

UFSC-CTC-INE  
INE5384 - Estruturas de Dados

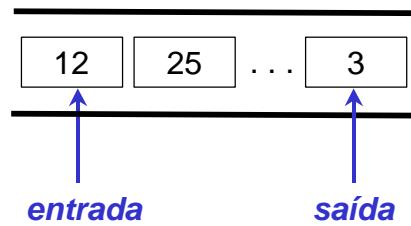
## Deques

Prof. Ronaldo S. Mello  
2002/2

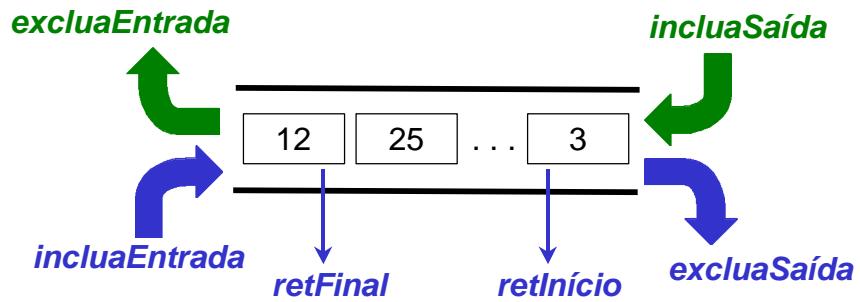
## Deques

- Um deque é uma especialização de uma fila:
  - inserções e exclusões de elementos podem ocorrer em **qualquer extremidade** da lista
- *Deque = DoubleEndedQUEue*
  - fila com duas saídas
- Exemplos:
  - Canal marítimo ou fluvial
  - Servidão com circulação em dois sentidos

## Exemplo de Deque



## Operações sobre um Deque

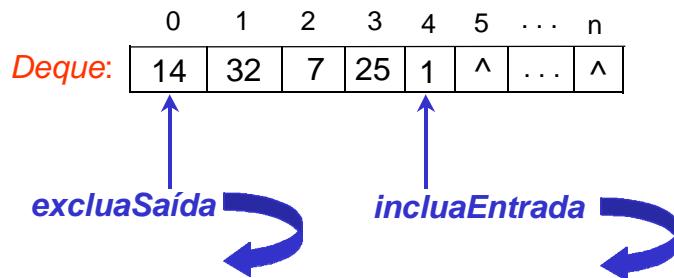


## Alternativas de Implementação

- Deque como vetor
- Deque como lista encadeada

### Deque como Vetor

NroElementos: 5



→ *Estratégia de deslocamento circular à direita continua valendo para operações de fila!*

# Deque como Vetor

*NroElementos:* 5

*Deque:*

0	1	2	3	4	5	...	n
14	32	7	25	1	^	...	^

The diagram illustrates a deque as a horizontal array of elements. The first element is at index 0, and the last element is at index n. Indexes 1 through 4 are also shown. Two blue arrows point upwards from the bottom of the array to specific elements: one arrow points to index 1 (containing 14) with the label "incluirSaída", and another arrow points to index 4 (containing 1) with the label "excluirEntrada".

 **Novidade: deslocamento circular à esquerda para operações adicionais do deque!**

# Simulação

InEnt: A

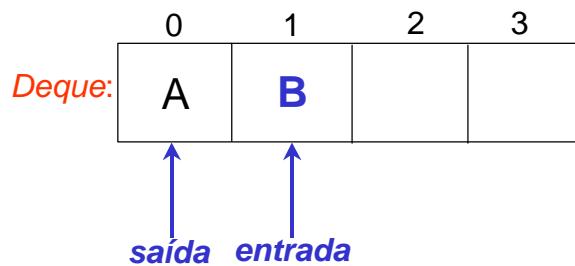
NroElementos: ~~0~~ 1

A diagram illustrating a Deque (double-ended queue) structure. At the top, four indices are labeled: 0, 1, 2, and 3. Below these indices is a horizontal row of four empty boxes. The first box contains the letter 'A' in blue. Two blue arrows point upwards from the bottom of the page towards the second and third boxes respectively. Below the first arrow is the word 'entrada' and below the second arrow is the word 'saída'.

## Simulação

*InEnt: A  
InEnt: B*

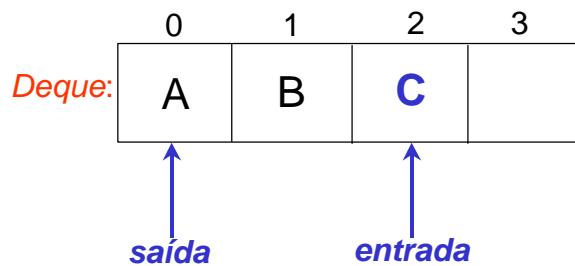
*NroElementos: 0 X 2*



## Simulação

*InEnt: A  
InEnt: B  
InEnt: C*

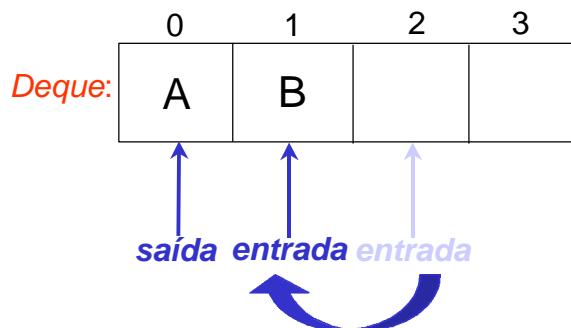
*NroElementos: 0 1 X 3*



## Simulação

*InEnt: A  
InEnt: B  
InEnt: C  
OutEnt*

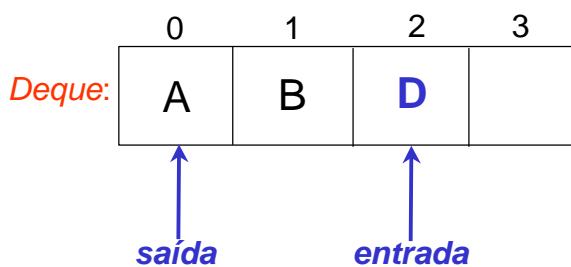
*NroElementos: 0 1 2 ✕ 2*



## Simulação

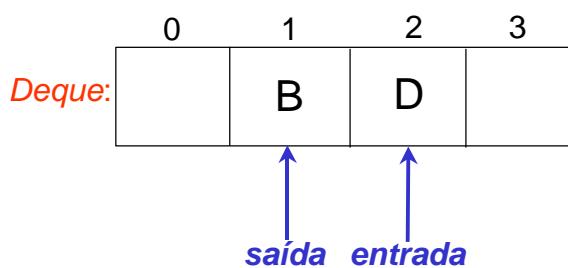
*InEnt: A  
InEnt: B  
InEnt: C  
OutEnt  
InEnt: D*

*NroElementos: 0 1 2 3 ✕ 3*



## Simulação

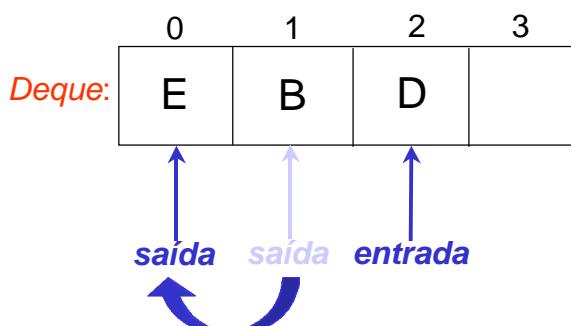
NroElementos: 0 1 2 3 2 ✕ 2



InEnt: A  
InEnt: B  
InEnt: C  
OutEnt  
InEnt: D  
**OutSaí**

## Simulação

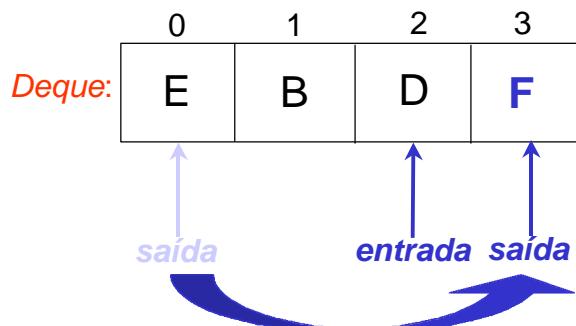
NroElementos: 0 1 2 3 2 3 ✕ 3



InEnt: A  
InEnt: B  
InEnt: C  
OutEnt  
InEnt: D  
OutSaí  
**InSaí: E**

## Simulação

NroElementos: 0 1 2 3 2 3 4 ✕ 4



InEnt: A  
InEnt: B  
InEnt: C  
OutEnt  
InEnt: D  
OutSaí  
InSaí: E  
InSaí: F

## Implementação

```
Classe DequeVetor
Subclasse de FilaVetor
início
...
Construtor DequeVetor (tamanho inteiro);
início
    fila ← NOVO objeto[tamanho];
    entrada ← 0;
    saída ← 0;
    nroElementos ← 0;
fim;
fim
```

## Implementação

```
Classe DequeVetor
Subclasse de FilaVetor
início
...
Método incluaNaEntrada(objeto Object);
início
    inclua(objeto Object);
    fim;
fim
```

## Correção na classe FilaVetor!

```
Classe FilaVetor
início
    fila objeto[ ];
    entrada, saída, nroElementos inteiro;
    ...
Método inclua(objeto Object)
início
    se nroElementos = fila.length então
        Exceção EstruturaCheia();
    se (entrada + 1) = fila.length então entrada ← 0
    senão se nroElementos > 0 então entrada ← entrada + 1;
        fila[entrada] ← objeto;
        nroElementos ← nroElementos + 1;
    fim;
fim
```



## Implementação

Classe DequeVetor

Subclasse de FilaVetor

início

...

```
Método excluaNaEntrada() retorna Object;  
início  
    resposta Object;  
    se nroElementos = 0 então Exceção EstruturaVazia();  
    resposta ← fila[entrada];  
    se entrada = 0 E nroElementos > 1 então entrada ← fila.length - 1  
    senão se nroElementos > 1 então entrada ← entrada - 1;  
    nroElementos ← nroElementos - 1;  
    retorna resposta;  
fim;  
fim
```

## Implementação

Classe DequeVetor

Subclasse de FilaVetor

início

...

```
Método excluaNaSaída() retorna Object;  
início  
    retorna exclua();  
fim;  
fim
```

## Correção na Classe FilaVetor!

```
Classe FilaVetor
início
    fila objeto[ ];
    entrada, saída, nroElementos inteiro;
    ...
    Método exclua() retorna objeto;
    início
        resposta objeto;
        se nroElementos = 0 então Exceção EstruturaVazia();
        resposta ← fila [saída];
        fila[saída] ← NULL;
        se (saída + 1) = fila.length então saída ← 0
        senão se nroElementos > 1 então saída ← saída + 1;
        nroElementos ← nroElementos - 1;
        retorna resposta;
    fim;
fim
```



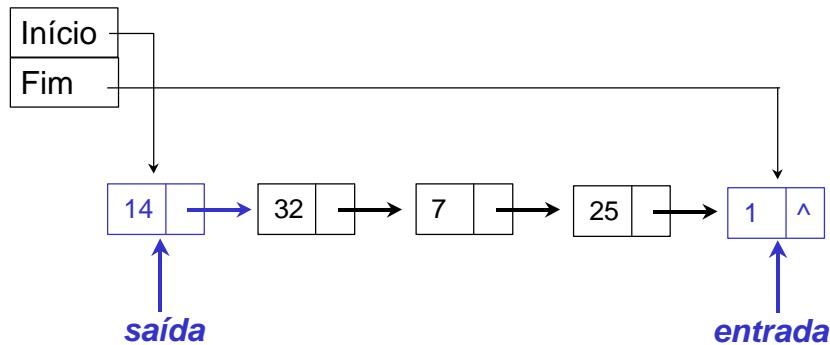
## Implementação

```
Classe DequeVetor
Subclasse de FilaVetor
início
    ...
    Método incluaNaSaída(objeto Object);
    início
        se nroElementos = fila.length então Exceção EstruturaCheia();
        se saída = 0 E nroElementos > 0 então saida ← fila.length - 1
        senão se nroElementos > 0 então saida ← saida - 1;
        fila[saida] ← objeto;
        nroElementos ← nroElementos + 1;
    fim;
fim
```

## Deque como Lista Encadeada

NroElementos: 5

Deque:



## Implementação

```
Classe DequeEncadeada  
Subclasse de FilaEncadeada  
início  
...  
    Construtor DequeEncadeada ();  
    início  
        fila ← NOVO ListaEncadeada();  
        nroElementos ← 0;  
    fim;  
fim
```

## Implementação

```
Classe DequeEncadeada
Subclasse de FilaEncadeada
início
...
Método incluaNaEntrada(objeto Object);
início
    incluir(objeto Object);
    fim;
fim
```

## Implementação

```
Classe DequeEncadeada
Subclasse de FilaEncadeada
início
...
Método excluaNaEntrada retorna Object;
início
    elem Elemento;

    elem ← fila.ObtemUltimoElemento();
    fila.excluiNoFinal();
    nroElementos ← nroElementos – 1;
    retorna elem.ObtemDados();
fim;
fim
```

## Implementação

```
Classe DequeEncadeada
Subclasse de FilaEncadeada
início
...
Método excluaNaSaida() retorna Object;
início
    retorna exclua();
    fim;
fim
```

## Implementação

```
Classe DequeEncadeada
Subclasse de FilaEncadeada
início
...
Método incluaNaSaida(objeto Object);
início
    fila.insereNoInicio(objeto);
    nroElementos ← nroElementos + 1;
    fim;
fim
```

## Exercícios com Deques

1) Suponha a seguinte classe:

```
Classe ExemploDeDeque  
início  
    deque DequeEncadeada; . . .  
fim;
```

Implemente os seguintes métodos:

a) *consulta(pos inteiro) retorna object;*

- retorna o objeto que está na posição *pos* (1 a “n”) do deque

b) *existeObjeto(obj Object) retorna booleano;*

- retorna verdadeiro se existe um objeto com valor igual ao valor de *obj*

## Exercícios com Deques

2) Um deque mantém uma lista de caracteres.

Implemente um método *ehSimétrico()* que retorna verdadeiro caso as duas metades desta lista sejam simétricas.

– Exemplos:

A B A A B A → verdadeiro

A B C X Y A → falso

X Y Z Y X → verdadeiro

## Exercícios com Deques

- 3) Uma lista mantém nomes de 40 alunos (objetos) por ordem de classificação no vestibular em um curso superior. Supondo que esta lista está implementada na forma de um deque, construa um método *exibe(pos inteiro, ordem caractere)* que escreve na tela os nomes de alunos a partir de *pos*, na ordem direta ou inversa ('d' / 'i') indicado por *ordem*

0	1	2	3	39
Rafael Souza	Bruna Silva	Marcos Santos	Clóvis Pereira	... Tânia Oliveira

saída (1º lugar)

entrada (40º lugar)

Exemplos:

*exibe(2, 'd')* → Marcos Santos, Clóvis Pereira, ..., Tânia Oliveira

*exibe(2, 'i')* → Marcos Santos, Bruna Silva, Rafael Souza