UFSC-CTC-INE

INE5384 - Estruturas de Dados

Filas

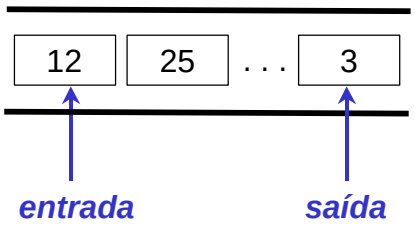
Prof. Ronaldo S. Mello

2002/2

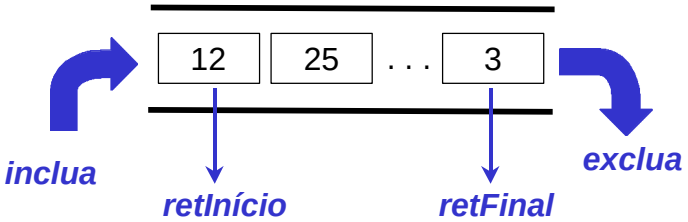
Fila

- Uma fila é uma seqüência de elementos definida por “ordem de chegada”:
  - fila de pessoas na caixa do banco
  - fila de automóveis no pedágio
  - fila de “objetos” quaisquer, ...
- Uma fila é um tipo especial de lista:
  - inserções e exclusões de elementos ocorrem nas extremidades da lista

Exemplo de Fila



Operações sobre uma Fila



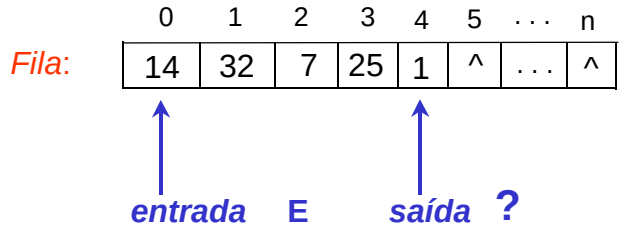
FIFO: “first-in; first-out”

Alternativas de Implementação

- Fila como vetor
- Fila como lista encadeada

Fila como Vetor

NroElementos: 5



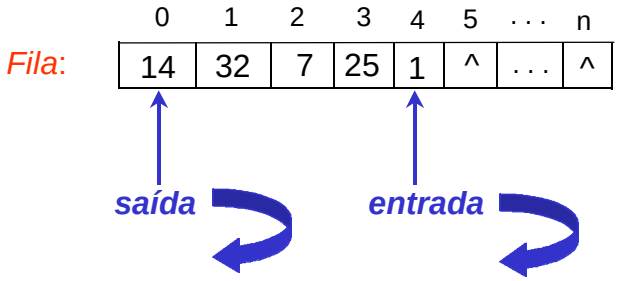
Onde devem ficar a entrada e a saída da fila:
No início e no fim, respectivamente?

Vetor X Encadeamento

Operação	Vetor	Encadeamento
Inserção início	O(n)	O(1)
Inserção final	O(1)	O(1)
Inserção posição "x"	O(n)	O(n)
Pesquisa elemento "x"	O(n)	O(n)
Pesquisa posição "x"	O(1)	O(n)
Exclusão início	O(n)	O(1)
Exclusão final	O(1)	O(n)
Exclusão posição ou elemento "x"	O(n)	O(n)

Fila como Vetor

NroElementos: 5

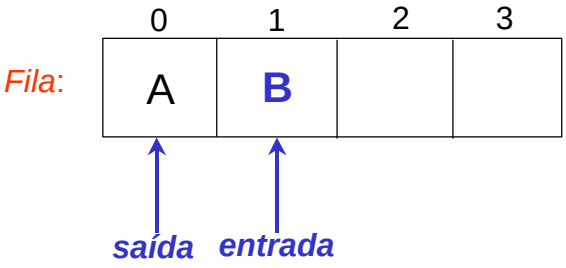


Solução melhor: entrada e saída são móveis e circulares! ➡ *Complexidade O(1)*

Simulação

In: A
In: B

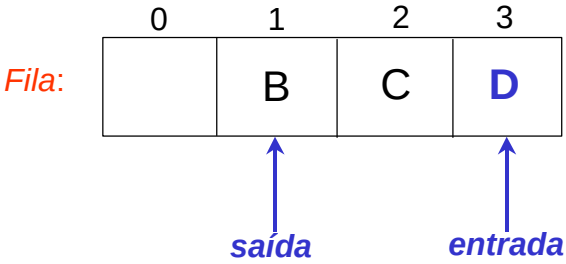
NroElementos: 0 ~~1~~ **2**



Simulação

In: A
In: B
In: C
Out
In: D

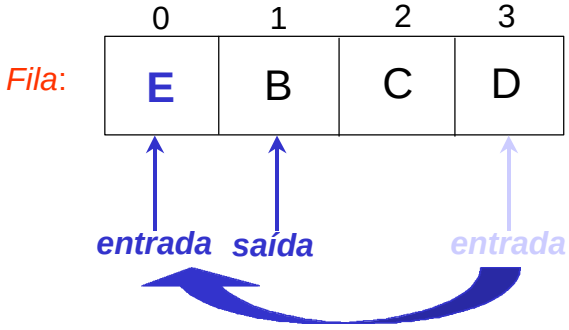
NroElementos: 0 1 2 3 ~~2~~ **3**



Simulação

In: A
In: B
In: C
Out
In: D
In: E

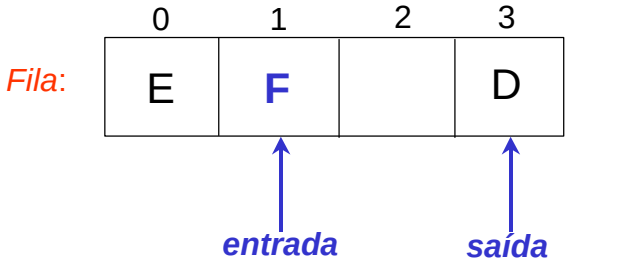
NroElementos: 0 1 2 3 2 ~~3~~ **4**



Simulação

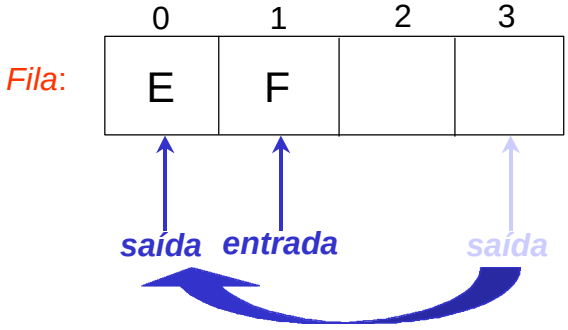
In: A
In: B
In: C
Out
In: D
In: E
Out
Out
In: F

NroElementos: 0 1 2 3 2 3 4 3 ~~2~~ **3**



Simulação

NroElementos: 0 1 2 3 2 3 4 3 2 ~~3~~ **2**



In: A
In: B
In: C
Out
In: D
In: E
Out
Out
In: F
Out

Implementação

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

Implementação

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

Implementação

```
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.lenght então saída ← 0
                                        senão saída ← saída + 1;
          nroElementos ← nroElementos - 1;
          retorna resposta;
    fim;
fim
```

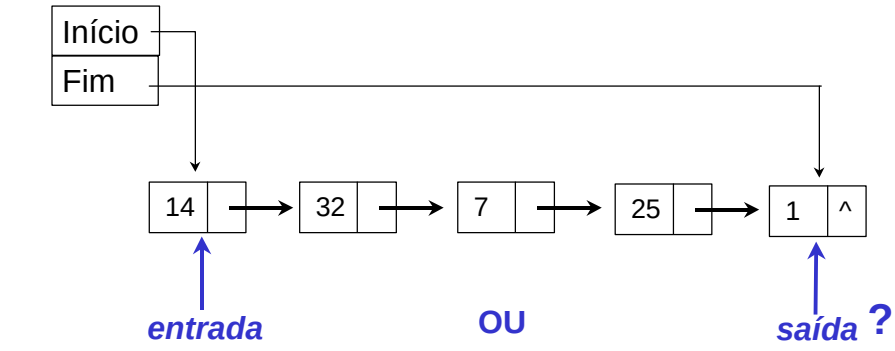
Implementação

```
Classe FilaVetor
início
    fila objeto[ ];
    entrada, saída, nroElementos inteiro;
    ...
    Método retInício() retorna objeto;
    início
          se nroElementos = 0 então Exceção EstruturaVazia();
          retorna fila[saída];
    fim;
    Método retFinal() retorna objeto;
    início
          se nroElementos = 0 então Exceção EstruturaVazia();
          retorna fila[entrada];
    fim;
fim
```

Fila como Lista Encadeada

NroElementos: 5

Fila:



Onde devem ficar a entrada e a saída da fila?

Vetor X Encadeamento

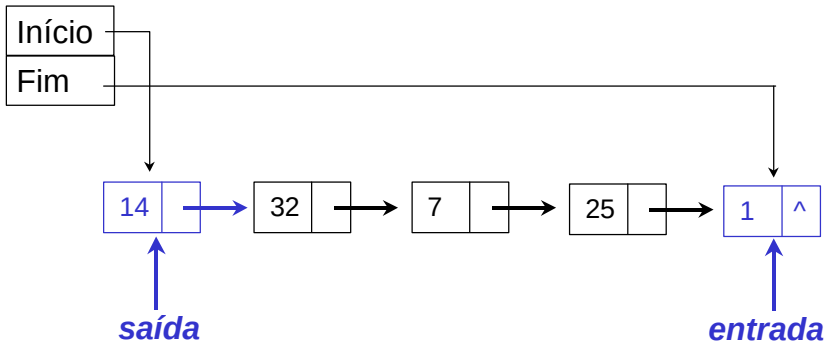
Operação	Vetor	Encadeamento
Inserção início	O(n)	O(1)
Inserção final	O(1)	O(1)
Inserção posição "x"	O(n)	O(n)
Pesquisa elemento "x"	O(n)	O(n)
Pesquisa posição "x"	O(1)	O(n)
Exclusão início	O(n)	O(1)
Exclusão final	O(1)	O(n)
Exclusão posição ou elemento "x"	O(n)	O(n)

Fila como Lista Encadeada

NroElementos: 5

Entrada da fila: No fim!
Saída da fila: No início!

Fila:



Implementação

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

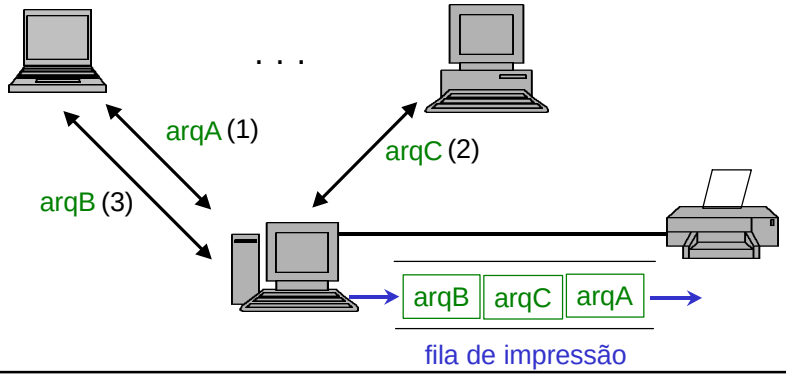
Exercícios

- Implementar na classe *FilaEncadeada*:
  - inclua(objeto object);
  - exclua() retorna object;
  - retInício() retorna object;
  - retFinal() retorna object;

  Métodos úteis da classe *ListaEncadeada*:
  - *insereNoFinal(objeto object);*
  - *excluiNoInício();*
  - *obtémPrimeiroElemento() retorna object;*
  - *obtémUltimoElemento() retorna object;*

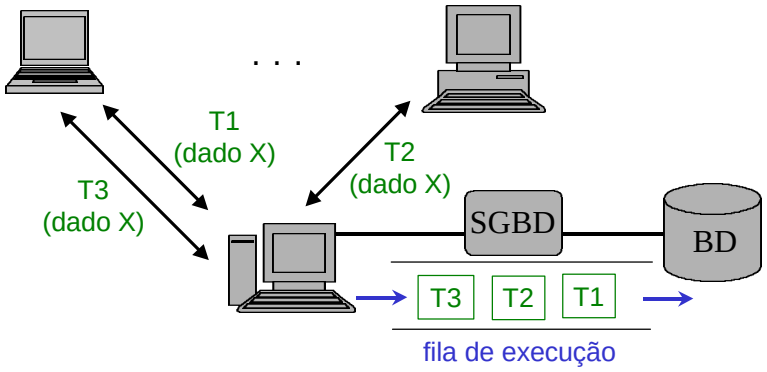
Exemplo de Aplicação de Filas

- Processos requisitando recursos compartilhados
  - **exemplo1: utilização de uma impressora**
  - exemplo2: transações de banco de dados



Exemplo de Aplicação de Filas

- Processos requisitando recursos compartilhados
  - exemplo1: utilização de uma impressora
  - **exemplo2: transações de banco de dados**



Exercício 1

- Dada uma lista de caracteres formada por uma seqüência alternada de letras e dígitos, construa um método que retorne uma lista na qual as letras são mantidas na seqüência original e os dígitos são colocados na ordem inversa.
- Exemplos:

  A 1 E 5 T 7 W 8 G → A 8 E 7 T 5 W 1 G

  3 C 9 H 4 Q 6 → 6 C 4 H 9 Q 3
- Sugestão:
  - usar uma fila e uma pilha
  - supor um método *ehDigito(ch caractere) retorna booleano* que retorna verdadeiro caso um caractere seja um dígito

Exercício 2

Um SGBD executa transações de aplicações por ordem de chegada, a menos que uma aplicação tenha maior prioridade que outra (*o SGBD atende 5 aplicações, tendo cada uma um código (0-4) e um valor de prioridade*). Escreva um método da classe SGBD que recebe um conjunto de "n" transações (*cada transação mantém o código da aplicação, um número seqüencial (número de ordem da transação solicitada pela aplicação) e um programa*) e organiza uma fila de execução destas transações. Transações podem ser inseridas na fila enquanto ela tiver menos que 10 elementos (tamanho máximo da fila). Alcançado este tamanho máximo, o SGBD deve executar a 1ª transação da fila.

Observações:

- as prioridades das aplicações podem ser mantidas em um vetor
- o conjunto de transações de entrada pode ser uma lista
- definir as classes que julgares necessárias
- os métodos definidos para listas e filas podem ser utilizados
- supor que existe um método da classe SGBD chamado *ExecutaProgramaTransação(programa string)*