ESTRUTURA DE DADOS - FILA

Estrutura de Dados I – ED1

FILA

- **FILA** é uma lista linear em que as inserções de novos nodos na estrutura são realizadas em uma das extremidades e as retiradas ou consultas aos nodos são realizadas na outra extremidade da lista.
- Graficamente tem-se:

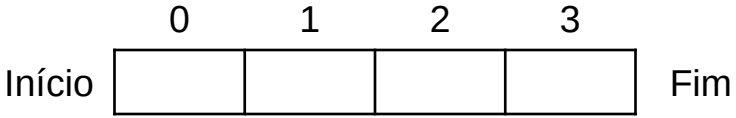
Fila

- No âmbito computacional as filas têm grande utilidade, estando presentes, por exemplo, nos controles de fluxo de dados numa rede de computadores ou de arquivos enviados a uma impressora compartilhada por vários usuários. Também é largamente utilizada em sistemas e algoritmos do dia-a-dia toda a vez que é preciso representar uma fila.

- Para a implementação dos algoritmos que manipulam uma fila com uso de memória estática, necessita-se de variáveis que fazem controle do tamanho da área onde a fila será implementada, de onde estão as extremidades *inicio* e *fim*.

Fila

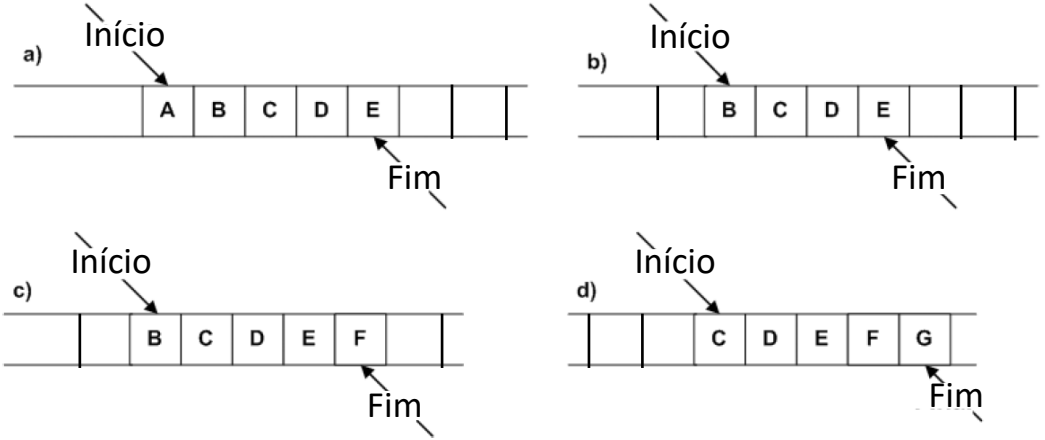
- As extremidades de uma fila são chamadas de Início e Fim:



- Uma vez que as inclusões são realizadas em uma extremidade e as retiradas na outra extremidade, o primeiro elemento inserido numa fila sempre será o primeiro a ser retirado, motivo pelo qual as filas são também chamadas de listas **FIFO (do inglês “First-Input First-Output”).**

Fila

a) Fila F, com elementos A, B, C, D, E
b) Desenfileirar(F) / dequeue(F)
c) Enfileirar(F, “F”) / enqueue(F, 'F')
d) Desenfileirar(F) && Enfileirar(F, “G”)



Fila

- enqueue(F, a)
- enqueue(F, b)
- enqueue(F, c)
- enqueue(F, d)
- dequeue(F)
- dequeue(F)
- enqueue(F, e)
- enqueue(F, f)
- dequeue(F)
- dequeue(F)
- dequeue(F)
- dequeue(F)

Implementaçã estática

Fila - Estática

- *Uma Fila vai usar uma estrutura composta de dados.*
- *Um vetor do tipo de dados que irá abrigar. Sobre este vetor se deverá criar uma estrutura lógica de controle para que este passe a funcionar como uma fila. Ou seja, somente incluir dados no FIM da fila e retirar dados do INÍCIO da fila e respeitar o tamanho da estrutura que abriga a base (início e fim da estrutura) Assim, é preciso ter variáveis que controlarão esses limites.*

Fila - Estática

- Com a utilização de memória estática, a inclusão de um novo nodo na fila é, na verdade, a ocupação de um nodo e não o acréscimo de um novo nodo na fila.
- Da mesma forma, a retirada é a liberação de um nodo e não a exclusão de um nodo da fila.

Fila - Estática

- Para o controle do tamanho e das extremidades da fila utilizam-se a s seguinte estrutura e variáveis:
  - **tam** : tamanho do vetor de dados
  - **total**: total de elementos armazenados na fila
  - **inicio** : ponteiro para o elemento armazenado no inicio da. Representa a extremidade INÍCIO da fila, onde está o primeiro elemento inserido na fila.
  - **fim**: ponteiro para o fim da fila (posição do vetor onde será armazenado o próximo elemento), a variável possui o índice do nodo posterior ao último elemento inserido na fila. Representa a extremidade FIM da fila.
  - **dados**: ponteiro para o vetor que será alocado para armazenar os dados.

```
struct Fila
{
    int tam;
    int total;
    int inicio;
    int fim;
    int *dados;
};
```

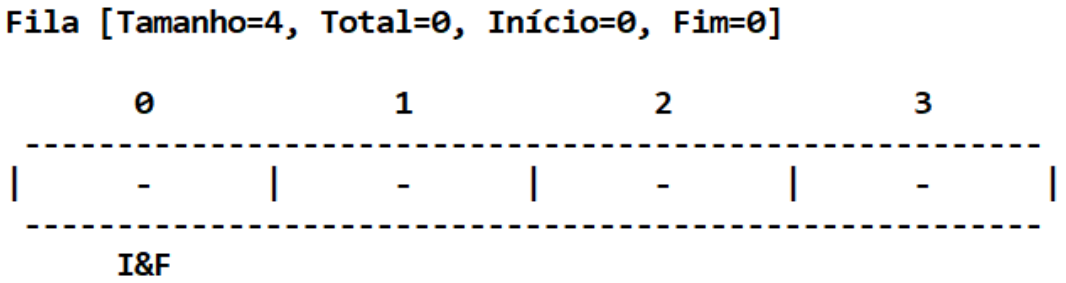
Fila - Estática

- Inicialmente, quando criada uma fila, ela estará vazia, ou seja sem elementos inseridos. Neste caso, os valores iniciais das variáveis de uma fila são:

```
void inicializaF(Fila *f, int tam) /// inicialização da fila
{
     f->tam = tam;
     f->inicio = 0;
     f->fim = 0;
     f->total = 0;
     f->dados = new int[tam];/// aloca memória para vetor
}
```

Fila - Estática

- Exemplo de uma fila vazia:
  - A fila suporta no máximo 4 elementos, pois o tamanho do vetor de dados é 4;
  - A fila está vazia, ou seja, total=0, nenhum elemento foi inserido!
  - Os ponteiros início e fim apontam para a posição zero do vetor:
    - O próximo elemento a ser inserido na estrutura é no final da fila, ou seja, posição zero.
    - O próximo elemento a ser retirado está no início da fila , ou seja, posição zero. Neste caso, quando a fila está vazia (total=0), não é permitido a retirada de elementos.



Fila - Estática

- Existem dois momentos especiais em uma fila:
  - Quando ela está vazia (nenhum elemento inserido)
  - Quando ela está cheia

```
bool vaziaF(Fila *f)// fila vazia
{
    if (f->total == 0)
        return true;
    else
        return false;
}
```

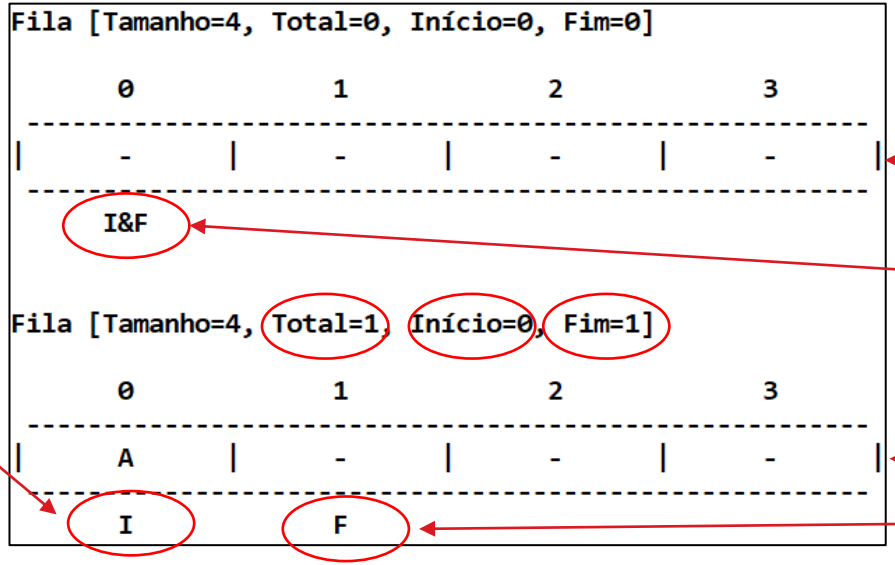
```
bool cheiaF(Fila *f)// fila cheia
{
    if (f->total == f->tam)
        return true;
    else
        return false;
}
```

Fila - Estática

- **Fazendo operações na Fila**
  - Inicialização: prepara a estrutura para armazenar a fila;
  - Enqueue (enfilera): enfilera um elemento na fila
  - Dequeue (desenfileirar): desenfilera um elemento
  - Peek (espiar): lê o primeiro elemento da fila, mas não o retira

Fila - Estática

- **Enfileirar**
  - Fila de tamanho 4, inicialmente vazia
  - Inserir o elemento “A” na fila



Fila vazia

Quando a fila está vazia, os ponteiros de INÍCIO e FIM ficam na mesma posição

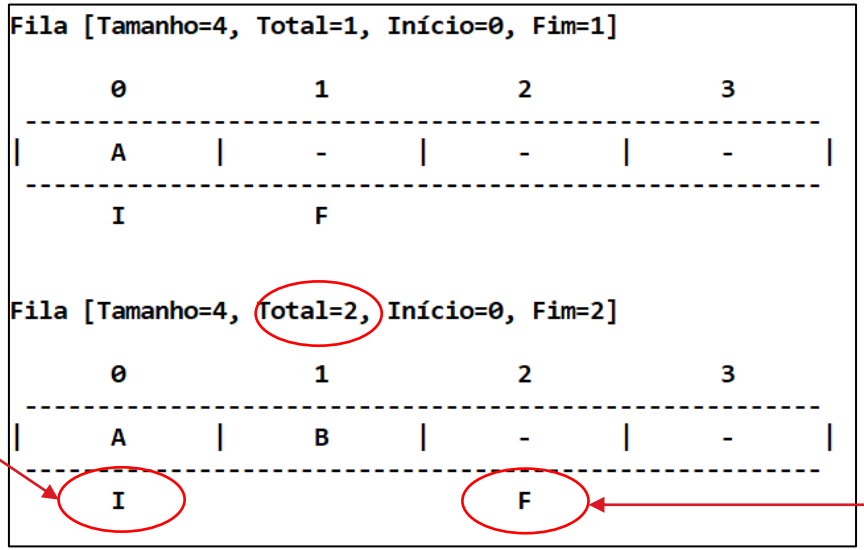
Posição para retirar um elemento do INÍCIO da fila

Fila com o elemento “A” inserido!

Posição para adicionar o próximo elemento no FIM da fila!

Fila - Estática

- **Enfileirar**
  - Inserir o elemento “B” na fila



Fila com um único elemento

O início da fila permanece o mesmo

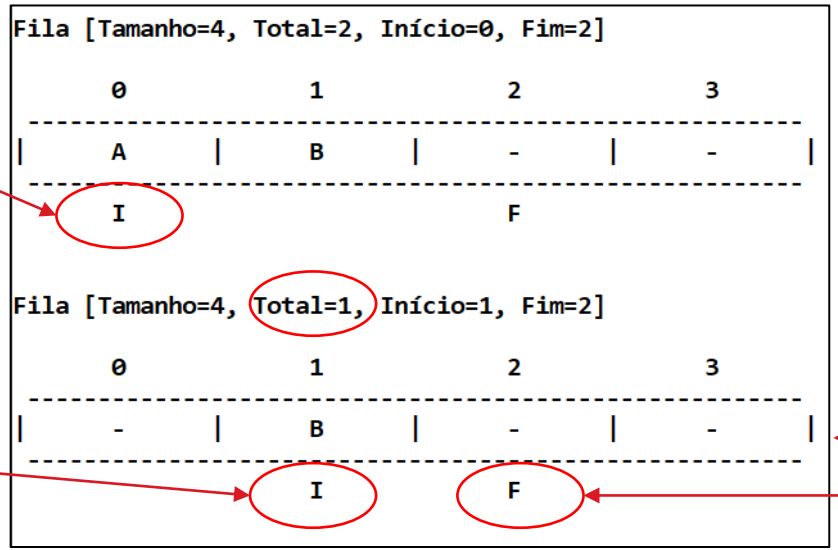
Fila com o elemento “B” inserido!

Posição para adicionar o próximo elemento no FIM da fila!

Fila - Estática

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “A”

O elemento do início fila será retirado.



Fila com os elementos “A” e “B”

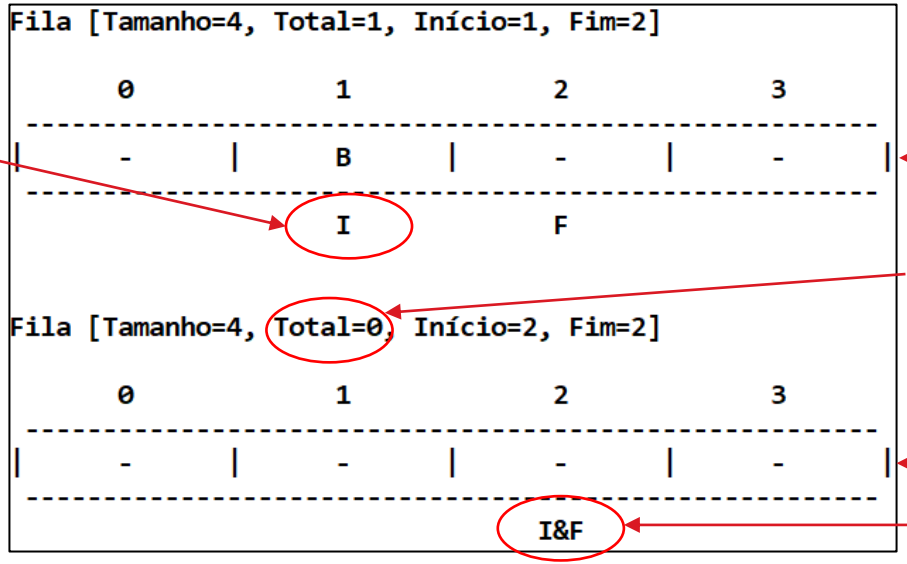
O elemento “B” passa para o INÍCIO da fila.

Fila com o elemento “A” retirado.

Posição para adicionar o próximo elemento no FIM da fila!

Fila - Estática

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “B” -> fila ficará vazia!



O elemento do início fila será retirado.

Fila com o elemento “B”

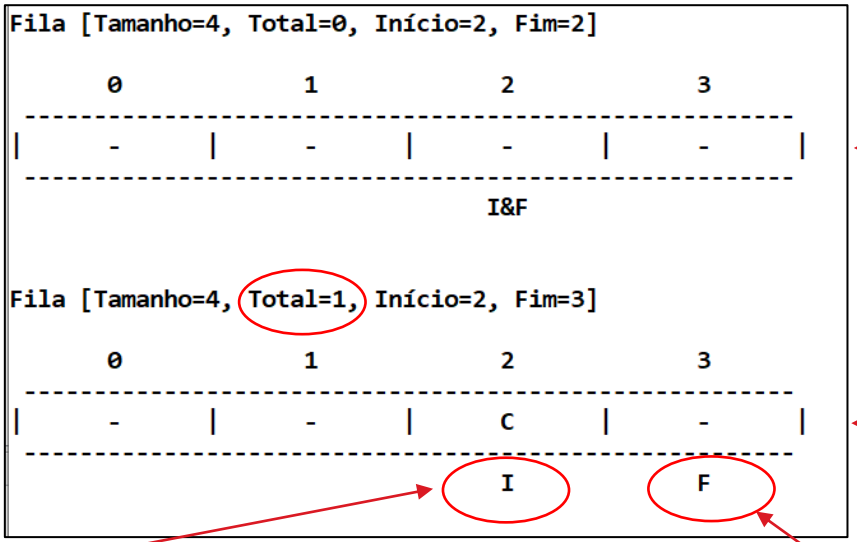
Fila vazia!

Fila com o elemento “B” retirado. Fila vazia.

Quando a fila está vazia, os ponteiros de INÍCIO e FIM ficam na mesma posição

Fila - Estática

- **Enfileirar**
  - Inserir o elemento “C” na fila



Fila vazia

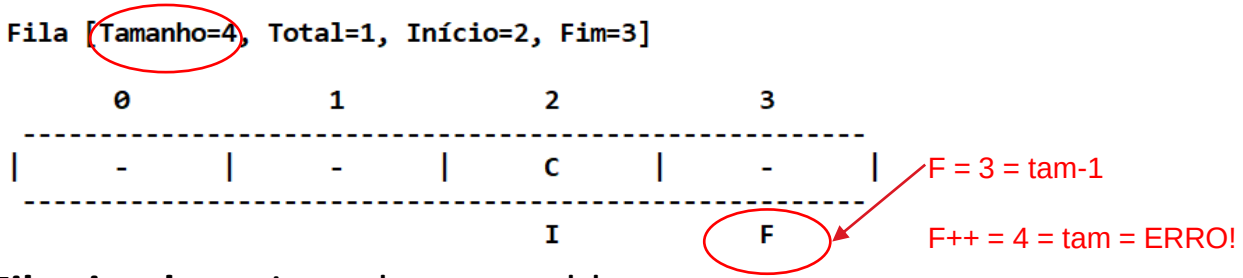
Fila com o elemento “C” inserido.

Posição para retirar um elemento do INÍCIO da fila

Posição para adicionar o próximo elemento no FIM da fila!

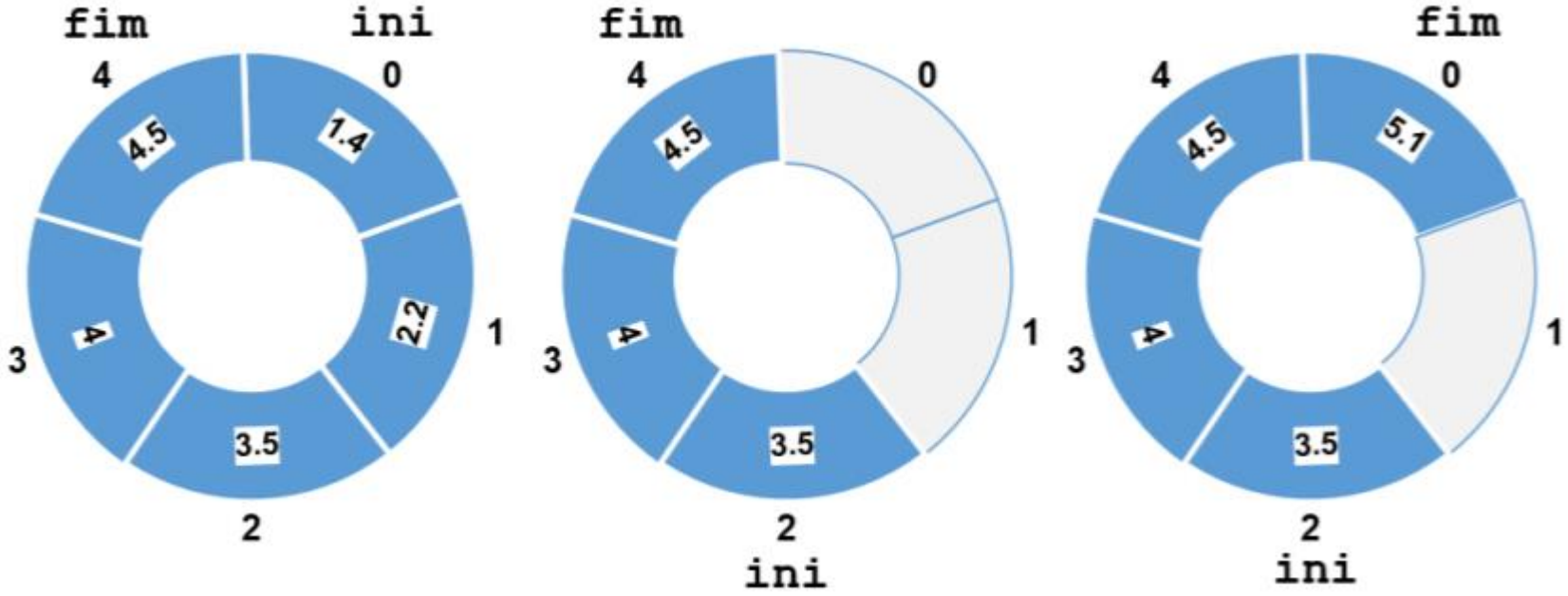
Fila - Estática

- **Enfileirar**
  - Na fila abaixo, há três posições livres no vetor de dados (total=1).
  - É possível inserir um novo elemento na fila ??
    - A resposta é **não**, pois se um novo elemento for inserido, o ponteiro para o final da fila "F" será incrementado para o valor 4, ou seja, vai exceder o tamanho do vetor!



  - **Fila circular** vai resolver o problema
    - Durante a inclusão, quando a fila não está cheia e o ponteiro FIM está última posição do vetor (F == tam – 1 && total< tam – 1) , F=0, ou seja, o final da fila passa a ser o índice 0 do vetor.
    - Durante a retirada, quando a fila não está vazia e o ponteiro INÍCIO está última posição do vetor (I == tam – 1 && total>0) , I=0, ou seja, o início da fila passa a ser o índice 0 do vetor.

Fila circular



Fila Circular

- **Enfileirar**

```
//Enqueue
bool enfileiraF(Fila *f, int valor) /// incluir valor na fila
{
    if (verificaInicializacaoF(f)==false || cheiaF(f) == true)
        return false;

    //inserir no final da fila
    f->dados[f->fim] = valor;

    //incrementa a quantidade de elementos armazenados na fila
    f->total++;

    //avança o ponteiro fim
    f->fim++;

    if(f->fim>=f->tam)
       f->fim = 0; //circular

    return true;
}
```

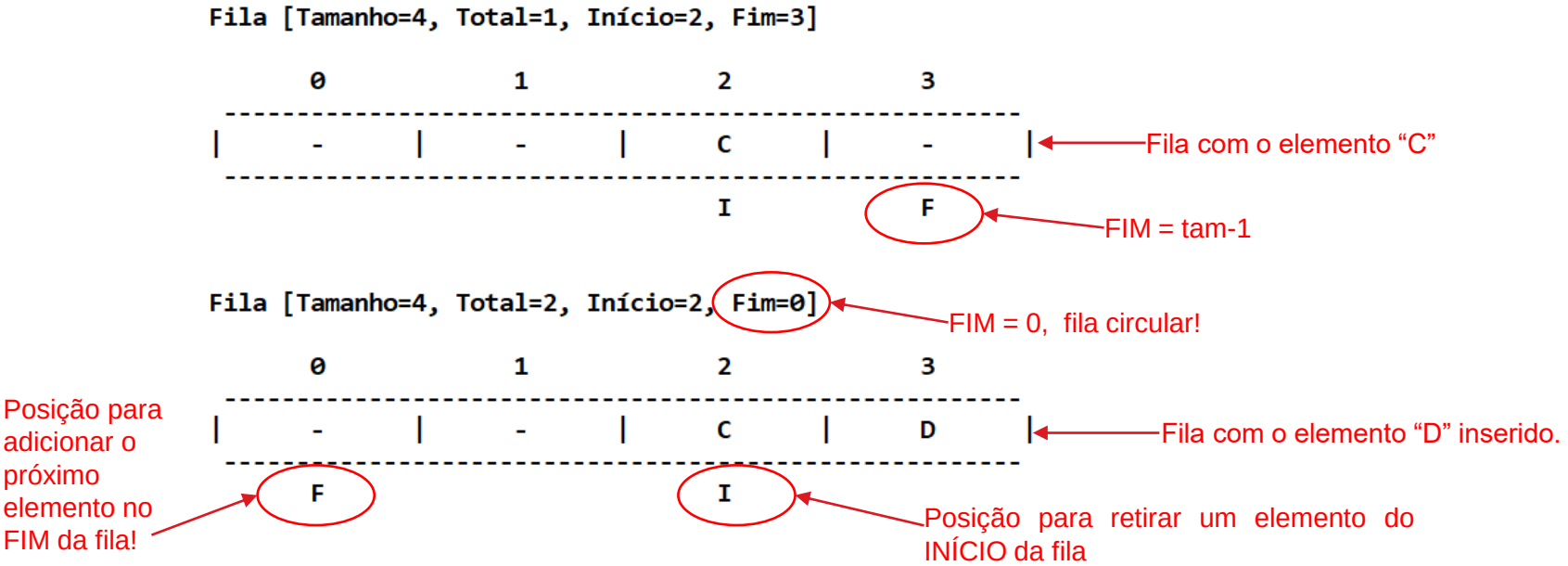
Fila Circular

- **Desenfileirar**

```
int desenfileiraF(Fila *f)   //retirar da fila
{
    int valor = 0;

    if (vaziaF(f) == true) // retorna 0 se fila vazia
        return valor;

    //remover do inicio da fila
    valor = f->dados[f->inicio];

    //diminui a quantidade de elementos armazenados na fila
    f->total--;

    //avança o ponteiro início
    f->inicio++;

    if(f->inicio>=f->tam)
       f->inicio = 0; //circular

    return valor;
}
```

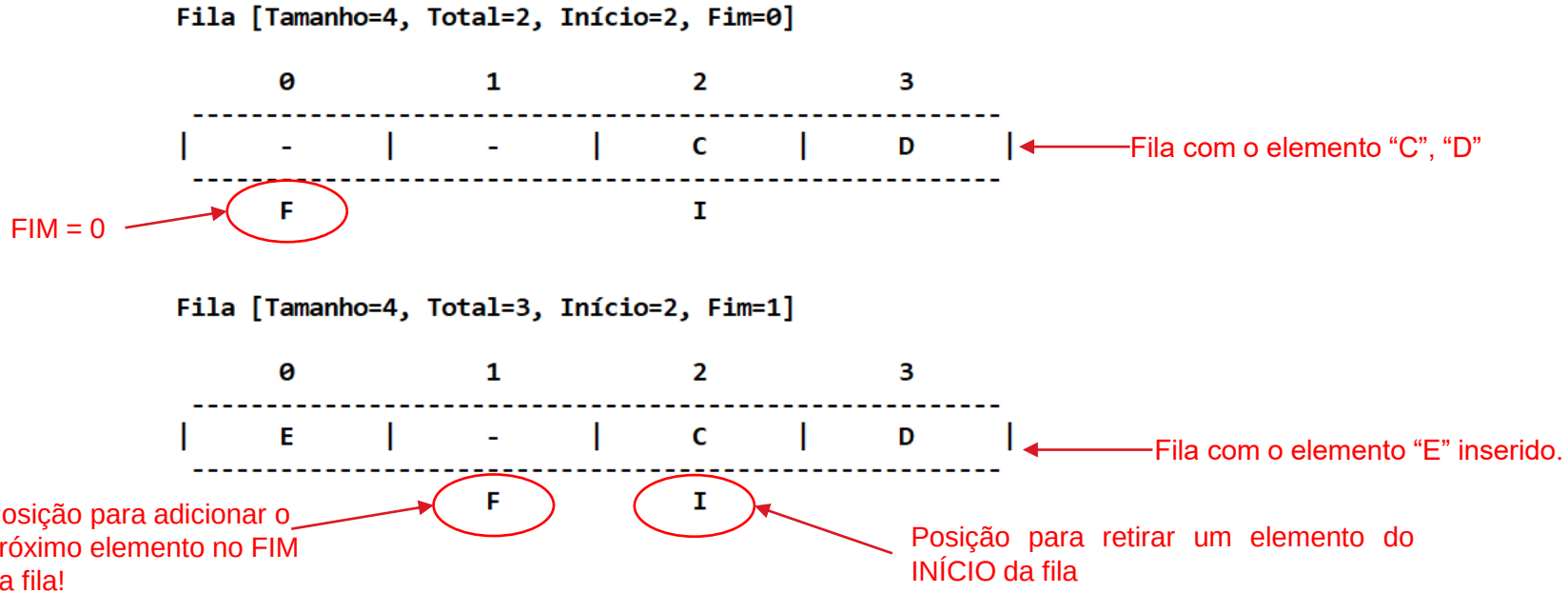
Fila Circular

- **Enfileirar**
  - Inserir o elemento “D” na fila



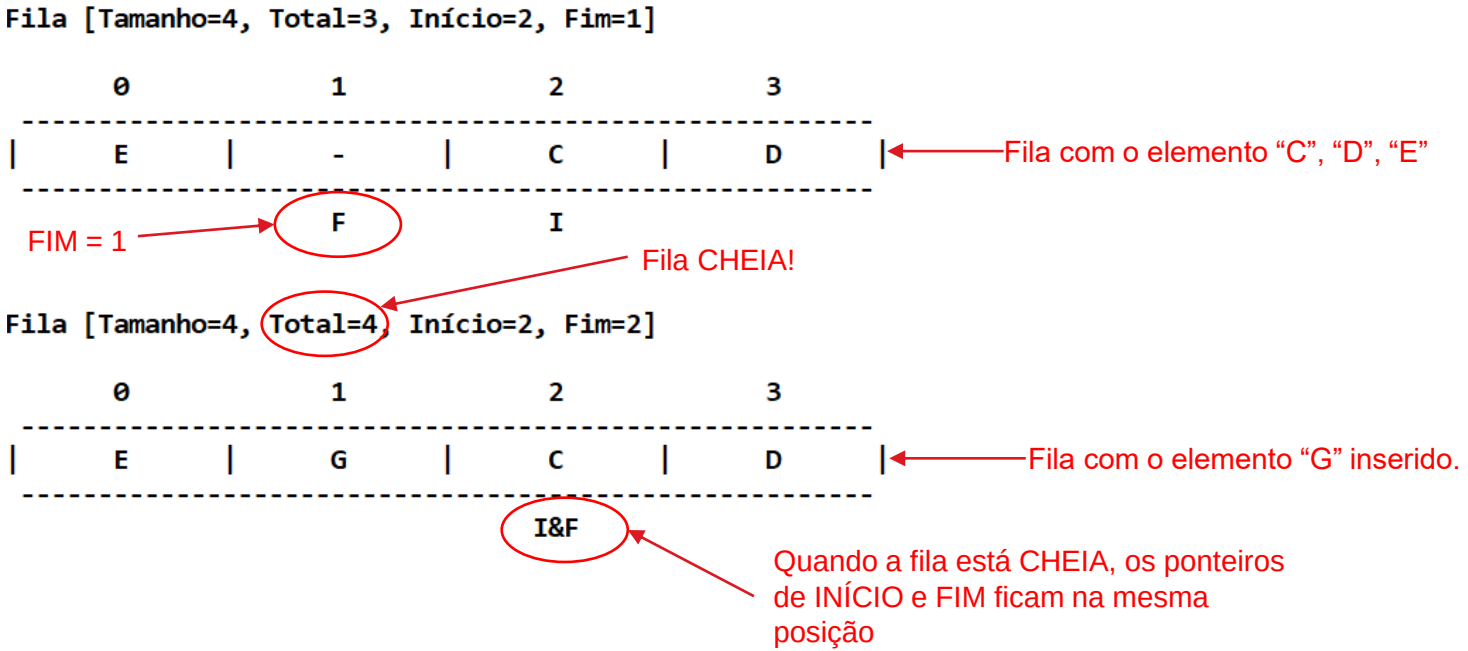
Fila Circular

- **Enfileirar**
  - Inserir o elemento “E” na fila



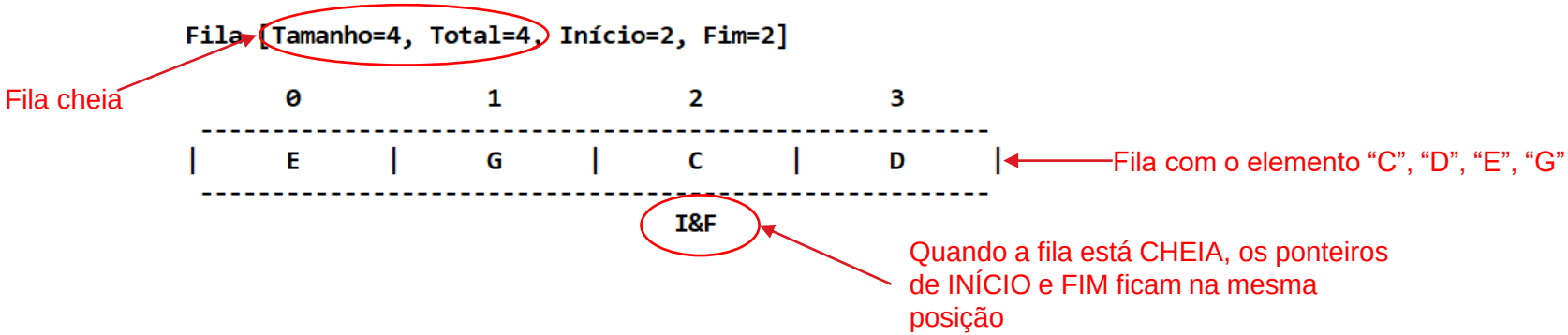
Fila Circular

- **Enfileirar**
  - Inserir o elemento “G” na fila



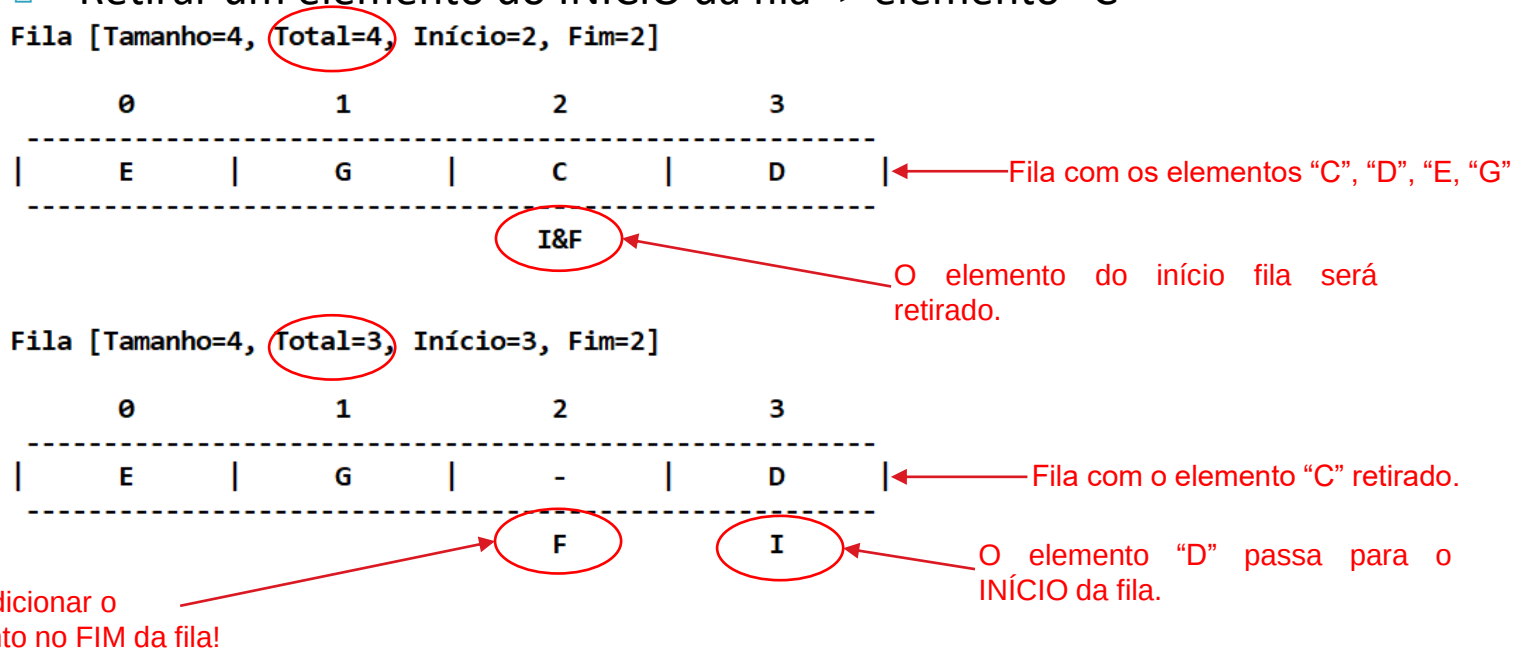
Fila Circular

- **Enfileirar**
  - É possível Inserir um novo elemento na fila?
    - A resposta é NÃO, pois a fila está cheia (total=tam), evitando que as informações da fila sejam perdidas, impedindo que um novo elemento sobrescreva um nó da fila.



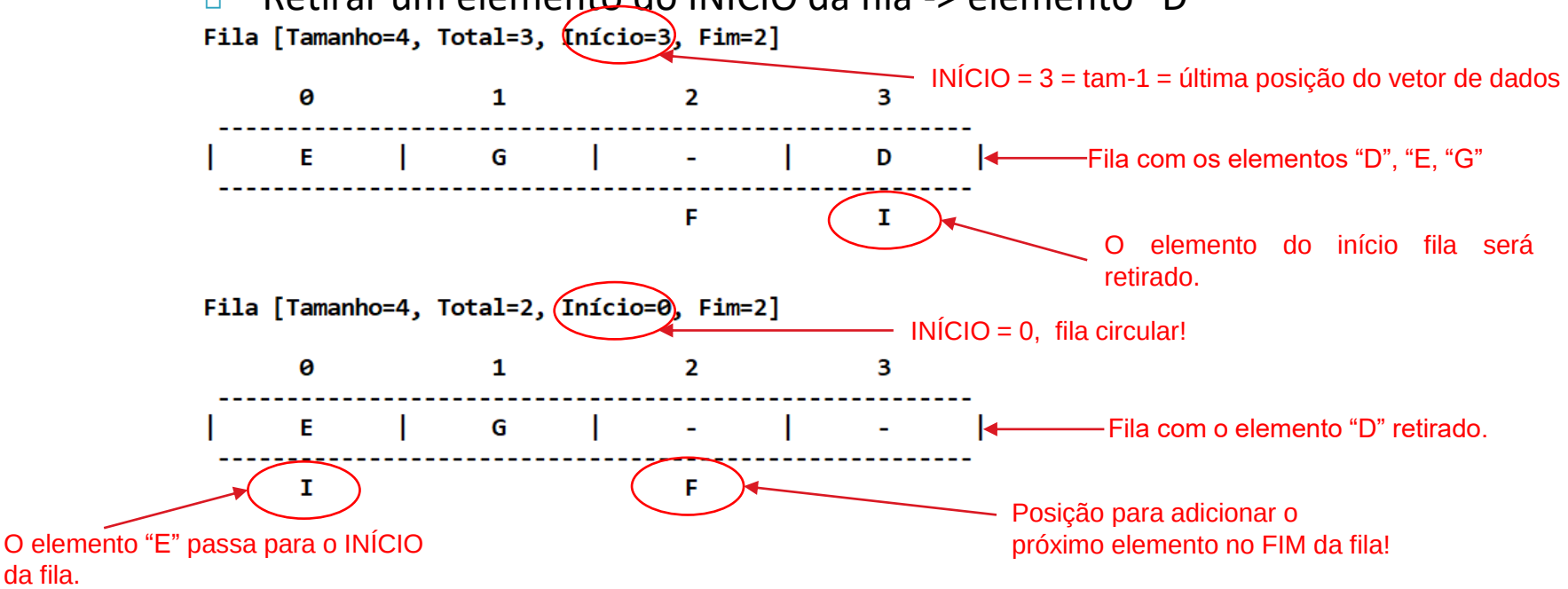
Fila Circular

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “C”



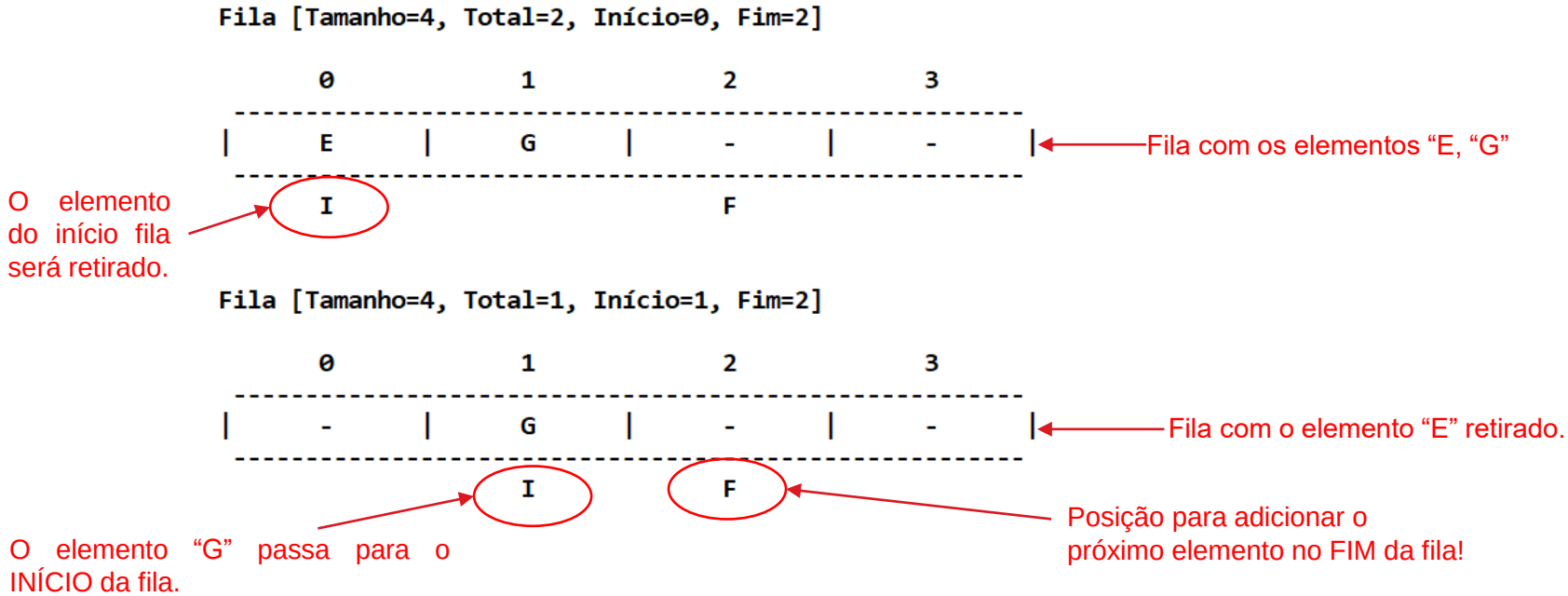
Fila Circular

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “D”



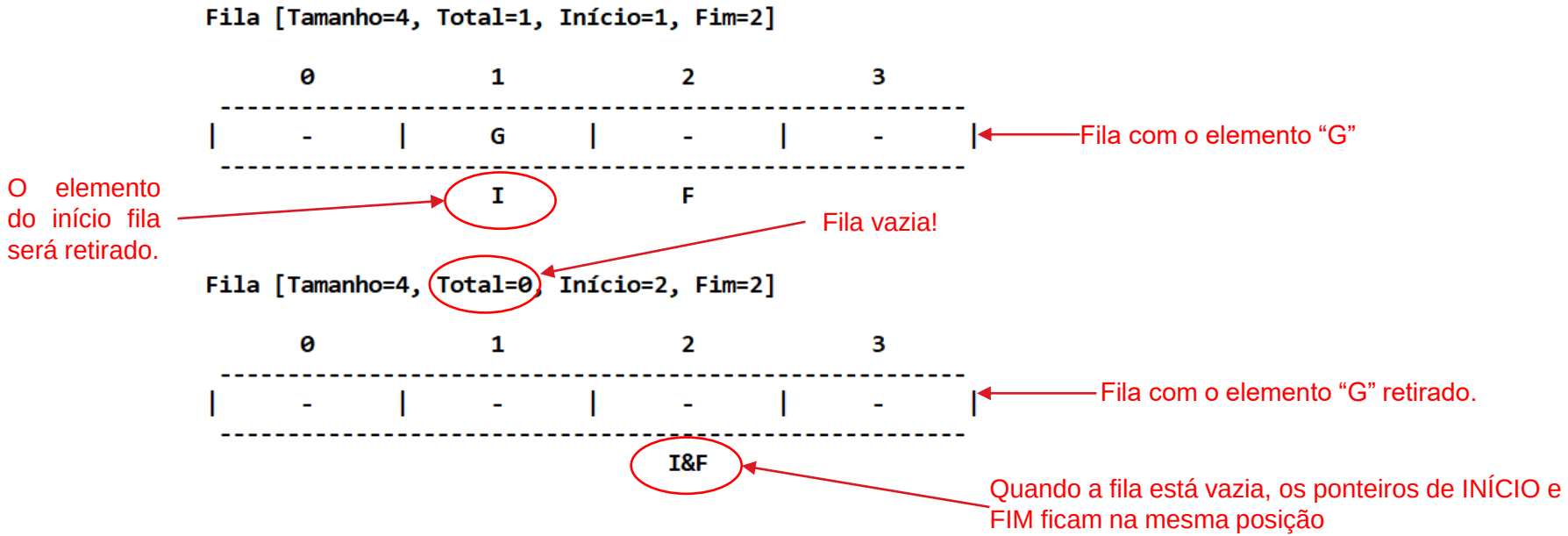
Fila Circular

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “E”



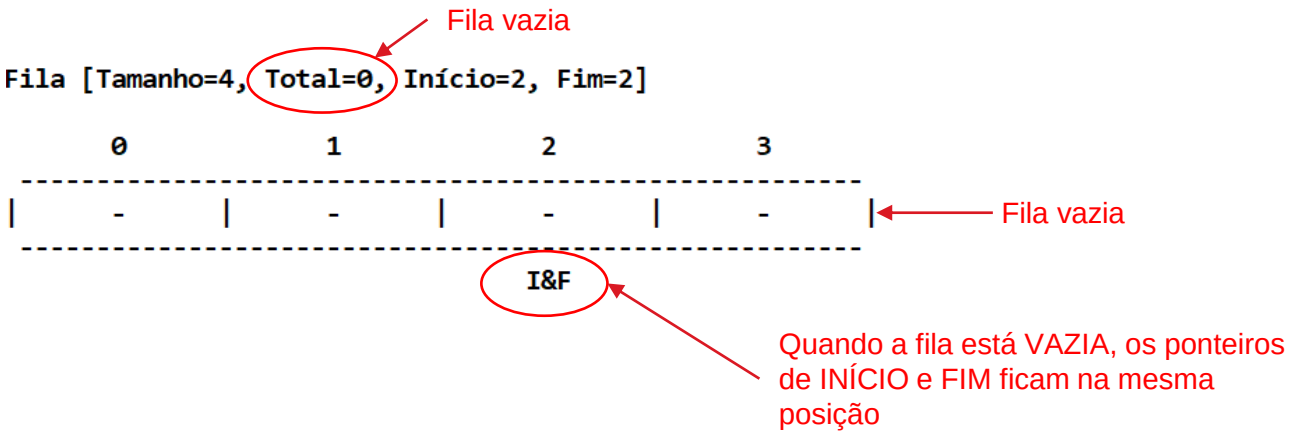
Fila Circular

- **Desenfileirar**
  - Retirar um elemento do INÍCIO da fila -> elemento “G” -> fila ficará vazia!



Fila Circular

- **Desenfileirar**
  - É possível retirar um elemento da fila?
    - A resposta é NÃO, pois a fila está vazia (total=0)



Fila Circular

- **Espiar / Peek**

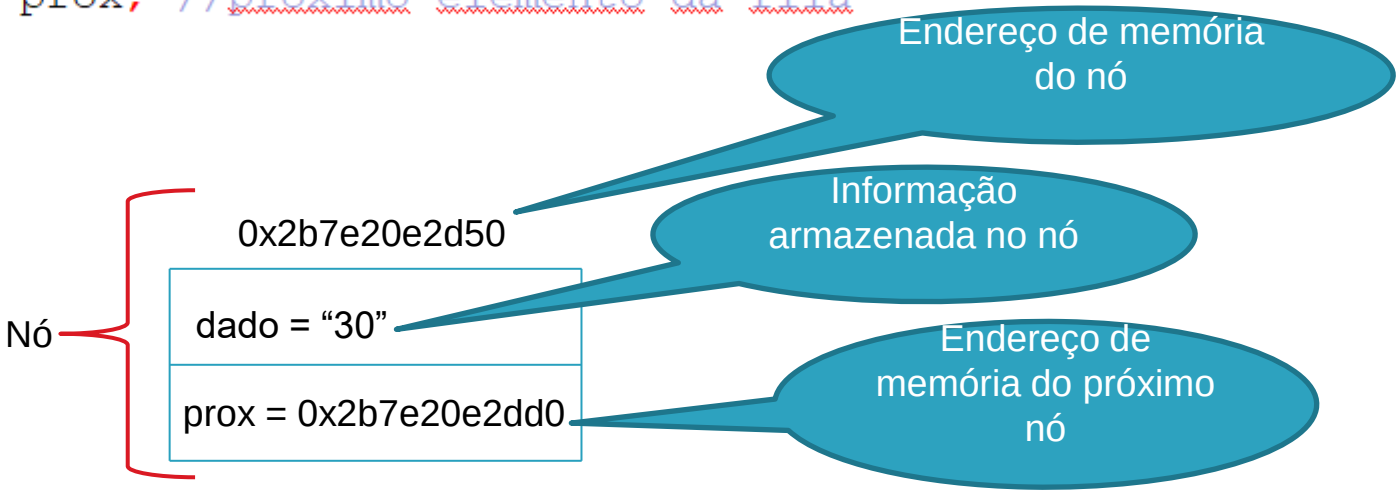
```
//Peek
int espiaF(Fila *f) ///peek
{
    int valor = 0;

    if(vaziaF(f)==true)// retorna 0 se fila vazia
        return valor;

    valor = f->dados[f->inicio];
    return valor;
}
```

Implementaçã dinâmica

Fila dinâmica

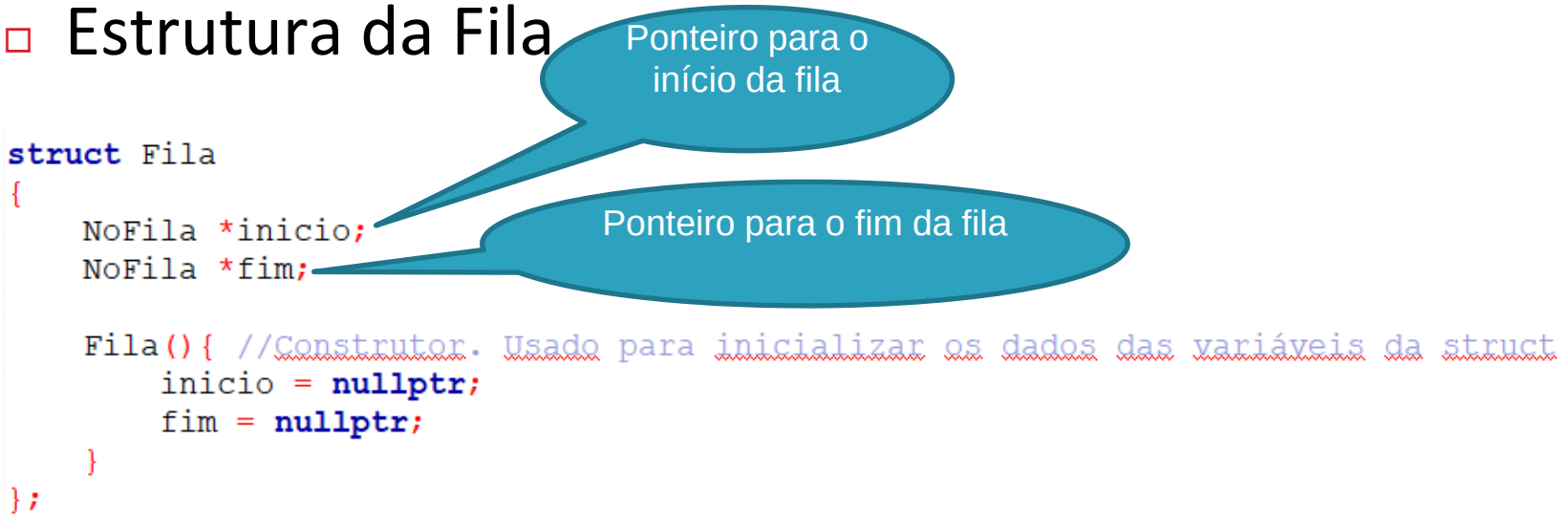
- Estrutura do nó/elemento

```
struct NoFila
{
     int dado; //informacao do nó
     NoFila *prox; //proximo elemento da fila
};
```



Fila dinâmica

- Estrutura da Fila




```
struct Fila
{
    NoFila *inicio;
    NoFila *fim;

    Fila(){ //Construtor. Usado para inicializar os dados das variáveis da struct
        inicio = nullptr;
        fim = nullptr;
    }
};
```

Fila dinâmica

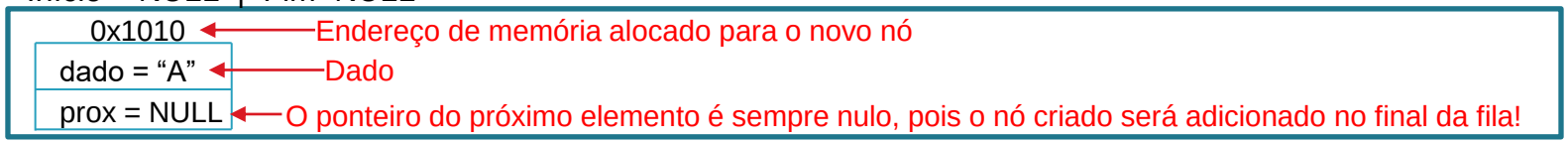
- Fila vazia

Início = NULL | Fim=NULL

Fila dinâmica: enfileirar (0x1010)

- Passo 1 – Criar um nó (0x1010) e gravar o seu dado

Início = NULL | Fim=NULL



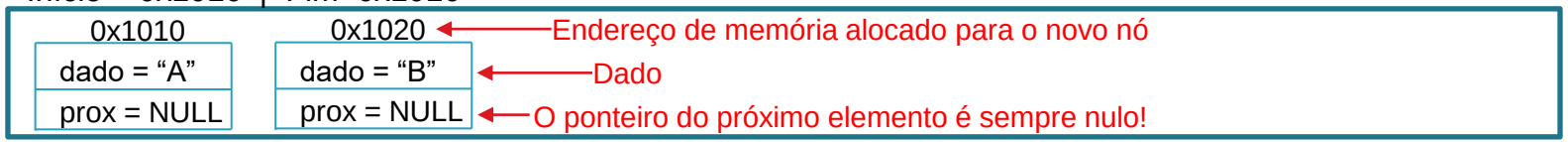
- Passo 2 – Atualizar os ponteiros *Início* e *Fim*.
  a) O ponteiro *Fim* sempre aponta para o nó recém criado, pois ele é o último da fila.
  b) Se a fila estava vazia (somente neste caso), o ponteiro *Início* aponta para o recém criado, pois ele é o único da fila.

Início = 0x1010 | Fim=0x1010

Fila dinâmica: enfileirar (0x1020)

- Passo 1 – Criar um nó(0x1020) e gravar o seu dado

Início = 0x1010 | Fim=0x1010



- Passo 2 – Atualizar os ponteiros *Início* e *Fim*.
  a) Se a fila não está vazia, o ponteiro *prox* do último elemento da fila (0x1010) deve apontar para o nó recém criado, pois ele é o único da fila.
  b) O ponteiro *Fim* sempre aponta para o nó recém criado, pois ele é o último da fila.

Início = 0x1010 | Fim=0x1020

Fila dinâmica: enfileirar


```
//Enqueue
bool enfileiraF(Fila *f, int dado)
{
    NoFila *novo = new NoFila();
    if (novo == NULL) /// não conseguiu alocar memória (novo == NULL)
        return false;

    novo->dado = dado;
    novo->prox = nullptr;
    if (f->inicio == NULL) //verifica se a fila está vazia, ou seja, (f->inicio == NULL)
        f->inicio = novo;
    else
        f->fim->prox = novo;

    f->fim = novo;
    return true;
}
```

Fila dinâmica: desenfileirar (0x1010)

- Passo 1: Desenfileirar (0x1010)
  a) Ler o valor do primeiro nó, o qual que será retirado da fila (“A”).
  b) Guardar numa variável auxiliar o ponteiro do nó a ser retirado(0x1010).

- Passo 2: Atualizar o ponteiro *Início*
  a) O ponteiro *Início* deve apontar para o ponteiro prox do elemento que está sendo retirado.

Fila dinâmica: desenfileirar (0x1010)

Início = **0x1020** | Fim=0x1020

- Passo 3: Desalocar memória e atualizar o ponteiro *Fim*
  a) Desalocar memória para o ponteiro guardado na variável auxiliar (0x1010).
  b) Se a fila ficou vazia o ponteiro *Fim* deve ser nulo (não é o caso deste exemplo).

Início = 0x1020 | Fim=0x1020

Fila dinâmica: desenfileirar (0x1020)

- Passo 1: Desenfileirar (0x1020)
  a) Ler o valor do primeiro nó, o qual que será retirado da fila (“B”).
  b) Guardar numa variável auxiliar o ponteiro do nó a ser retirado(0x1020).

- Passo 2: Atualizar o ponteiro *Início*
  a) O ponteiro *Início* deve apontar para o ponteiro prox do elemento que está sendo retirado.

Fila dinâmica: desenfileirar (0x1020)

Início = **NULL** | Fim=0x1020

- Passo 3: Desalocar memória e atualizar o ponteiro *Fim*
  a) Desalocar memória para o ponteiro guardado na variável auxiliar (0x1020).
  b) Se a fila ficou **vazia** o ponteiro *Fim* deve ser nulo.

Início = NULL | Fim=**NULL**

Fila dinâmica: desenfileirar


```
//Dequeue
int desenfileiraF(Fila *f)
{
    int dado = 0;

    // se não estiver vazia, retira valor
    if (vaziaF(f) == false) //verifica se a fila não está vazia, ou seja, (f->inicio != NULL)
    {
        dado = f->inicio->dado; //pega o dado armazenado no primeiro nó
        NoFila *apagar = f->inicio; //guarda o primeiro nó em uma variável auxiliar;
        f->inicio = f->inicio->prox; // atualiza o início da fila
        delete apagar;  // libera a memória

        if (f->inicio == NULL) //verifica se a fila está vazia, ou seja, (f->inicio == NULL)
            f->fim = nullptr;
    }

    return dado;
}
```

Fila dinâmica: fila vazia


```
bool vaziaF(Fila *f)
{
    if (!f->inicio) //verifica se a fila está vazia, ou seja, (f->inicio == NULL)
        return true;
    else
        return false;
}
```

Fila dinâmica: espia / peek


```
//peek
int espiaF(Fila* f)
{
    int dado = 0;

    if (vaziaF(f) == false) //verifica se a fila não está vazia, ou seja, (f->inicio != NULL)
    {
        dado = f->inicio->dado;
    }

    return dado;
}
```

Fila dinâmica: mostra


```
//show
void mostraF(Fila *f)
{
    cout << "Fila: ";

    if(vaziaF(f) == false) //verifica se a fila não está vazia
    {

        cout << "[";

        NoFila *no = f->inicio;
        while (no != NULL) //faça enquanto (no != NULL)
        {
            cout << no->dado;
            no = no->prox;

            if(no != NULL) //verifica se o próximo nó não é nulo (no != NULL)
                cout << ", ";
        }
        cout << "]" << endl;
    }
    else
        cout << "vazia!\n";
}
```

Fila dinâmica: busca


```
// retorna true se o valor existe na fila
// retorna false se o valor não existe na fila
bool buscaF(Fila *f, int dado)
{

    NoFila *no = f->inicio;
    while (no != NULL) //faça enquanto (no != NULL)
    {
        if(no->dado == dado)
            return true;

        no = no->prox;
    }

    return false;
}
```

Fila dinâmica: desalocar memória


```
void destroiF(Fila *f)
{

    NoFila *no = f->inicio;
    while (no != NULL) //faça enquanto (no != NULL)
    {
        NoFila *apagar = no; //guarda o nó em uma variável auxiliar;

        no = no->prox;

        delete apagar;
    }

    f->inicio = nullptr;
    f->fim = nullptr;
}
```