FILAS (Queues)

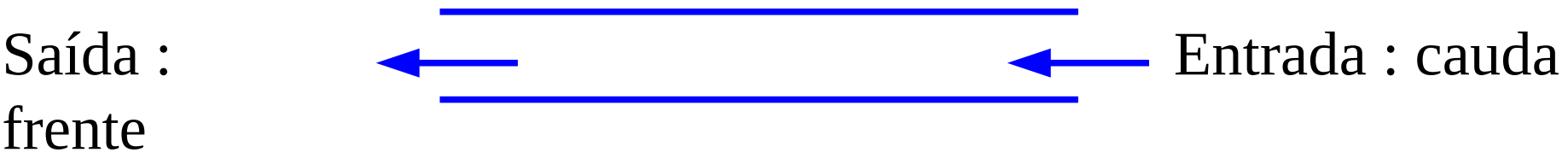
- Estrutura linear de acesso sequêncial;

- Ordena pela seqüência cronológica PEPS – Primeiro que Entra é o Primeiro que Sai (FIFO - First In First Out);

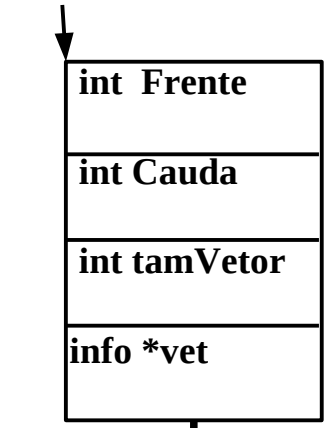
- Manipulação pelas duas extremidades, aqui utilizaremos como *frente* e *cauda* (outras nomenclaturas podem ser encontradas na literatura);

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

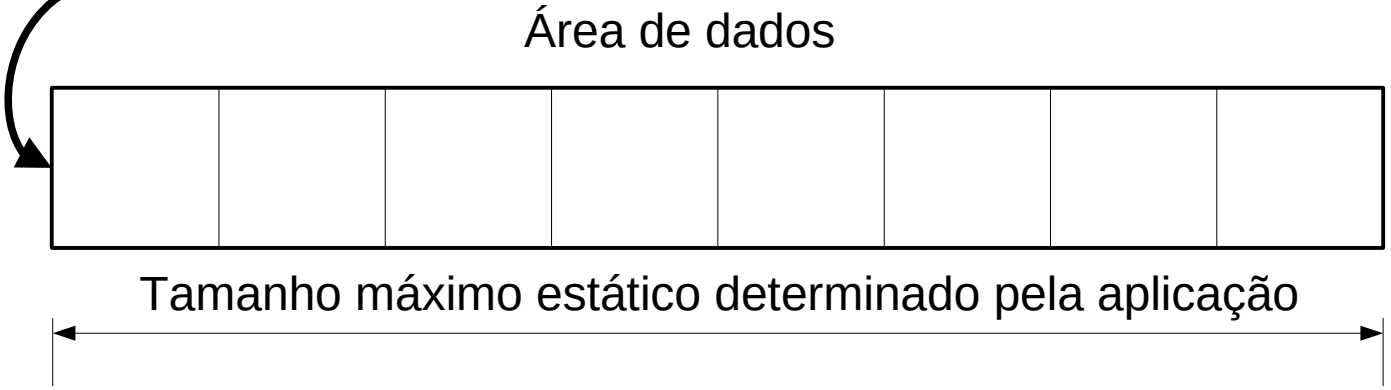
- A fila se alonga desde a sua *cauda* até a sua *frente*.



Estudo da Fila de tamanho máximo estático:



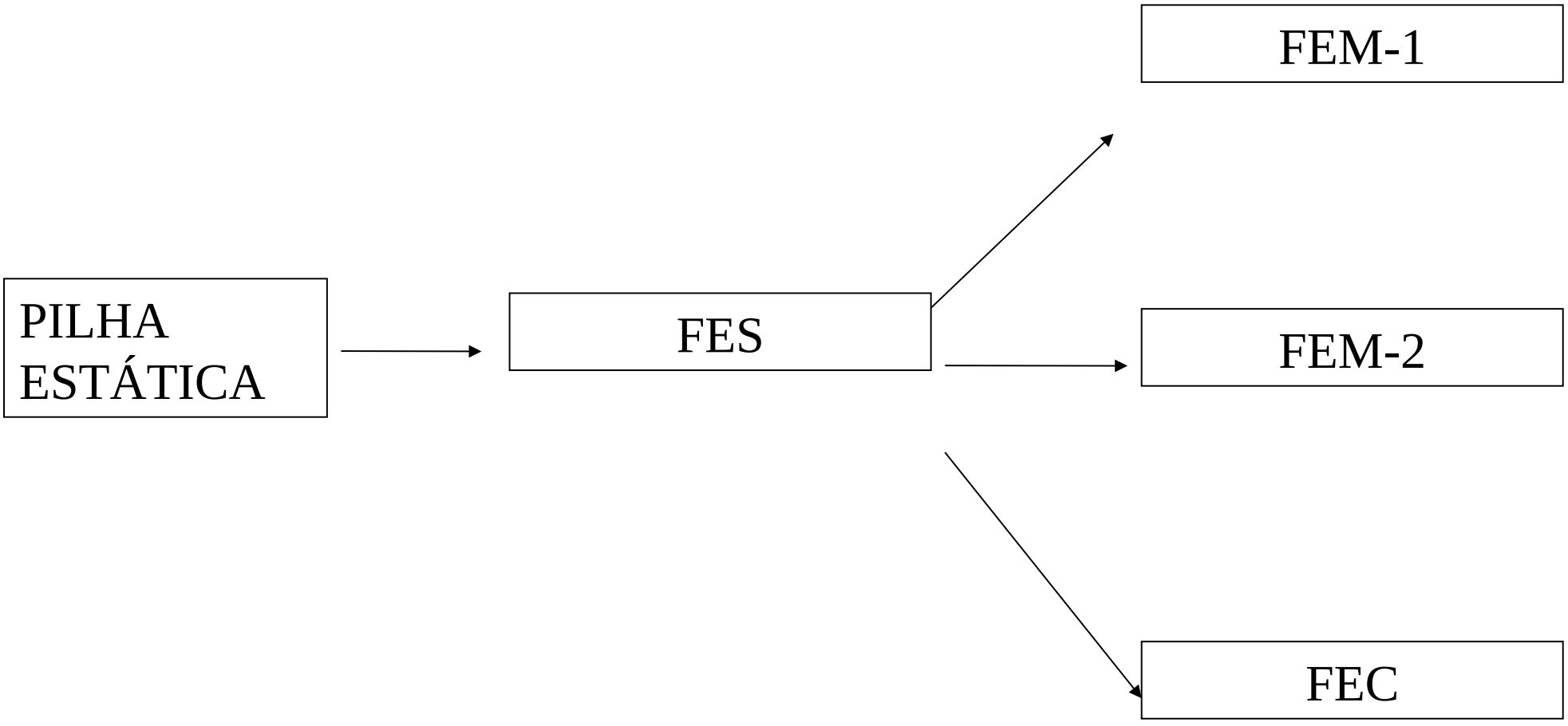

```
Descritor:
typedef struct {
        info  *vet;
        int  tamVetor;
              int  frente;  /*indexa a saída da fila */
              int  cauda;     /*indexa a entrada na fila */
          } Fila;
```



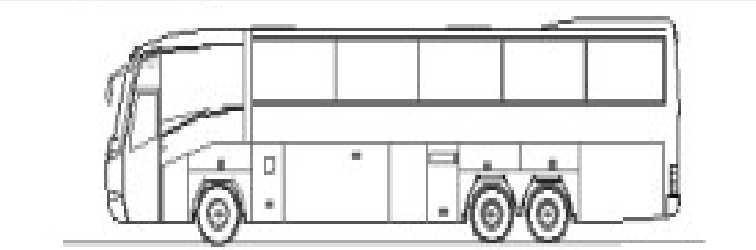
Estudo da Fila de tamanho máximo estático::

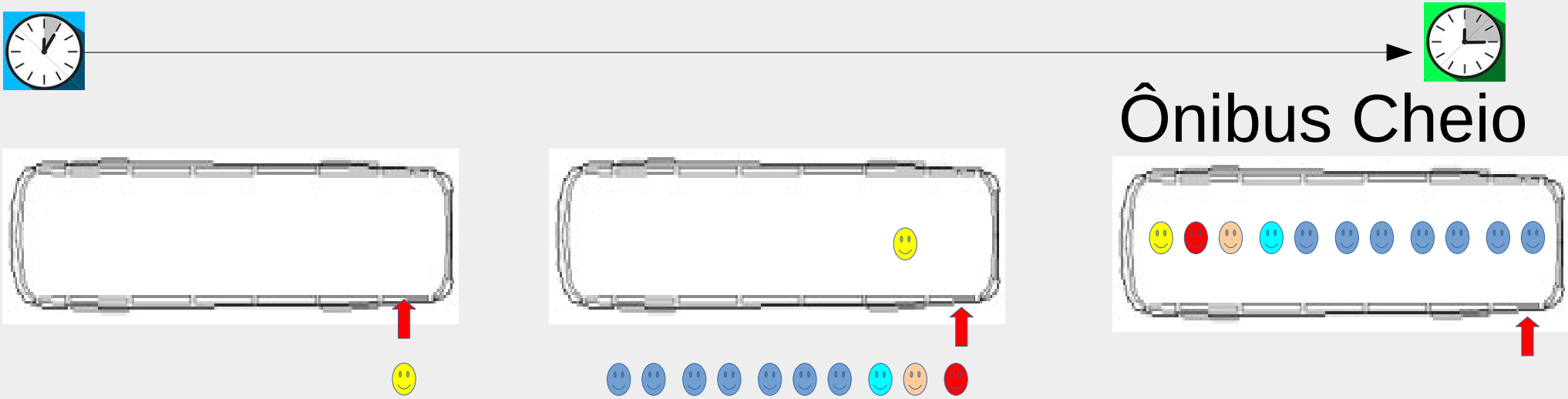
→ O ponto de partida será a PE recém estudada;

→ Incrementaremos os modelos de filas estáticas desde o mais simples (FES), passando pelos modelos com movimentação de dados (FEM) até mais o modelo circular (FEC), o mais eficiente.

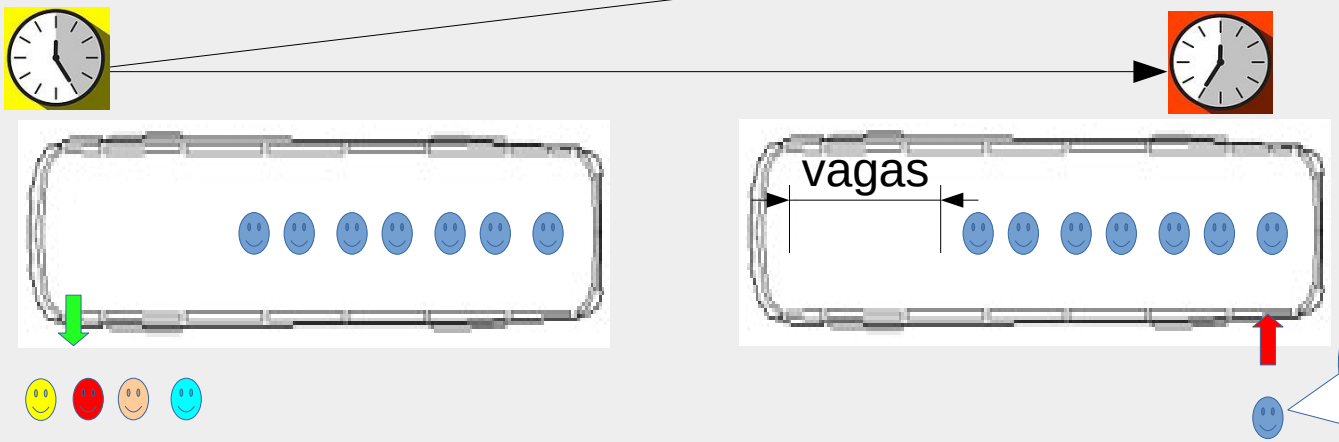


1) FES





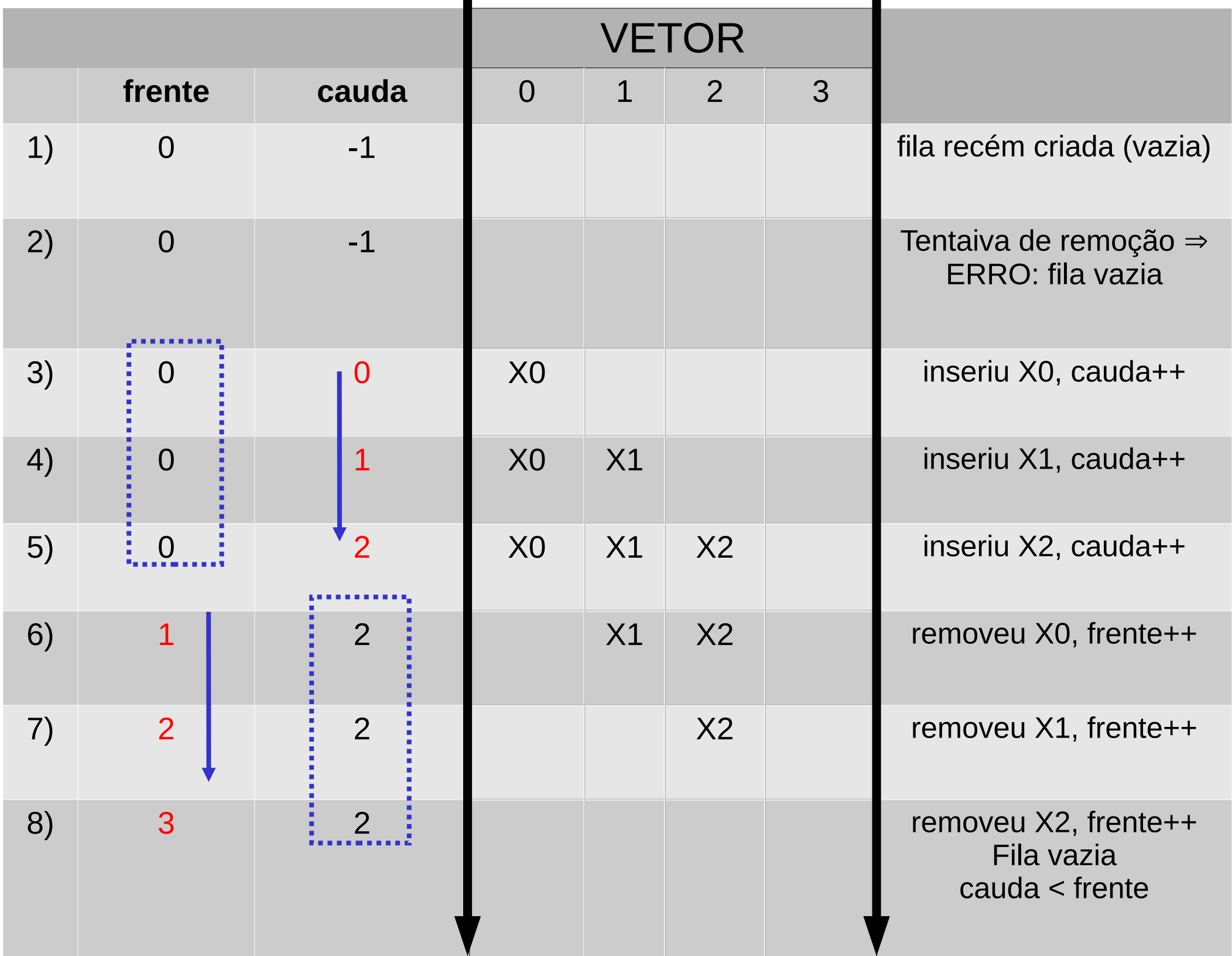
Ônibus Cheio



Ônibus Cheio?
Falso cheio!

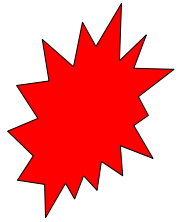
1) FES: Fila Estática Simplória - adaptada diretamente de uma PE

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

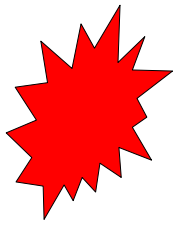
INCONSISTÊNCIAS NA FES

			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*

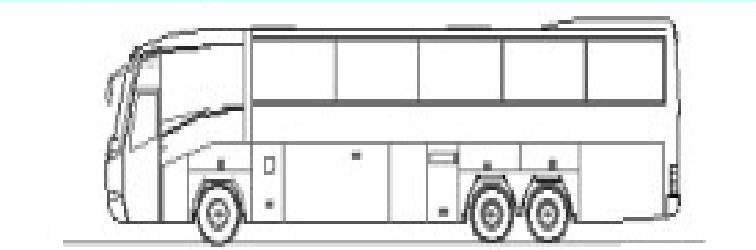


ESSA FES NÃO GERENCIA ADEQUADAMENTE O ESPAÇO NO VETOR E AINDA GERA INCONSISTÊNCIAS.

A FES SÓ É DISCUTIDA AQUI PARA TERMOS UMA REFERÊNCIA QUE MOTIVA AS OUTRAS FORMAS DE IMPLEMENTAÇÃO DISCUTIDAS A SEGUIR...



2) FEM-1



(t_0)

(t_1)

(t_2)

Ônibus Cheio

(t_3)

(t_4)

(t_5)

(t_6)

(t_7)

(t_8)

(t_9)

(t_{10})

(t_{11})

(t_{12})

Ônibus Cheio!

2) FEM-1: Fila Estática com Movimentação de Dados a cada remoção – 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++)
    vet[i] = vet[i+1];
cauda = cauda-1;
 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 mo** **(cor**
7)	**0**	**2**	**X1**	**X2**	**X3**		
8)	**0** **0**	**2** **1**	**X2**	**X2** **X3**	**X3**		**re** **uerda**
9)	**0**	**2**	**X2**	**X3**	**X4**		**nseriu X4**
10)	**0**	**3**	**X2**	**X3**	**X4**	**X5**	**inseriu X5 ⇒** **fila realmente cheia !!!!!!**

Essa Movimentação gera custos frequentes Computacionais indesejáveis (complexidade de tempo)

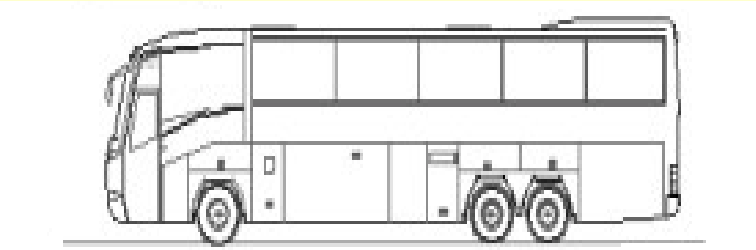
A FEM-1 produz muitos deslocamentos

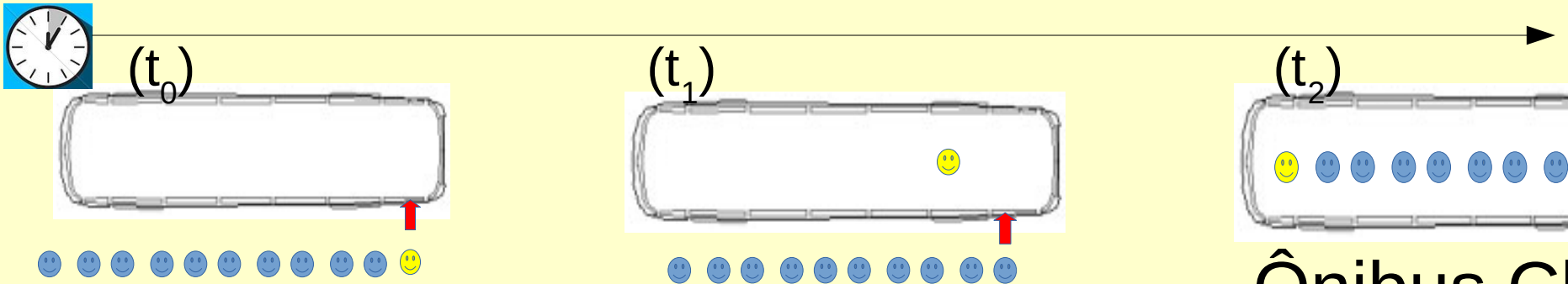
3) Uma estratégia computacionalmente mais "econômica" consiste em mover os dados apenas quando a compactação for necessária e realizável.

Chamaremos esta fila de FEM-2:

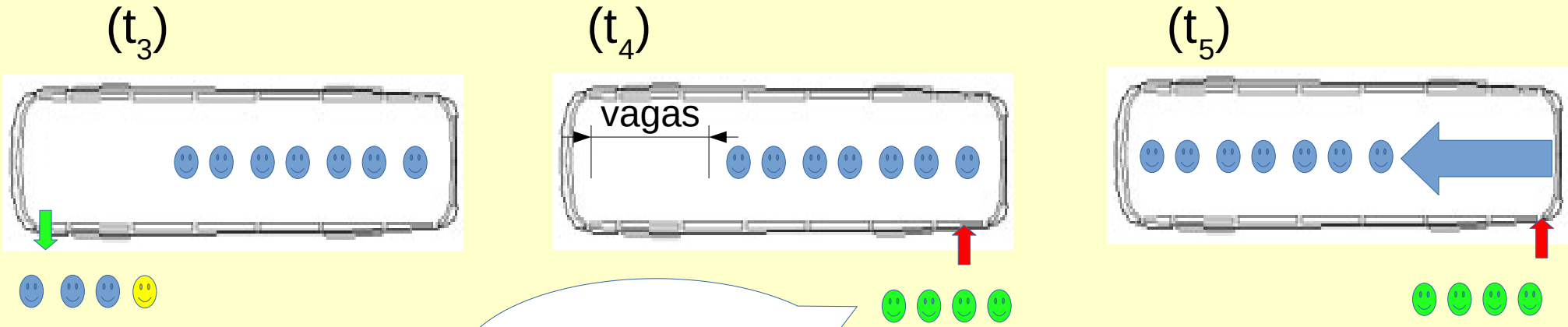
- **Estrutura híbrida → *frente* variável, conforme FES, aliado à compactação como na FEM-1;**
- **Executa a compactação ao detectar um falso sinal de "fila cheia". Ao invés de compactar a cada remoção.**

3) FEM-2



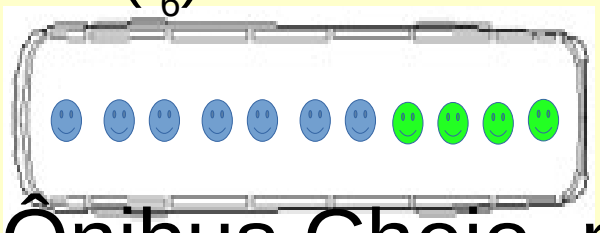


Ônibus Cheio



Não!
Tem espaço na frente

Ônibus Cheio?



Ônibus Cheio, mesmo

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

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

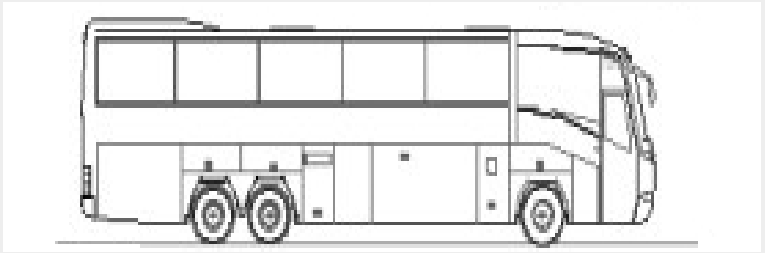
```
inserção(....)
  If (cauda ==comprimentoDoVetor-1)
        tamanhoDaFila = cauda - frente + 1;
        If (tamanhoDaFila < comprimentoDoVetor)
              for(i=0; i < tamanhoDaFila; i++)
                    vet[i]=vet[i+frente];
              cauda -= frente;
              frente = 0;
              inserção no final da fila;

        Senão
            FILA realmente cheia;

  Senão  inserção no final da fila;
```

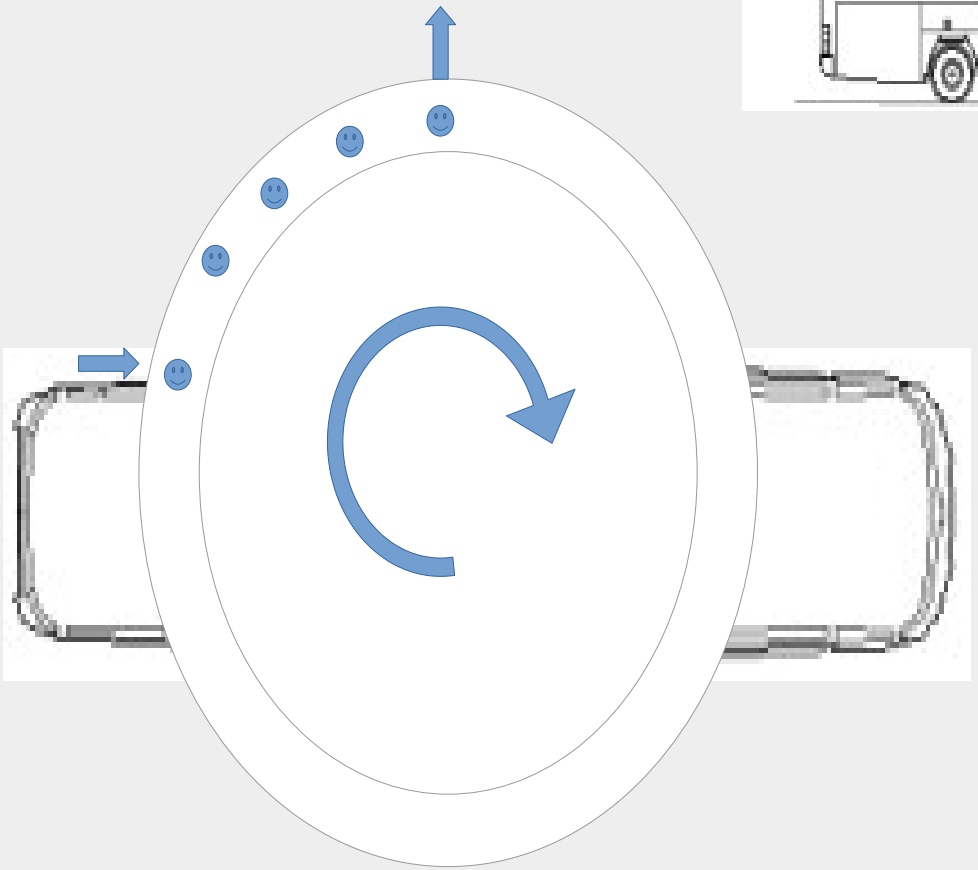
	frente	cauda	vetor 0	vetor 1	vetor 2	vetor 3	
1)	0	-1					fila recém criada
2)	0	-1					remoção ? ERRO: fila vazia
3)	0	0	X0				inseriu X0, cauda++
4)	0	1	X0	X1			inseriu X1, cauda++
5)	0	2	X0	X1	X2		inseriu X2, cauda++
6)	0	3	X0	X1	X2	X3	inseriu X3, cauda++
7)	1	3		X1	X2	X3	Removeu, frente++
8)	2	3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

	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	Remoção X0, frente++
8)	2	3			X2	X3	Remoção X1, frente++
9)	2 0 0	3 1 2			X2 X2 X4	X3 X3 X3	Remoção X4: Compactou e Inseriu, corrigiu frente e cauda

Essa Movimentação gera custos relativamente menores. Mais ainda ainda indesejáveis (complexidade de tempo)

4) Fila Circular - FEC





4) FEC: Fila Circular

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

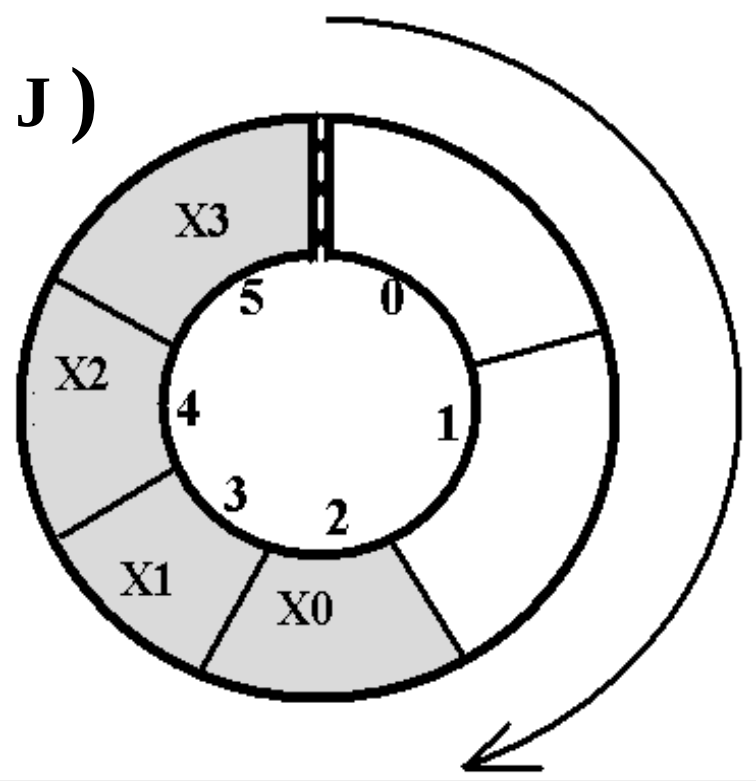
É bastante vantajosa quando se trata da implementação sobre vetor pois viabiliza a reutilização das posições desocupadas sem o custo da movimentação de dados vista na FEM-1 e FEM-2.

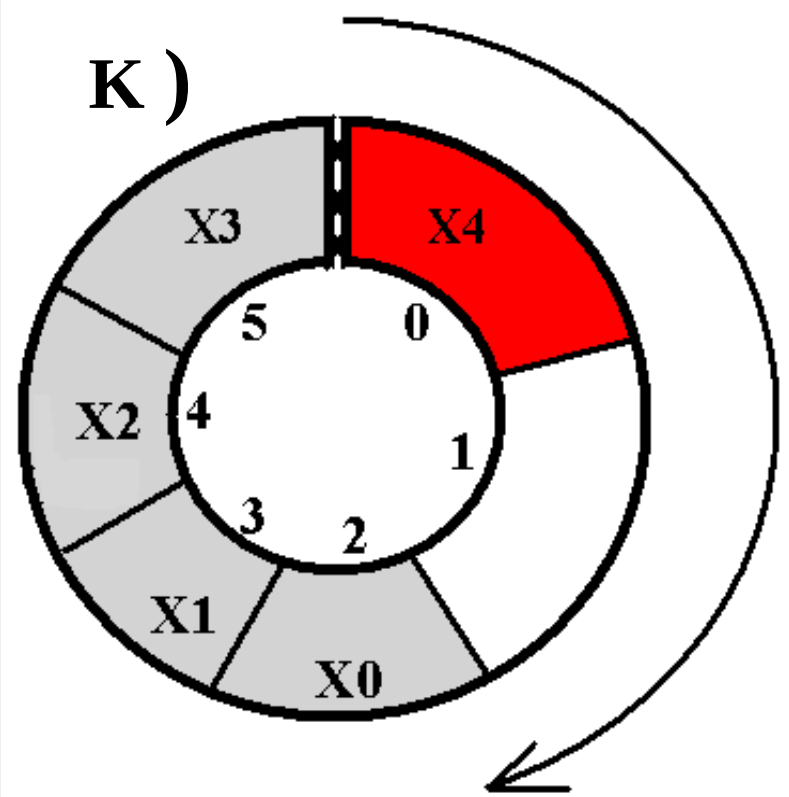
Atenção:

NA FEC *cauda* < *frente* NÃO INDICA 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**





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 tamVetor;
        int frente; /* indexa o início da Fila */
        int cauda;   /*indexa o final da Fila */
        int tamanhoDaFila; /*num de elementos*/
   } 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;
  SENÃO
          cauda++;
 vet[cauda]=novo;
 tamanho atual da fila ++
SENÃO
   fila realmente cheia!!
```

Alternativa p/ controle da “circularidade”:

```
cauda = (cauda+1)%tamanho do vetor
vet[cauda]=novo
tamanho atual da fila ++
```

Remoção( )

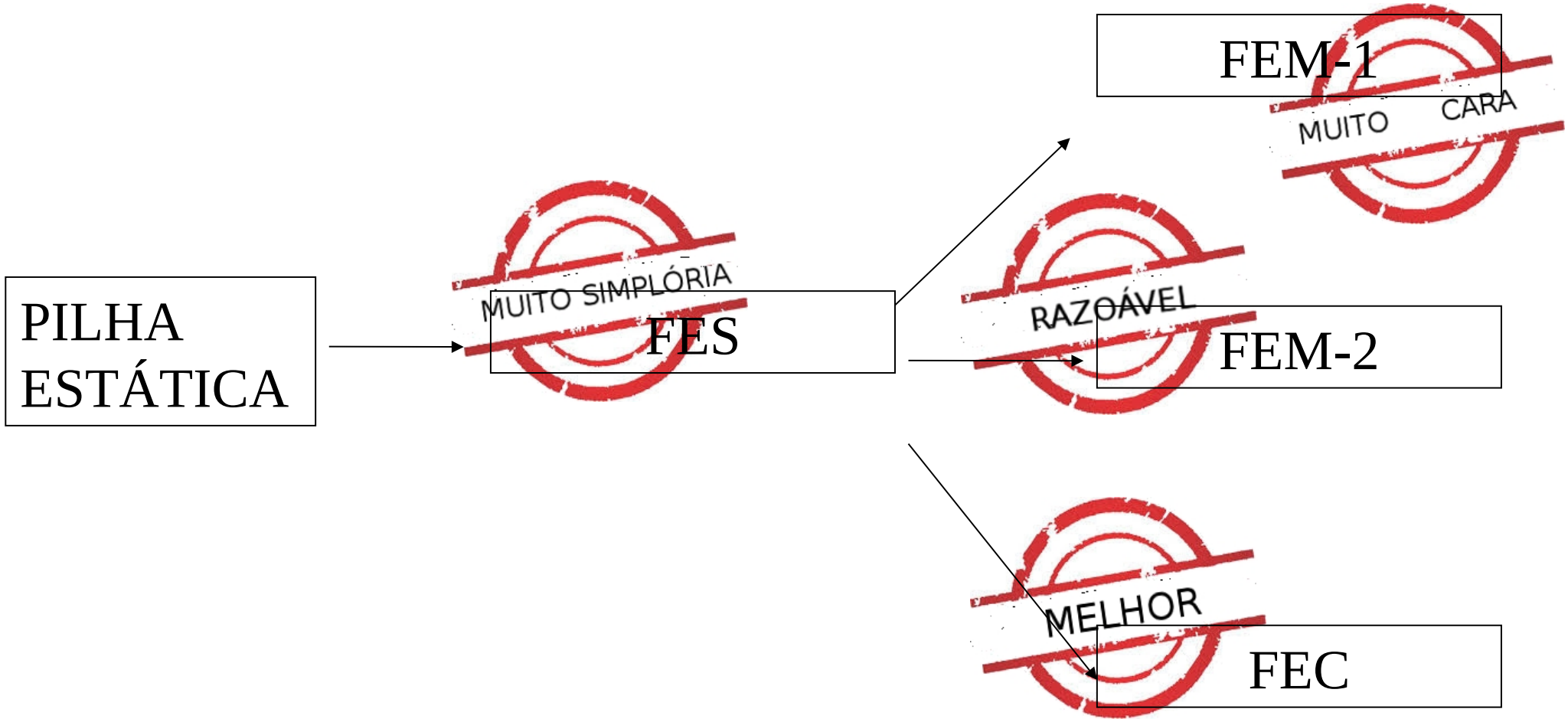
```
SE(tamanho da fila = = 0)
        fila vazia
  SENÃO
        SE (frente = = tamanho do vetor-1)
                frente = 0
        SENÃO
                        frente++
        tamanho da fila - -
```

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:

Implemente as quatro estratégias discutidas anteriormente com a seguinte funcionalidade:

```
Fila * cria(int tamVet, int tam info);

Fila * destroi(Fila *p);

int buscaNaFrente(Fila *p, info *reg);

int buscaNaCauda(Fila *p, info *reg);

int testaVazia(Fila *p);

int testaCheia(Fila *p);

int reinicializa(Fila *p);

int enfileira(Fila *p, info *novo);

int desenfileira(Fila *p, info *reg);
```