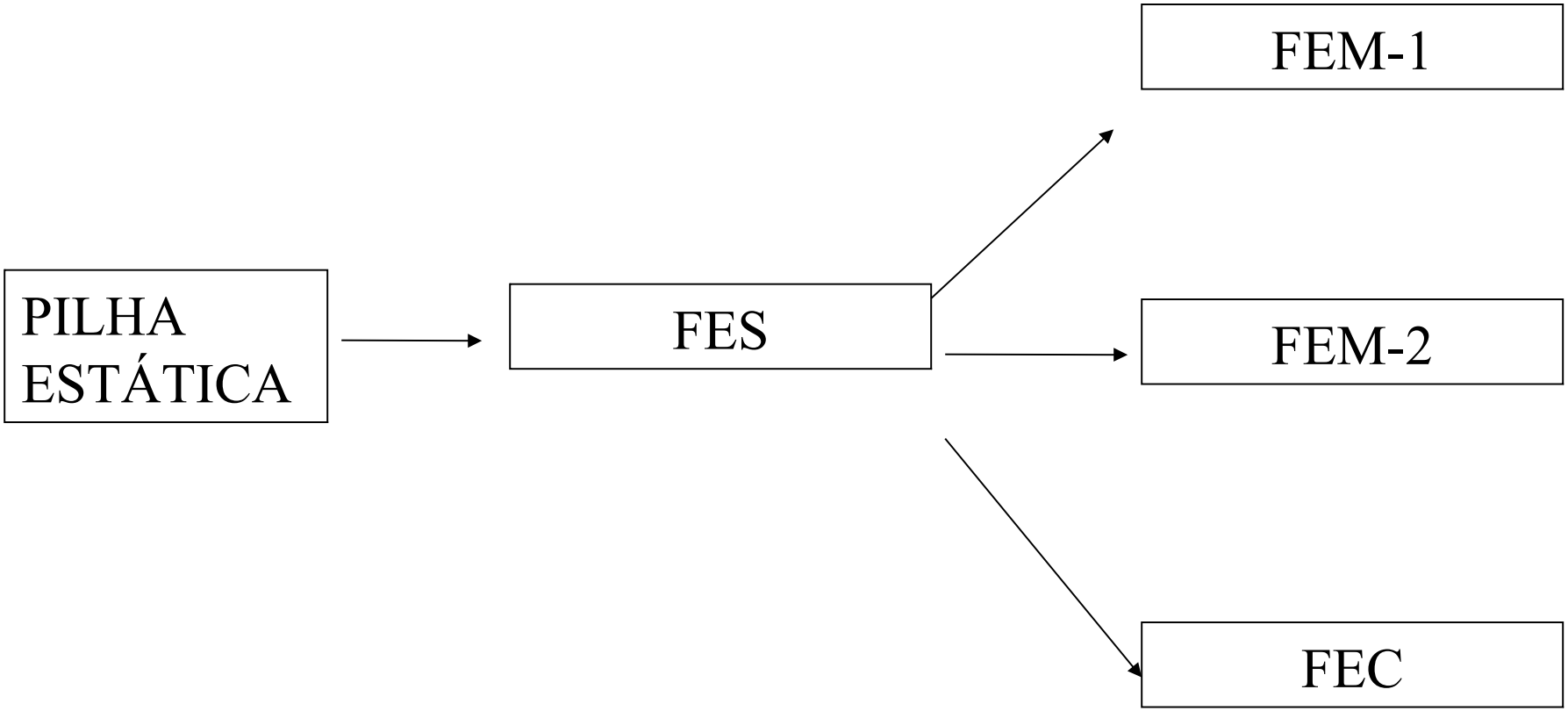
- Inserções sempre na *cauda* (entrada), remoções sempre da *frente* (saída);

- A fila se alonga desde a sua *cauda* até a sua *frente*.

Estudo da Fila Estática:

→ O ponto de partida será a PE recém estudada;

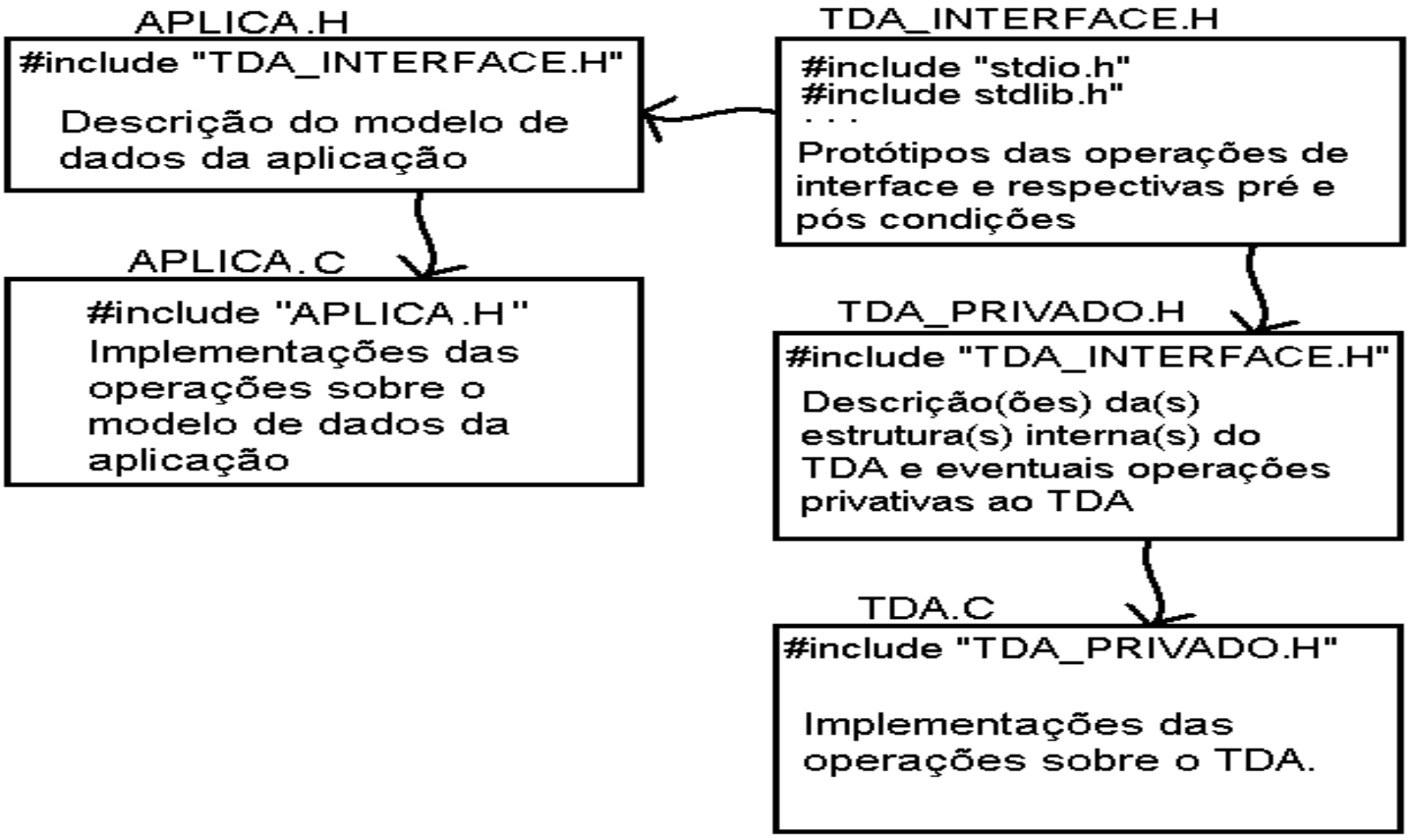
→ Incrementaremos os modelos de filas estáticas desde o mais simples (FES) até mais sofisticado (FEC).



Estudo da Fila Estática:

→ Incrementaremos os modelos de filas estáticas desde o mais simples (FES) até mais sofisticado (FEC).

→ A arquitetura do TDA é a mesma utilizada para a PE



1) FES: Fila Estática Simplória - adaptada diretamente de uma PE

```
typedef struct {
            void  **vetFila;
            int  comprimentoDoVetor;
            int  frente;  /*indexa a saída da fila */
            int  cauda;  /*indexa a entrada na fila */
            int  tamInfo;
      } Fila;
```

a) Inicialização: cauda = -1, frente = 0;
b) Número de elementos na Fila: cauda – frente + 1;
c) Fila vazia : cauda < frente;
d) Fila cheia: cauda == ComprimentoDoVetor – 1;
e) Inserções incrementam a *cauda;*
f) Remoções incrementam a *frente*

			VETOR				
	frente	**cauda**	0	1	2	3	
1)	0	-1					fila recém criada (vazia)
2)	0	-1					Tentaiva de remoção ⇒ ERRO: fila vazia
3)	0	0	X0				inseriu X0, cauda++
4)	0	1	X0	X1			inseriu X1, cauda++
5)	0	2	X0	X1	X2		inseriu X2, cauda++
6)	1	2		X1	X2		removeu X0, frente++
7)	2	2			X2		removeu X1, frente++
8)	3	2					removeu X2, frente++ Fila vazia cauda < frente

INCONSISTÊNCIAS NA FES

			VETOR				
	frente	**cauda**	0	1	2	3	
...	...	...	...	...	...	...	...
j)	2	3			P	Q	Fracasso ao tentar nova inserção pois o status é FILA CHEIA: *cauda==tamVet-1* **Porém há espaço no vetor: vet[0] e vet[1]**
...	...	...	...	...	...	...	...
k)	3	3				W	Fila com um único elemento: W
k+1)	4	3					Removeu W: frente++ Fila CHEIA *cauda==tamVet-1* e ao mesmo tempo Fila VAZIA *causa < frente*

2) FEM-1: Fila Estática com Movimentação de Dados a cada <u>remoção</u> – adaptação sobre a FES

Para evitar o alarme falso da FES, a cada remoção move-se toda a fila na direção do início do vetor aproveitando o espaço deixado pela remoção:

```
for (i=0; i < tamanhoFila; i++)
        memcpy(p->vet[i], p->vet[i+1], p->tamInfo);
p->cauda = p->cauda-1;
p->frente = 0;
```

Portanto:

a) Inserções incrementam a *cauda, conforme a FES...*
b) Porém a *frente* fica fixa no início do vetor (índice zero);
c) Tamanho da fila: *cauda* - *frente* + 1 = *cauda* - 0 +1 = *cauda*+1;
d) Inicialização: cauda = -1, frente = 0;
e) Fila vazia: cauda< frente;
f) Fila cheia: cauda= comprimentoDoVetor - 1.

	frente	cauda	vetor 0	1	2	3	
1)	0	-1					fila recém criada (vazia)
2)	0	-1					remoção ⇒ ERRO: fila vazia
3)	0	0	X0				inseriu X0, cauda++
4)	0	1	X0	X1			inseriu X1, cauda++
5)	0	2	X0	X1	X2		inseriu X2, cauda++
6)	**0** **0**	**2** **1**	**X1**	**X1** **X2**	**X2**		**removeu e moveu fila à esquerda (compactou)**
7)	**0**	**2**	**X1**	**X2**	**X3**		**inseriu X3**
8)	**0** **0**	**2** **1**	**X2**	**X2** **X3**	**X3**		**removeu e moveu fila à esquerda (compactou)**
9)	**0**	**2**	**X2**	**X3**	**X4**		**inseriu X4**
10)	**0**	**3**	**X2**	**X3**	**X4**	**X5**	**inseriu X5 ⇒ fila realmente cheia !!!!!!**

	frente	cauda	vetor 0	1	2	3	
1)	0	-1					fila recém criada (vazia)
2)	0	-1					remoção ⇒ ERRO: fila vazia
3)	0	0	X0				inseriu X0, cauda++
4)	0	1	X0	X1			inseriu X1, cauda++
5)	0	2	X0	X1	X2		inseriu X2, cauda++
6)	**0** **0**	**2** **1**	**X1**	**X1** **X2**	**X2**		**removeu e moveu fi à esquerda (co pa)**
7)	**0**	**2**	**X1**	**X2**	**X3**		
8)	**0** **0**	**2** **1**	**X2**	**X2** **X3**	**X3**		**rem à esquerda tou)**
9)	**0**	**2**	**X2**	**X3**	**X4**		**inseriu X4**
10)	**0**	**3**	**X2**	**X3**	**X4**	**X5**	**inseriu X5 ⇒ fila realmente cheia !!!!!!**

Movimentação Computacionalmente Custosa

A FEM-1 produz muitos deslocamentos

3) Uma estratégia computacionalmente mais "econômica" consiste em mover os dados apenas quando a compactação for necessária e realizável.

Chamaremos esta fila de FEM-2:

- **Estrutura híbrida → *frente* variável, conforme FES, aliado à compactação como na FEM-1;**
- **Executa a compactação ao detectar um falso sinal de "fila cheia". Ao invés de compactar a cada remoção.**

3) FEM-2: Solução híbrida - Compactando na hora certa

Ao falso sinal de “fila cheia”, compacta-se.

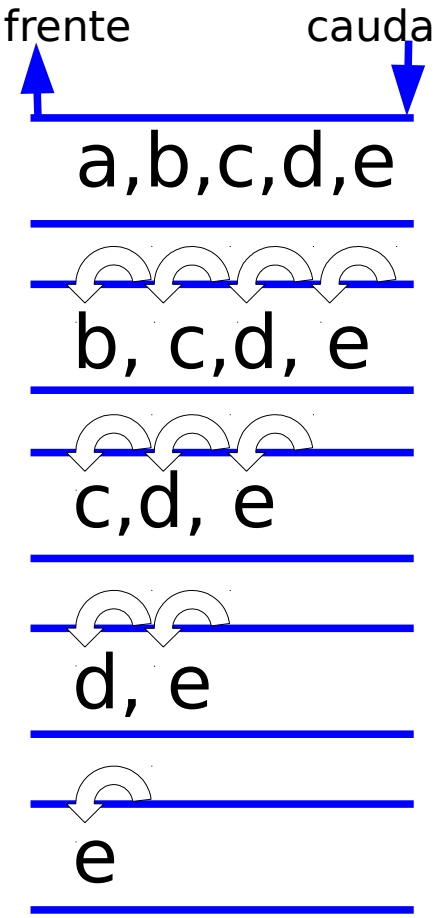
```
inserção(....)
  Se (cauda ==comprimentoDoVetor-1)
        tamanhoDaFila = cauda - frente + 1;
        Se(tamanhoDaFila < comprimentoDoVetor)
              for(i=0; i < tamanhoDaFila; i++)
                  memcpy(p->vet[i], p->vet[i+frente], p->tamInfo);
              p->cauda -= p->frente;
              p->frente = 0;
              inserção no final da fila;

        Senão
            FILA realmente cheia;

  Senão  inserção no final da fila;
```

	frente	cauda	vetor 0	1	2	3	
1)	0	-1					fila recém criada
2)	0	-1					remoção ? ERRO: fila vazia
3)	0	0	X0				inseriu X0, cauda++
4)	0	1	X0	X1			inseriu X1, cauda++
5)	0	2	X0	X1	X2		inseriu X2, cauda++
6)	0	3	X0	X1	X2	X3	inseriu X3, cauda++
7)	1	3		X1	X2	X3	Removeu, frente++
8)	2	3			X2	X3	Removeu, frente++
9)	2	3			X2	X3	Inserção X4: Compactou e Inseriu, corrigiu frente e cauda
	0	1	X2	X3			
	0	2	X2	X3	X4		

Independentemente de FEM1 ou 2, a movimentação para compactação da Fila pode ser uma operação bastante lenta.

FEM-1: Para *n* itens em uma FEM-1, uma remoção implica em *n-1* movimentos...

Esse número pode vir a crescer bastante. Para uma sequência de remoções consecutivas.



após 5 inserções: n=5

(n-1) movimentos

remove ‘a’, 4 movimentos,

remove ‘b’, 3 movimentos,

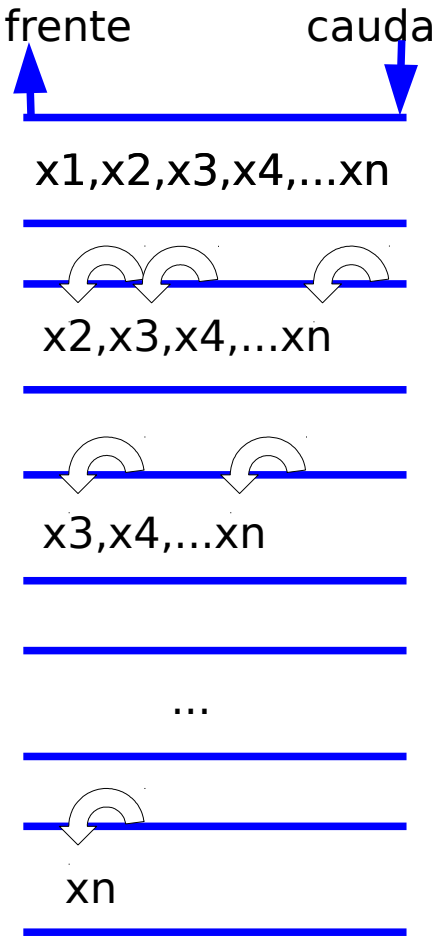
remove ‘c’, 2 movimentos

remove ‘d’, 1 movimento

Total=10 movimentos

A movimentação para compactação da Fila pode ser uma operação bastante lenta.

FEM-1: sequência de remoções consecutivas



A sequência de movimentos corresponde a uma PA:
(n-1),...,3,2,1 ou 1,2,3,..., (n-1)

PA:
posição do último: k=n-1
valor do ultimo: a_k=n-1

O total de movimentos para um caso genérico:
Termo geral da PA: $a_k=a_1+(k-1)r$

fazendo k=n-1, $a_n=1+(n-1-1)*1=n-1$

Soma de uma PA: $S_k= ((a_1+a_k)*k)/2$

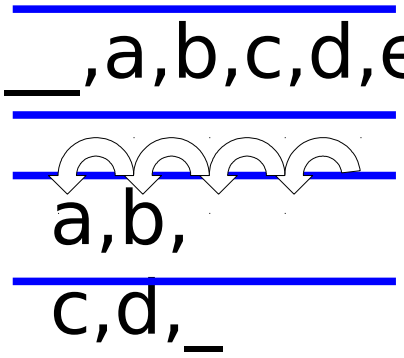
$S_n=((1+n-1)*(n-1))/2=(n(n-1))/2=(n^2-n)/2$

Se o custo computacional for proporcional aos movimentos, para n itens teremos um custo quadrático .

A movimentação para compactação da Fila pode ser uma operação bastante lenta.

Considerando a FEM-2 nas mesmas condições utilizadas para o exemplo da FEM-1, para a mesma sequencia de *n* inserções seguida de *n* remoções, não haveria o estado de “falso-cheio”, portanto não ocorreria movimentação de dados.

No entanto, quando ocorre a movimentação de dados na FEM-2 o custo computacional também será desfavorável se houver uma combinação inserção/remoção em sequência...



Vet[0] está vago, inserção com falso cheio

Para n inseridos: n movimentos,

4) FEC: Fila Circular

Considera o vetor como um arranjo circular, como se o seu final se ligasse ao seu início, não havendo interrupção.

É bastante vantajosa quando se trata da implementação sobre vetor pois viabiliza a reutilização das posições desocupadas sem o custo da movimentação de dados vista na FEM-1 e FEM-2.

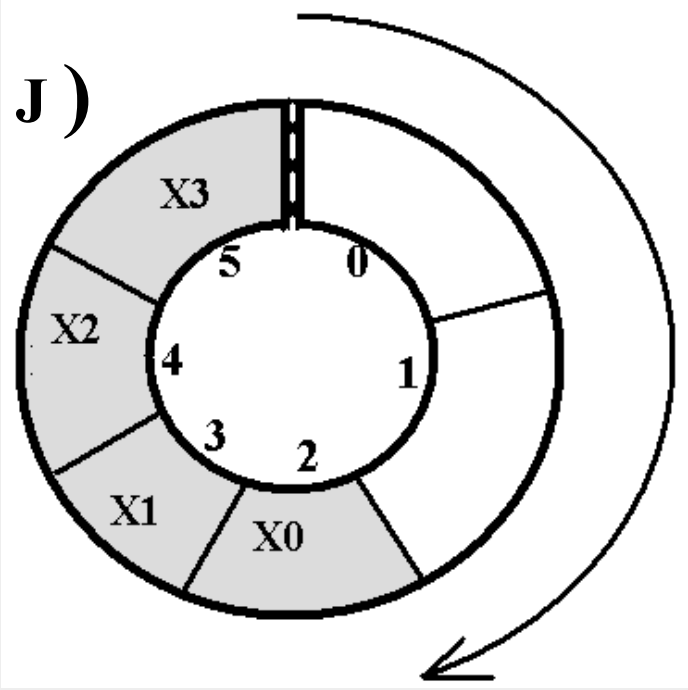
Atenção:

PARA A FEC *cauda* < *frente* NÃO MAIS IMPLICARÁ EM FILA VAZIA

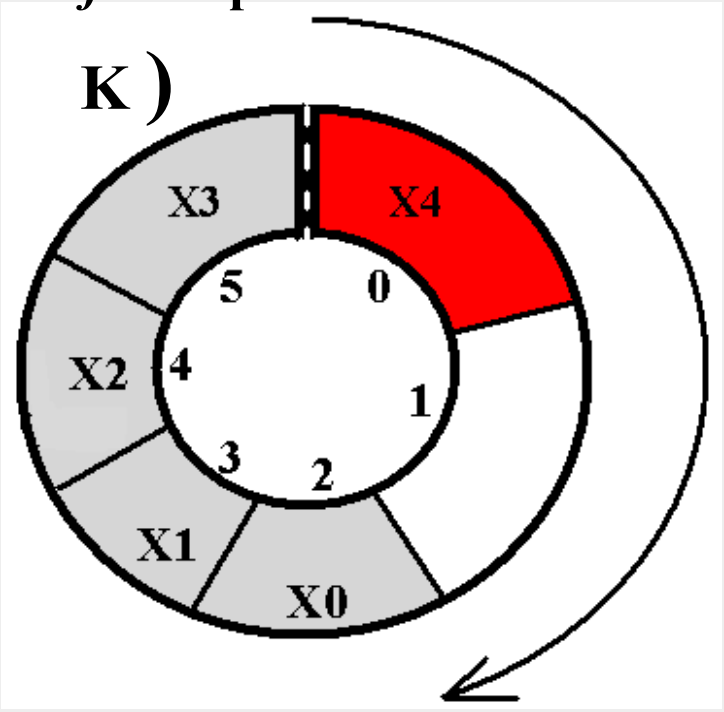
			vetor						
	frente	**cauda**	0	1	2	3	4	5	
...	...	...			...	...	...	...	...
j)	2	5			X0	X1	X2	X3	
j+1)	2	0	X4		X0	X1	X2	X3	Inserção circular de X4

***cauda* < *frente* porém a fila não está vazia**

J)



K)



Da *frente* para a *cauda* no sentido horário: X0,X1,X2,X3,X4

E agora...

Se *cauda* < *frente* não mais implica em fila VAZIA !

Como testar tal condição ?

Uma alternativa é acrescentar o campo *tamanhoDaFila* na estrutura interna do TDA.

a) Inicialização: *cauda* = -1, *frente* = 0;

b) Tamanho da fila ⇒ anotado no descritor;

c) Fila vazia :

tamanho da fila = 0;

d) Fila cheia :

tamanho da fila = comprimentoDoVetor

Estrutura Fila Estática Circular:

```
typedef struct {
        void **vetFila;
        int comprimentoDoVetor;
        int frente; /* indexa o início da Fila */
        int cauda;   /*indexa o final da Fila */
        int tamanhoDaFila; /*num de elementos*/
        int tamInfo;
   } Fila;
```

inserção( )

```
SE (tamanho atual da fila < tamanho do vetor)
    /*     há espaço no início do vetor     */
   SE (cauda = = tamanho do vetor-1)
         /* utilize o aspecto circular  */
        cauda = 0;
        vetor[cauda] = novo;
   SENÃO
        vetor[++cauda] = novo
   tamanho atual da fila ++
SENÃO
    fila realmente cheia!!
```

Alternativa p/ controle da "circularidade":

```
cauda = (cauda+1)%tamanho do vetor

vetor[cauda] = novo

tamanho atual da fila ++
```

Remoção( )

```
SE(tamanho da fila = = 0)
        fila vazia
 SENÃO
        SE (frente = = tamanho do vetor-1)
                frente = 0
        SENÃO
                frente++
        tamanho da fila - -
```

Alternativa p/ controle da “circularidade”:

frente = (frente+1)%tamanho do vetor

tamanhoDaFila - -

	frente	cauda	tamanho	vetor 0	1	2	3	
1)	0	-1	0					fila recém criada
2)	0	0	1	X0				inseriu X0
3)	0	1	2	X0	X1			inseriu X1
4)	0	2	3	X0	X1	X2		inseriu X2
5)	1	2	2		X1	X2		Removeu (X0)
6)	1	3	3		X1	X2	X3	inseriu X3
7)	2	3	2			X2	X3	Removeu (X1)
8)	3	0	1	X4			X3	inseriu X4 (circulou a cauda)
9)	3	1	2	X4	X5		X3	inseriu X5
10)	3	2	4	X4	X5	X6	X3	inseriu X6
11)	3	2	4	X4	X5	X6	X3	Tenta a inserção X7 ⇒ ERRO: fila CHEIA
12)	0	2	3	X4	X5	X6		Removeu (X3) (circulou a frente)
13)	1	2	2		X5	X6		Removeu (X4)
14)	2	2	1			X6		Removeu (X5)
15)	3	2	0					Removeu(X6): tamanho = 0 ⇒ fila VAZIA !!!

Da Pilha Estática para a FES basta adaptar código;

Da FES para a FEM-1 basta alterar a remoção;

Da FES para a FEM-2 basta alterar a inserção;

Da FES para a FEC basta alterar a inserção, remoção tratando a determinação do tamanho da fila.

Exercícios:

1) Implemente as quatro estratégias discutidas anteriormente com a seguinte funcionalidade:

```
int cria(ppFila pp, int tamVet, int tam info);

int destroi(ppFila pp);

int buscaNaFrente(pFila p, void *reg);

int buscaNaCauda(pFila p, void *reg);

int testaVazia(pFila p);

int testaCheia(pFila p);

int reinicializa(pFila p);

int enfileira(pFila p, void *novo);

int desenfileira(pFila p, void *reg);
```

2) Implemente o TDA *MultiFilaCircular*.

3) Em relação ao *TDA-FEC* original:

Proponha um mínimo de alterações para implementar a função que calcula o tamanho da FEC, considerando que o descritor não possui o campo *tamanhoDaFila*.

Nesse caso o valor do comprimento da fila será calculado.

Para este cálculo é necessário levar em conta:

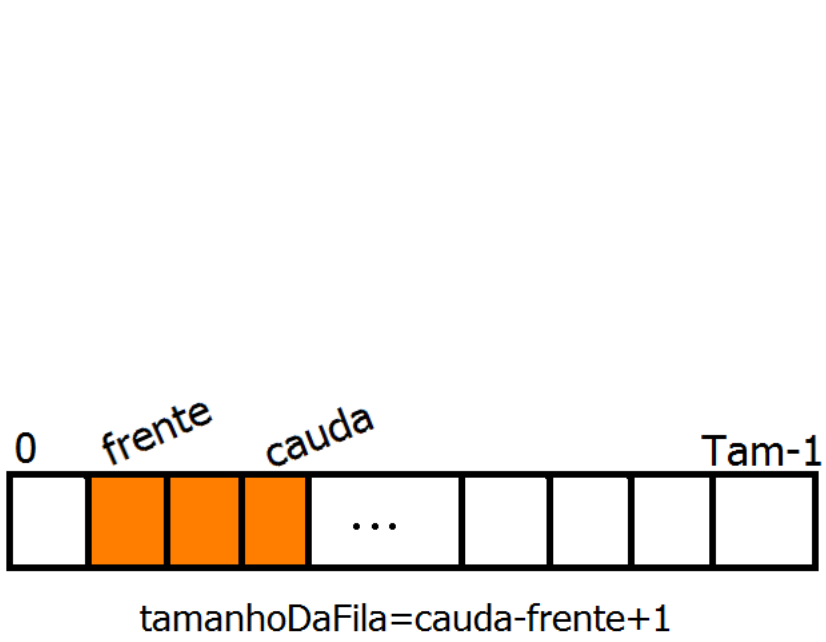
3.1) A situação convencional, sem a efetiva "circulação" do vetor. Quando cauda ≥ frente sempre;

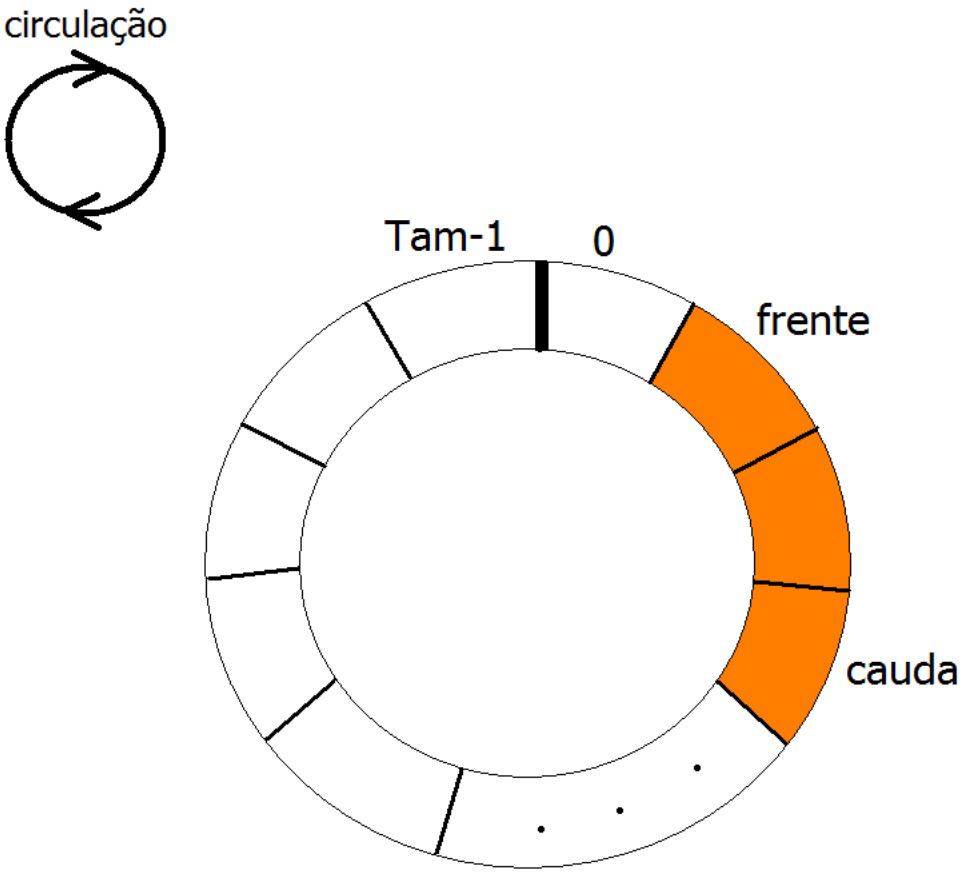
3.2) As situações quando cauda < frente.

Esses aspectos serão discutidos nos próximos slides...

3.1) Situação convencional, sem o uso efetivo da “circulação” do vetor:

Se (cauda ≥ frente): tamFila=cauda-frente+1





3.2) Mas, se ocorrer (cauda < frente)?

Essa situação só ocorrerá se ou a cauda ou a frente circularem ultrapassando a última célula do vetor (índice *Tam-1*).

Teremos duas possibilidades para esse caso:

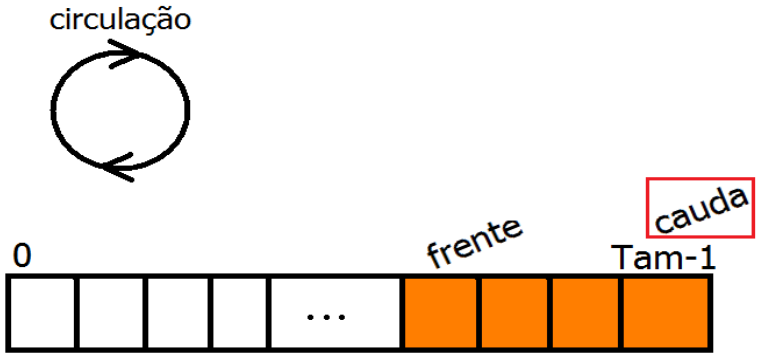
3.2.1) A *cauda* < *frente* devido a uma circulação da cauda;

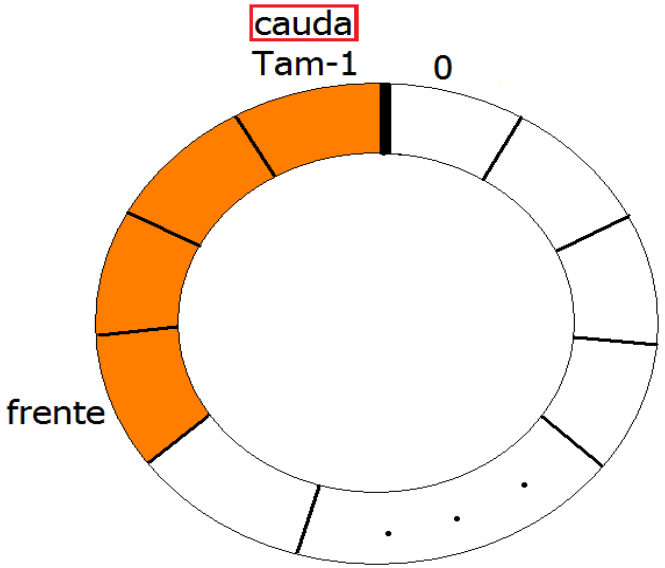
3.2.2) A *cauda* < *frente* devido a uma circulação da frente;

Discutimos ambas a seguir...

3.2.1) A *cauda* < *frente* devido a uma circulação da **cauda**

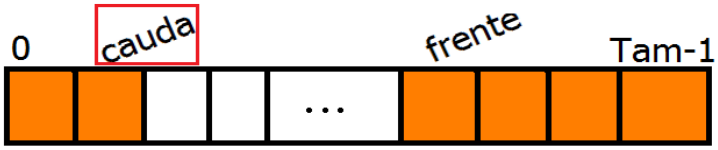
A)



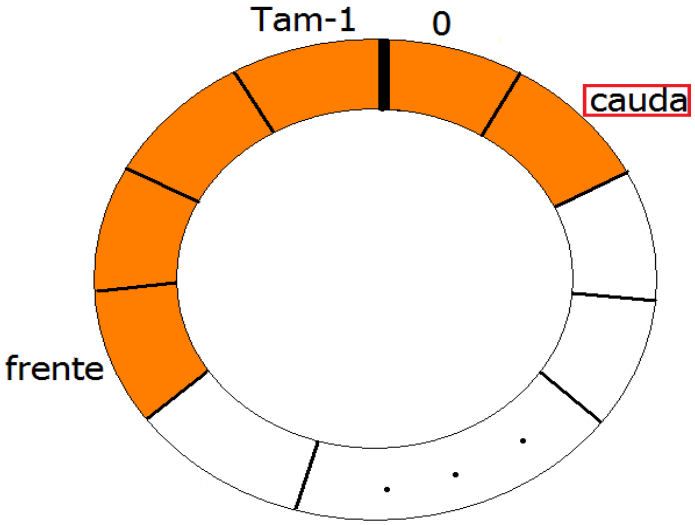


B)

Duas inserções depois:



tamanhoDaFila=(Tam-frente)+(cauda+1)

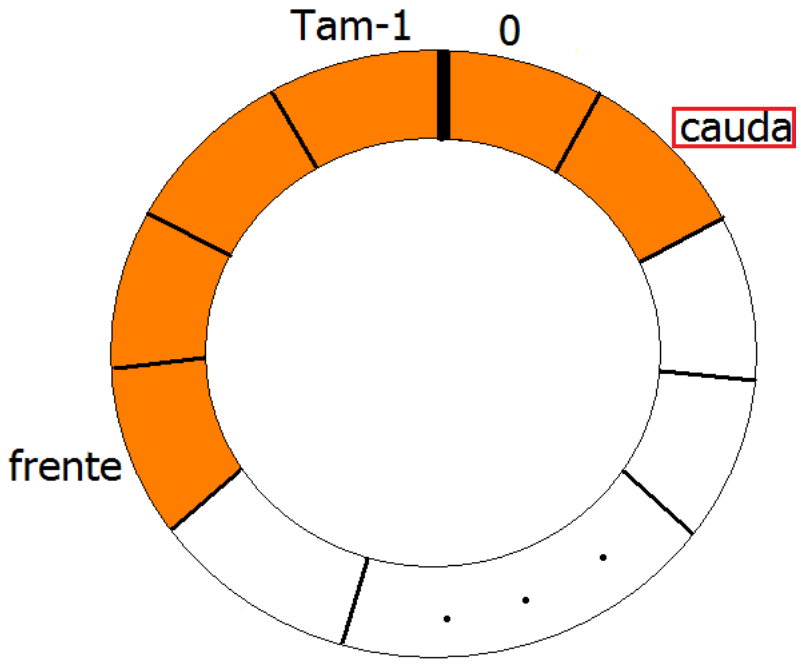


3.2.1) A *cauda* < *frente* devido a uma **circulação da cauda**

Nesse caso:

TamanhoDaFila = ((Tam-1) – frente + 1) + (cauda + 1) =

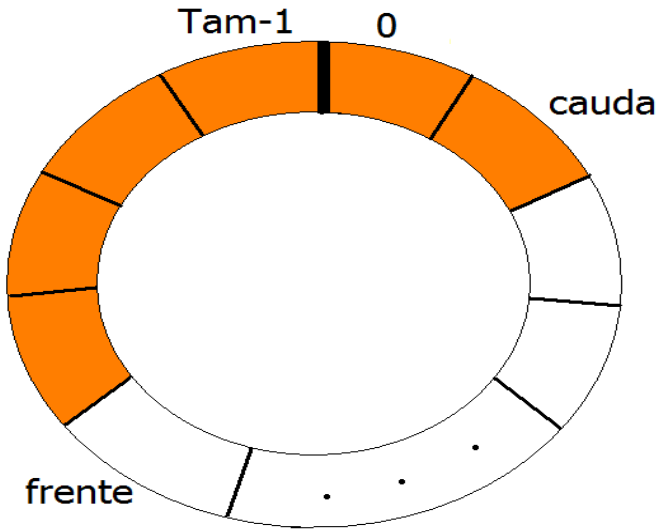
= (Tam – frente) + (cauda + 1)



3.2.2) A *cauda* < *frente* devido a uma **circulação da frente** na remoção do item na cauda (último item na fila), **esvaziando a FEC**.

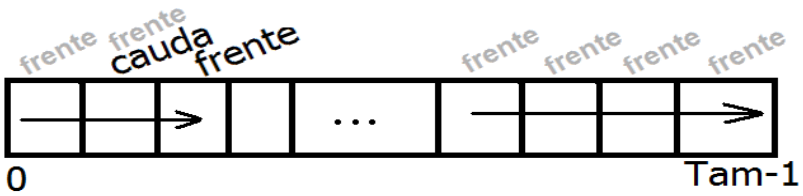
A)

tamanhoDaFila=(Tam-frente)+(cauda+1)

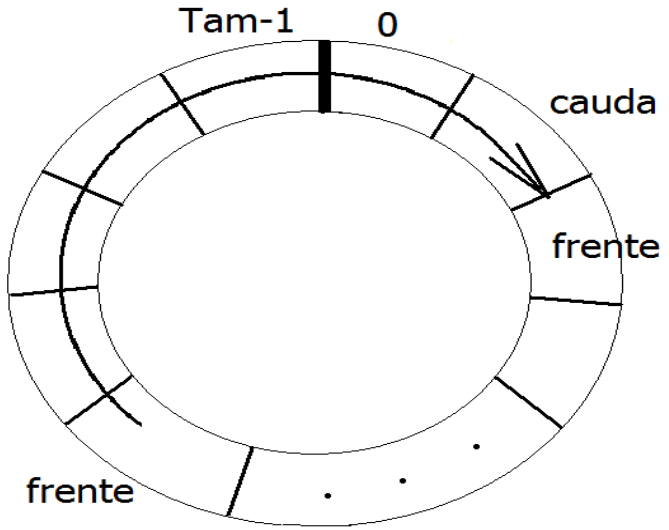


B)

circulação



tamanhoDaFila=(Tam-frente)+(cauda+1)



3.2.2) A *cauda* < *frente* devido a uma **circulação da frente** na remoção do item na cauda (último item na fila), **esvaziando a FEC**.

- Nesse caso a fila tornou-se vazia! É preciso introduzir um recurso para detectar esta condição.
- Para tanto basta acrescentar uma *flag* no nó de dados marcando se o nó foi removido.
- Por fim teremos:

  Se (cauda ≥ frente): tamFila=cauda-frente+1

  Senão: /* cauda < frente */

    Se (cauda foi removida): tamFila = 0

    Senão: tamFila = (tamVet-frente)+(cauda+1)