

# Estruturas de Dados

## Filas

Departamento de Computação  
Prof. Martín Vigil  
Adaptado de prof. Jean Martina e Aldo Wangenheim

2020.1



UNIVERSIDADE FEDERAL  
DE SANTA CATARINA

# Extensões do conceito de Lista Encadeada

- ▶ A ideia da Lista Encadeada vista até agora é o modelo mais geral e simples;
- ▶ Pode ser especializada e estendida das mais variadas formas;
- ▶ Especializada:
  - ▶ Pilhas;
  - ▶ Filas;
- ▶ Estendida:
  - ▶ Listas Duplamente Encadeadas;
  - ▶ Listas Circulares Simples e Duplas.

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).

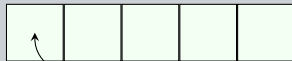
Fila Vazia



# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).

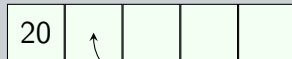


Enfileerar 20

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Enfileirar 55

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Enfileirar 4

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).

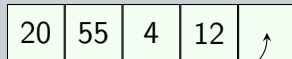


Enfileirar 12

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



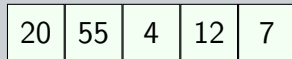
Enfilear 7



# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).

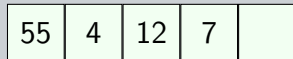


Desenfiletrar 20

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Desenfiletrar 55

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Desenfileirar 4

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Desenfileirar 12

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o **primeiro a entrar é o primeiro a sair** (FIFO = *first in, first out*).



Desenfileirar 7

# Conceito de Filas

## Fila

É uma estrutura de dados cujo funcionamento é inspirado no conceito “natural” de enfileiramento, onde o primeiro a entrar é o primeiro a sair (FIFO = *first in, first out*).

Fila Vazia



# Fila

## Modelagem funcional

- ▶ Inserir dado ao fim da fila (***enqueue***);
- ▶ Remover dado do início da fila (***dequeue***);
- ▶ Verificar tamanho da fila;

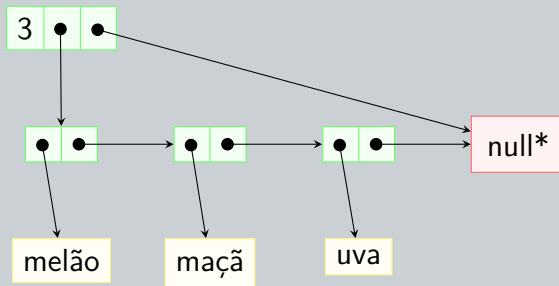
# Fila usando Lista Encadeada

## Modelagem funcional

- ▶ Inserir dado ao fim da fila (**enqueue**) é  $\Theta(n)$  😞
- ▶ Remover dado do início da fila (**dequeue**) é  $\Theta(1)$  😊
- ▶ Verificar tamanho da fila é  $\Theta(1)$  😊



# Fila usando Lista Encadeada Modificada



# Fila usando Lista Encadeada

## Modelagem funcional

- ▶ Inserir dado ao fim da fila (**enqueue**) é  $\Theta(1)$  😊
- ▶ Remover dado do início da fila (**dequeue**) é  $\Theta(1)$  😊
- ▶ Verificar tamanho da fila é  $\Theta(1)$  😊

# Modelagem da Cabeça de Fila Encadeada

- ▶ Aspecto Estrutural:
  - ▶ Necessitamos um ponteiro para o primeiro elemento da fila;
  - ▶ Necessitamos um ponteiro para o último elemento da fila;
  - ▶ Necessitamos um inteiro para indicar quantos elementos a fila possui.

```
estrutura Fila {  
    Elemento *_primeiro;  
    Elemento *_ultimo;  
    int _quantidade;  
};
```

# Método *inicializaFila()*

- ▶ Inicializamos `_primeiro` como nulo;
- ▶ Inicializamos `_ultimo` como nulo;
- ▶ Inicializamos o `_quantidade` como “0”;

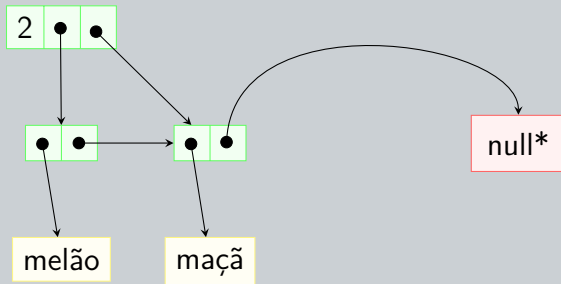
```
inicializaFila()  
inicio  
  Fila* novo <- ALOQUE(Fila)  
  novo._primeiro <- null;  
  novo._ultimo <- null;  
  novo._quantidade <- 0;  
  RETORNE novo;  
fim;
```

# Método enqueue( $T^*$ dado)

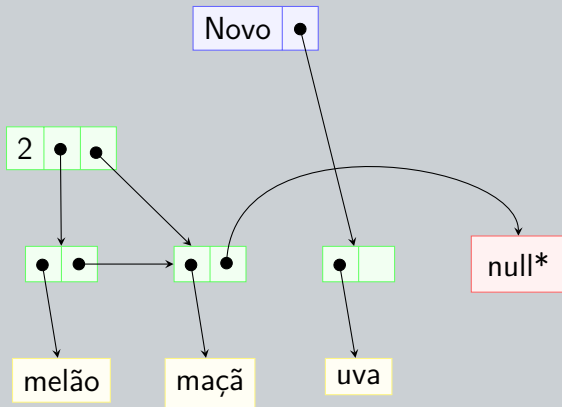
Ilustração simplificada em

<https://www.cs.usfca.edu/~galles/visualization/QueueLL.html>

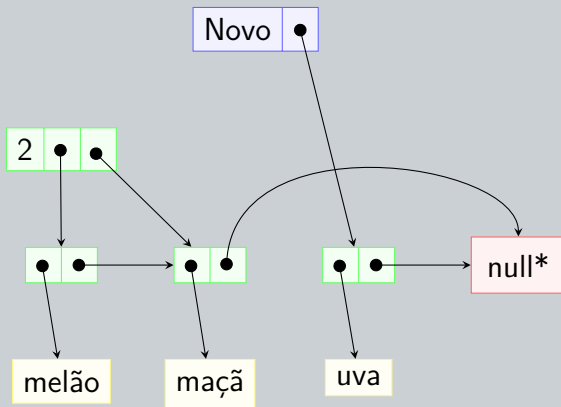
# Método *enqueue*



## Método *enqueue*

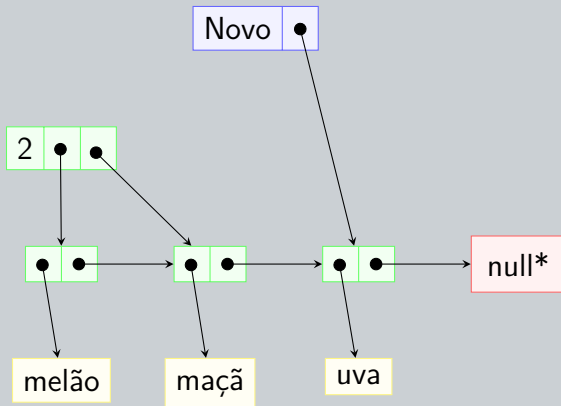


# Método *enqueue*

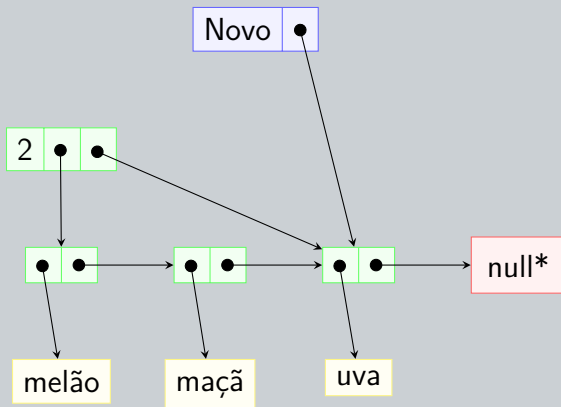




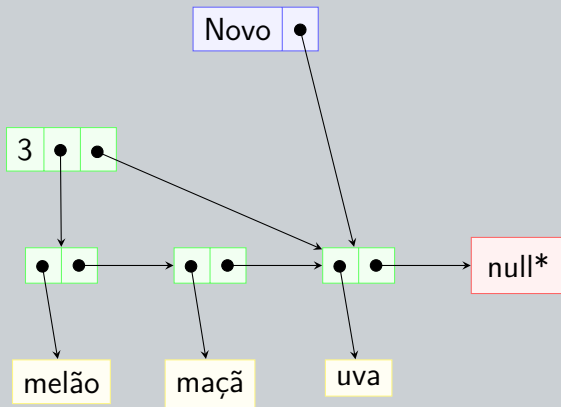
# Método *enqueue*



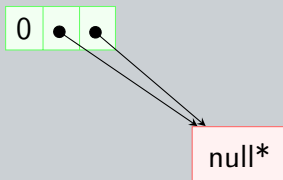
# Método *enqueue*



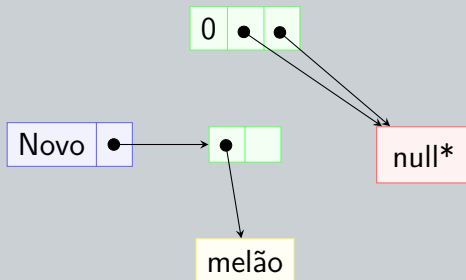
# Método *enqueue*



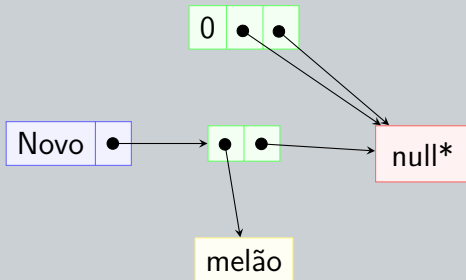
# Método *enqueue* - Caso Especial Fila Vazia



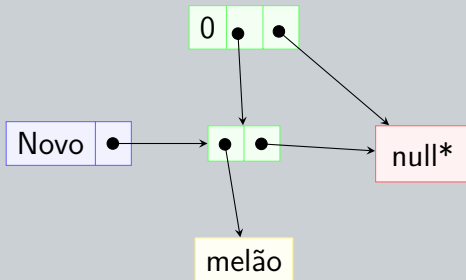
# Método *enqueue* - Caso Especial Fila Vazia



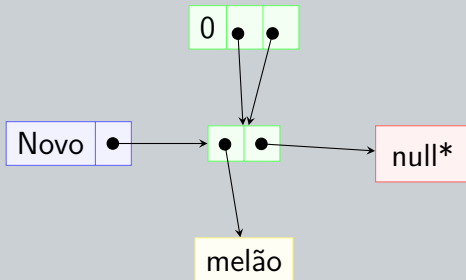
# Método *enqueue* - Caso Especial Fila Vazia



# Método *enqueue* - Caso Especial Fila Vazia

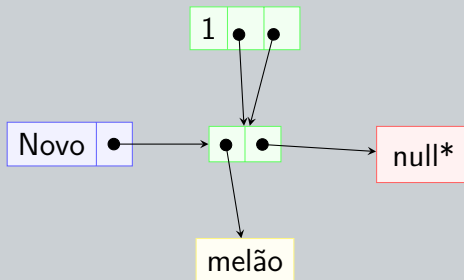


# Método *enqueue* - Caso Especial Fila Vazia





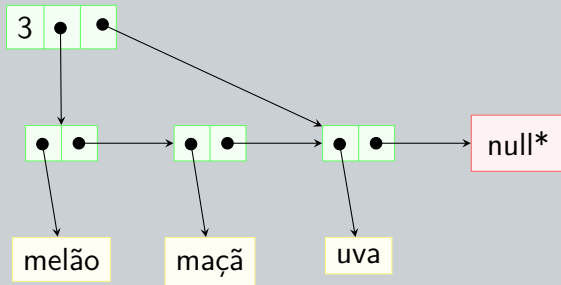
# Método *enqueue* - Caso Especial Fila Vazia



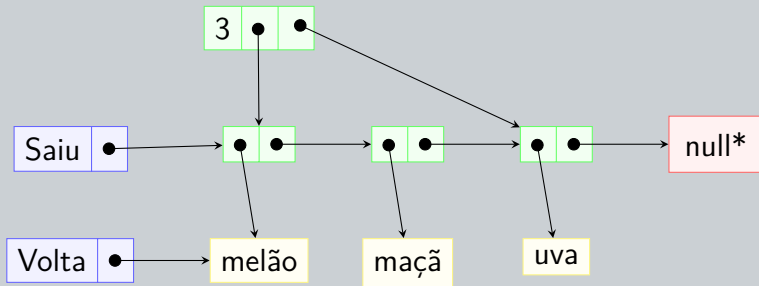
# Método *enqueue*

```
enqueue(Fila* fila, T* dado)
  Elemento *novo; // auxiliar.
  inicio
    novo <- ALOQUE(Elemento);
    SE ( novo == null) THROW(ERROFILACHEIA);
    SE filaVazia(fila) ENTAO
      fila._primeiro <- novo
    SENA
      fila._ultimo._proximo <- novo;
    FIM SE
    novo._proximo <- null;
    novo._dado <- dado;
    lista._ultimo <- novo;
    lista._quantidade <- lista._quantidade + 1;
  FIM SE
fim;
```

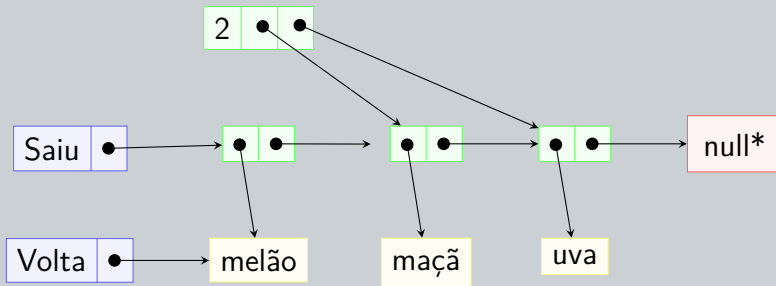
# Método *dequeue()*



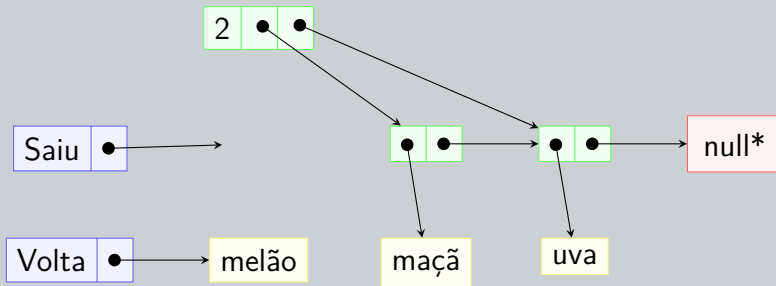
# Método *dequeue()*



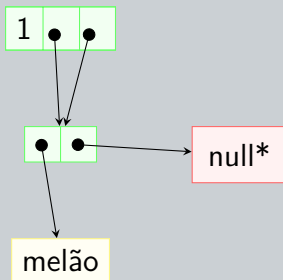
# Método *dequeue()*



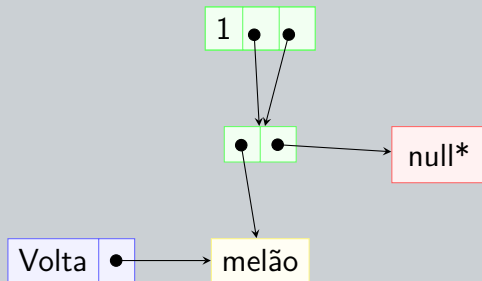
# Método *dequeue()*



# Método *dequeue()* - Caso Especial Fila Unitária

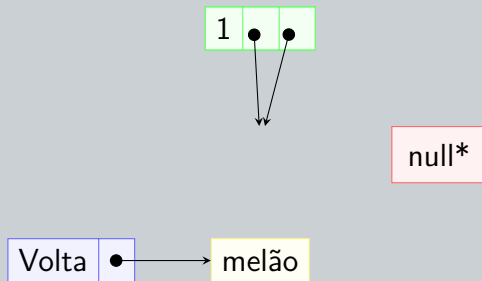


# Método *dequeue()* - Caso Especial Fila Unitária

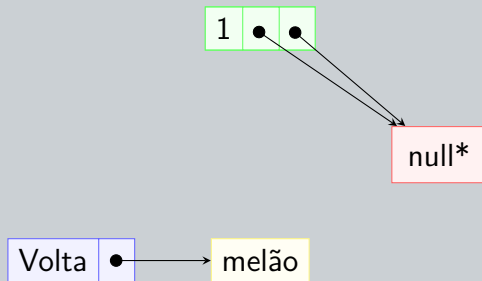




# Método *dequeue()* - Caso Especial Fila Unitária



# Método *dequeue()* - Caso Especial Fila Unitária



# Método *dequeue()*

```
T* dequeue(Fila* fila)
Elemento *saiu; //Variável auxiliar elemento.
T *volta; //Variável auxiliar tipo T.
início
  SE (listaVazia(fila)) ENTAO
    THROW(ERROLISTAVAZIA);
  SENA0
    saiu <- fila._primeiro;
    volta <- saiu._dado;
    fila._primeiro <- saiu._proximo;
    //Se SAIU for o único, próximo é NULO e está certo.
    SE (fila._quantidade = 1) ENTAO
      //Fila unitária: devo anular o _ultimo também.
      fila._ultimo <- null;
    FIM SE
    fila._quantidade <- fila._quantidade - 1;
    LIBERE(saiu);
    RETORNE(volta);
  FIM SE
fim;
```

# Aplicações de Filas

- ▶ É importante para gerência de dados/processos por ordem cronológica:
  - ▶ Fila de impressão em uma impressora de rede;
  - ▶ Fila de pedidos de uma expedição ou tele-entrega;
- ▶ É importante para simulação de processos sequenciais:
  - ▶ chão de fábrica: fila de camisetas a serem estampadas;
  - ▶ comércio: simulação de fluxo de um caixa de supermercado;
  - ▶ tráfego: simulação de um cruzamento com um semáforo;
- ▶ Útil para algoritmos de busca em largura.

# Perguntas????



UNIVERSIDADE FEDERAL  
DE SANTA CATARINA



Este trabalho está licenciado sob uma Licença Creative Commons Atribuição 4.0 Internacional. Para ver uma cópia desta licença, visite

<http://creativecommons.org/licenses/by/4.0/>.



UNIVERSIDADE FEDERAL  
DE SANTA CATARINA