

# SCC 223 – ESTRUTURAS DE DADOS 1

## FILAS

---

Profa. Elaine Parros Machado de Sousa



Aula baseada em material do professor Rudinei Goularte.

# Conteúdo

- Filas (*Queues*)
- Deques (*Double Ended Queues*)

# Fila

- O quê é?
- Para quê serve?




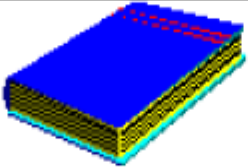
# Exemplo de Aplicação

- **Problema:** automação de uma biblioteca
  - todos os livros devem ser cadastrados
  - o sistema deve informar se um livro está disponível ou não nas estantes
  - caso o livro não esteja disponível, o usuário pode aguardar em uma **fila de espera**
    - quando o livro for devolvido, o primeiro da fila de espera pode retirá-lo
- **Sua tarefa:** desenvolver esse sistema

# Modelagem do Problema

- **1º passo: abstração**

- identificar os elementos do mundo real que são relevantes para a solução do problema

fila de espera para o livro	livros do acervo	disponível?
 último ---> <--- 1º	 trigonometria	não
 último ---> <--- 1º	 química inorgânica	não
fila vazia!	 estruturas de dados	sim

# Modelagem do Problema

- Elementos relevantes
  - um cadastro de livros
  - indicação da disponibilidade dos livros
  - uma fila de espera para cada livro, com indicação da ordem das pessoas
  - primeiro e último da fila
  - cadastro de pessoas: nome, endereço e telefone

# Modelagem do Problema

- **2º passo:** quais são as **operações** possíveis nas filas?
  - Entrar na fila
    - Quem entra, entra onde?
  - Sair da fila
    - Quem sai, sai de onde?
  - Outras?

# Fila (*Queue*)

- O que é?
  - **Estrutura para armazenar um conjunto de elementos, que funciona da seguinte maneira:**
    - ✓ novos elementos sempre entram no fim da fila
    - ✓ o único elemento que se pode retirar da fila em um dado momento é seu primeiro elemento



# Fila (*Queue*)

- Para que serve?
  - Modelar situações em que é preciso armazenar um **conjunto organizado de elementos**, no qual o **primeiro elemento a entrar** no conjunto será também **o primeiro elemento a sair** do conjunto, e assim por diante...
  - Política **FIFO**: *first in, first out*

# Aplicações

- Exemplos de aplicações de filas
  - Filas de espera e algoritmos de simulação
  - Controle, pelo sistema operacional, de recursos compartilhados: processador, memória, impressoras...
  - *Buffers* de Entrada/Saída
  - Estrutura de dados auxiliar em alguns algoritmos como a busca em largura em árvores
  - ...

## TAD Fila - exemplo

Operação	Fila	Resultado
cria(F)	1º da fila →	
entra(F,a)	1º da fila → <b>a</b>	
entra(F,b)	1º da fila → a, <b>b</b>	
entra(F,c)	1º da fila → a, b, <b>c</b>	
sai(F)	1º da fila → b, c	return ( <b>a</b> )
entra(F,d)	1º da fila → b, c, <b>d</b>	
sai(F)	1º da fila → c, d	return ( <b>b</b> )

# TAD Fila - exemplo

- Operações principais
  - **cria(F)**: cria uma fila **F** vazia
  - **entra(F,x)**: insere o elemento **x** no final da fila **F**.
    - retorna **true** se foi possível inserir, **false** caso contrário
  - **sai(F)**: remove o elemento no início de **F**, e retorna esse elemento.
    - retorna **NULL** se não foi possível remover

# TAD Fila - exemplo

- Operações auxiliares
  - **frente(F)**: retorna o 1º elemento de **F**, sem remover
  - **tamanho(F)**: retorna o número de elementos em **F**
  - **vazia(F)**: indica se a fila **F** está vazia
  - **cheia(F)**: indica se a fila **F** está cheia
    - útil para implementações com alocação estática de memória

# Implementação

- Abordagens mais comuns de implementação:
  - 1) **Sequencial** com alocação **estática** de memória
  - 2) **Encadeada** com alocação **dinâmica** de memória

# Implementação – Sequencial Estática

- **Início:** aponta para/indica o primeiro da fila, ou seja, **o primeiro elemento a sair**
- **Fim:** aponta para/indica o fim da fila, ou seja, **posição onde o próximo elemento entrará**



# Implementação – Sequencial Estática

- Qual o estado inicial da fila (quando a fila é criada)?
- Qual a condição para fila vazia?
- Qual a condição para fila cheia?





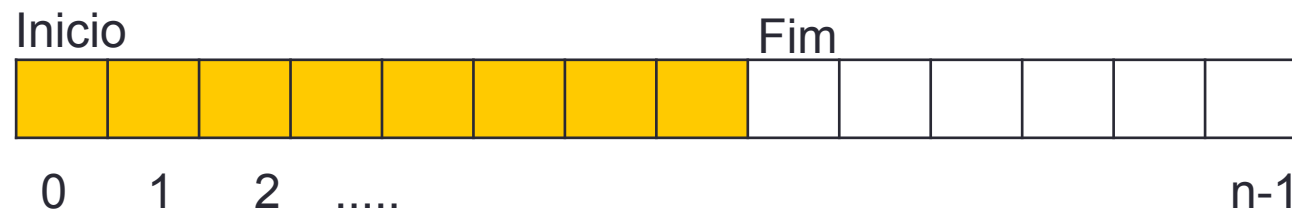
# Exemplo de uso

Vetor com 6 posições:

- `cria(F)`
  - `Início = 0, Fim = 0`
- `entra(F,a), entra(F,b), entra(F,c)`
  - Qual o estado de F, Início e Fim?
- `entra(F,d), entra(F,e), entra(F,f)`
  - `cheia(F) == TRUE`
- `sai(F), sai(F)`

# Implementação – Sequencial Estática

- Inserção
  - complexidade?
  - nas inserções (entra), o contador **Fim** é a incrementado  $\Rightarrow O(1)$



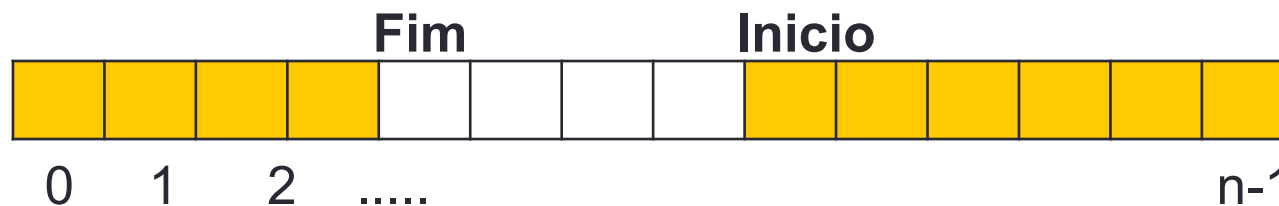
- Remoção
  - complexidade?
  - nas remoções (sai), todos os elementos são deslocados  $\Rightarrow O(n)$
  - É possível melhorar isso?

## Implementação – Sequencial Estática CIRCULAR

- **Solução:** fazer com que o início não seja fixo na primeira posição do vetor
  - permitir que **Fim** volte para o início do vetor quando esse atingir o final do vetor
- Implementação de **fila circular**

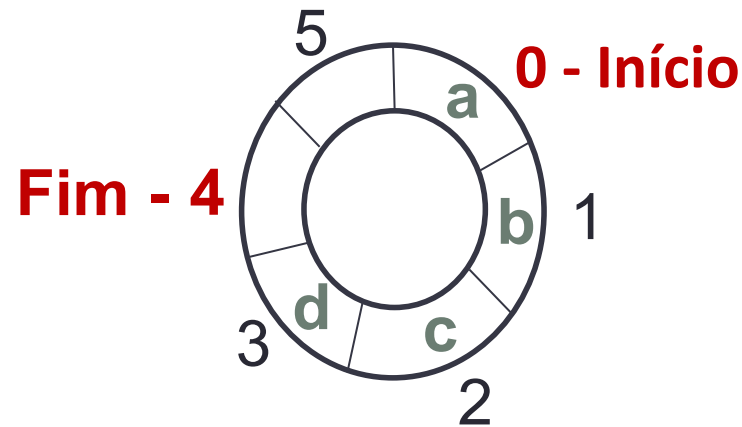
➤ Ex: fila circular com 4 posições vazias e

**Fim < Início**



## Implementação – Sequencial Estática CIRCULAR

- **Fila Circular** – pode ser vista como um “anel”

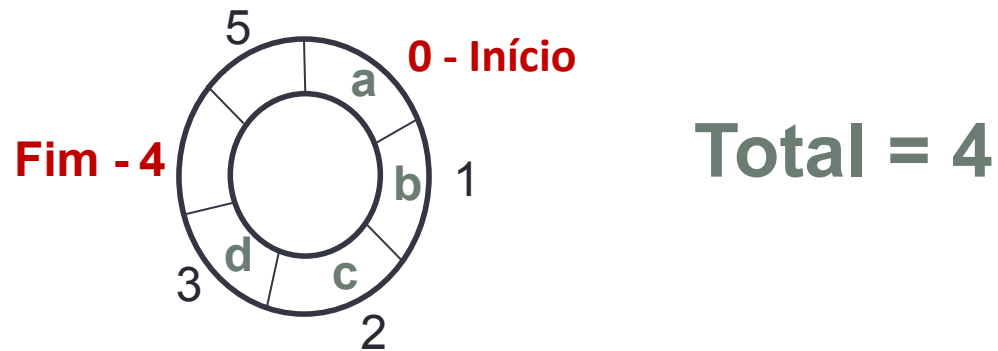


- Qual o estado inicial da fila (quando a fila é criada)?
- Qual a condição para fila vazia?
- Qual a condição para fila cheia?



## Implementação – Sequencial Estática CIRCULAR

- **Solução:** campo extra para guardar número de elementos



- Qual o estado inicial da fila (quando a fila é criada)?
  - **Total=0, Início=0, Fim=0**
- Qual a condição para fila vazia?
  - **Total=0**
- Qual a condição para fila cheia?
  - **Total=tamanho do vetor**

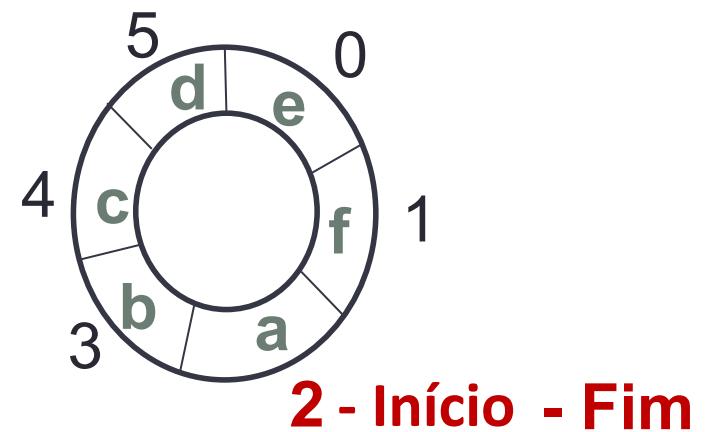
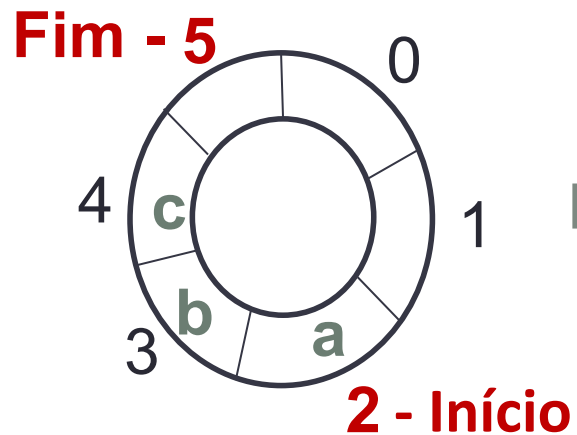
# Implementação – Sequencial Estática CIRCULAR

- Exemplo

- Inicio = 2,
- Fim = 5,
- Total = 3



- entra(d),
- entra(e),
- entra(f)



**Total = 6**

# TAD Fila – Exemplo

(fila.h)

```
#define TAM 100
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
typedef struct fila_ FILA;
```

```
typedef int ITEM;
```

```
FILA *fila_cria(void);
```

```
boolean fila_entra(FILA *fila, ITEM item);
```

```
ITEM fila_sai(FILA *fila);
```

```
ITEM fila_busca(FILA *fila, int chave);
```

```
int fila_tamanho(FILA *fila);
```

```
...
```

# TAD Fila – Exemplo de Implementação Sequencial Circular

```
(fila.c)
```

```
...
```

```
#include "fila.h"
```

```
struct fila_{
```

```
    ITEM itens[TAM];
```

```
    int inicio;    /*posicao do 1o elemento da fila*/
```

```
    int fim;       /*posicao para inserção de elemento na fila*/
```

```
    int tamanho;
```

```
};
```

```
...
```



```
/*Cria logicamente uma fila, inicialmente vazia*/  
FILA *fila_cria(void){  
    /*alocação da fila - dinâmica por ser TAD*/  
    FILA *fila = (FILA *) malloc(sizeof(FILA));  
    if (fila!=NULL) {  
        fila->inicio = 0;  
        fila->fim = 0;  
        fila->tamanho = 0; /* fila vazia*/  
    }  
    return (fila);  
}
```

```
/*verifica se fila está cheia*/
```

```
int fila_cheia(FILA *fila) {  
    return (fila->tamanho == TAM);  
}
```

```
/*verifica se fila está vazia*/
```

```
int fila_vazia(FILA *fila) {  
    return (fila->tamanho == 0);  
}
```

```
/*inserção na fila - sempre no final*/
```

```
boolean fila_entra(FILA *fila, ITEM item){
```

```
    if (fila != NULL && (!fila_cheia(fila)) ){
```

```
        fila->itens[fila->fim] = item;
```

```
        fila->fim = (fila->fim + 1) % TAM;
```

```
        fila->tamanho ++;
```

```
        return(TRUE);
```

```
    }
```

```
    return(FALSE);
```

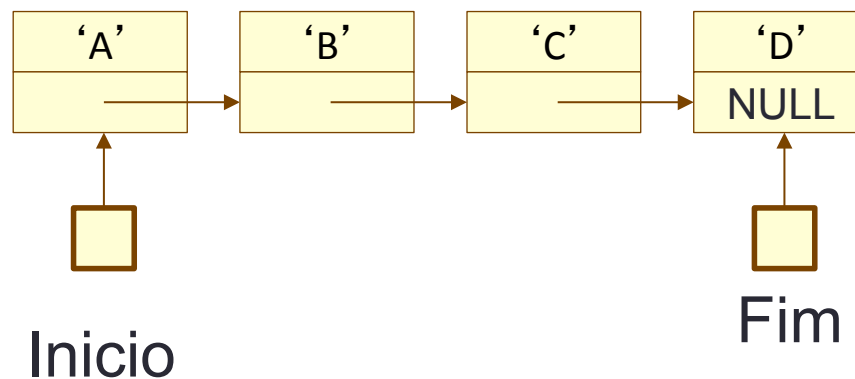
```
}
```

*/\*remoção da fila – sempre do início\*/*

```
ITEM fila_sai(FILA *fila){  
    if (fila != NULL && (!fila_vazia(fila)) ) {  
        ITEM ret = fila->itens[fila->inicio];  
        fila->inicio = (fila->inicio + 1) % TAM;  
        fila->tamanho --;  
        return (ret);  
    }  
    return (NULL);  
}
```

# Implementação – Encadeada Dinâmica

- Elementos armazenados em **nós** alocados **dinamicamente**
- **Ponteiros** indicam a localização do elemento seguinte da fila
  - possível manter dois ponteiros: um para o **início** e outro para o **final** da fila => acesso direto às posições de inserção (**entra**) e remoção (**sai**)



## Implementação – Encadeada Dinâmica

- As operações **inserir** e **remover** implementadas dinamicamente são bastante eficientes
  - complexidade?
  - ponteiros para início e fim devem ter valor NULL quando a fila estiver vazia

# Estática versus Dinâmica

Operação	Estática	Dinâmica
Criar fila	$O(1)$	$O(1)$
Apagar fila	$O(1)$	$O(n)$
Inserir (entrar)	$O(1)$	$O(1)$
Remover (sair)	$O(1)$ (circular)	$O(1)$
Frente	$O(1)$	$O(1)$
Vazia	$O(1)$	$O(1)$
Cheia	$O(1)$	$O(1)$
Tamanho	$O(1)$	$O(1)$ (c/ contador)

# Deque

- ***Double Ended Queue***
- Deques são estruturas similares às filas, que permitem **inserir** e **remover** de **ambos os extremos**



# TAD Deques

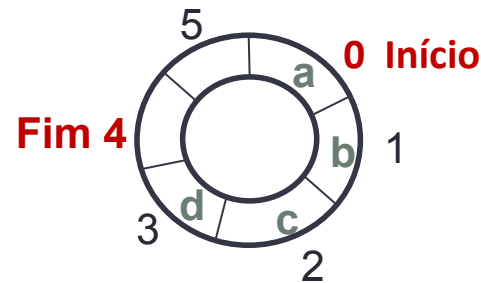
- Operações principais
  - **inserir\_inicio(D,x)**: insere o elemento **x** no início da deque **D**.
    - retorna **true** se foi possível inserir e **false** caso contrário
  - **inserir\_fim(D,x)**: insere o elemento **x** no final da deque **D**.
    - retorna **true** se foi possível inserir e **false** caso contrário
  - **remover\_inicio(D)**: remove o elemento no início de **D**, e retorna esse elemento.
    - retorna **null** se não foi possível remover
  - **remover\_fim(D)**: remove o elemento no final de **D**, e retorna esse elemento.
    - retorna **null** se não foi possível remover

# TAD Deques

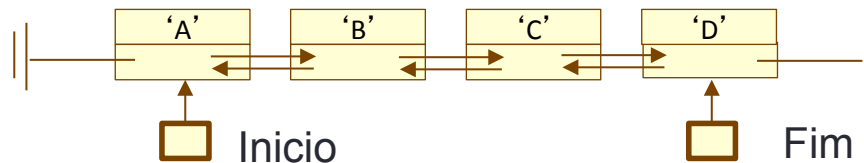
- Operações auxiliares
  - **primeiro(D)**: retorna o elemento no início de **D**.
    - retorna **null** se o elemento não existe
  - **ultimo(D)**: retorna o elemento no final de **D**.
    - retorna **null** se o elemento não existe
  - **contar(D)**: retorna o número de elementos em **D**
  - **vazia(D)**: indica se a deque **D** está vazia
  - **cheia(D)**: indica se a deque **D** está cheia
    - útil para implementações estáticas

# Implementação do TAD Deques

- Com inserção e remoção de elementos em ambos os extremos...
  - Implementação **sequencial estática circular**



- Implementação **dinâmica (DUPLAMENTE) encadeada**



- Nesses casos, as operações do TAD são  **$O(1)$**

# DEQUES

- Exemplo de aplicação?
  - Implementação de **UNDO** e **REDO**

## Exercícios – não precisa entregar

- Implemente o TAD – fila, versão encadeada dinâmica
- Implemente o TAD – deque, versão encadeada dinâmica
- Implemente um procedimento para inverter uma fila encadeada dinâmica, com encadeamento simples (o primeiro elemento se tornará o último e vice-versa)
  - é possível inverter a fila, com encadeamento simples, sem usar uma estrutura auxiliar (inversão usando a própria fila)?

# Exercício para Entrega em 08/10

- Veja **Exercício 5.pdf** no Tidia