INSTITUTO FEDERAL
Sudeste de Minas Gerais
Campus Manhuaçu

Estruturas de Dados I
Filas

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*

Instituto Federal do Sudeste de Minas Gerais

29 de janeiro de 2025

Filas

Descrição

- Filas são um tipo abstrato de dados com a característica de que **o primeiro elemento a ser inserido é o primeiro a ser removido** (política FIFO – First in First Out).

Filas

Descrição

- Filas são um tipo abstrato de dados com a característica de que **o primeiro elemento a ser inserido é o primeiro a ser removido** (política FIFO – First in First Out).
- Considera-se uma analogia com filas de elementos, como pessoas, processos etc.

Filas

Descrição

- Filas são um tipo abstrato de dados com a característica de que **o primeiro elemento a ser inserido é o primeiro a ser removido** (política FIFO – First in First Out).
- Considera-se uma analogia com filas de elementos, como pessoas, processos etc.
- Os usos de filas incluem filas de impressão e filas de processamento em sistemas operacionais, entre outros.

Operações

O tipo abstrato de dados (TAD) fila deve, obrigatoriamente, suportar os métodos:

- enfileirar(o): Insere o objeto no fim da fila.

Operações

O tipo abstrato de dados (TAD) fila deve, obrigatoriamente, suportar os métodos:

- enfileirar(o): Insere o objeto no fim da fila.
- desenfileirar(): Retira o objeto no inicio da fila e o retorna; se a fila estiver vazia, ocorre um erro.

Operações

O tipo abstrato de dados (TAD) fila deve, obrigatoriamente, suportar os métodos:

- enfileirar(o): Insere o objeto no fim da fila.
- desenfileirar(): Retira o objeto no inicio da fila e o retorna; se a fila estiver vazia, ocorre um erro.

Adicionalmente, podemos definir os seguintes métodos auxiliares:

- tamanho(): Retorna o número de objetos na fila.

Operações

O tipo abstrato de dados (TAD) fila deve, obrigatoriamente, suportar os métodos:

- enfileirar(o): Insere o objeto no fim da fila.
- desenfileirar(): Retira o objeto no inicio da fila e o retorna; se a fila estiver vazia, ocorre um erro.

Adicionalmente, podemos definir os seguintes métodos auxiliares:

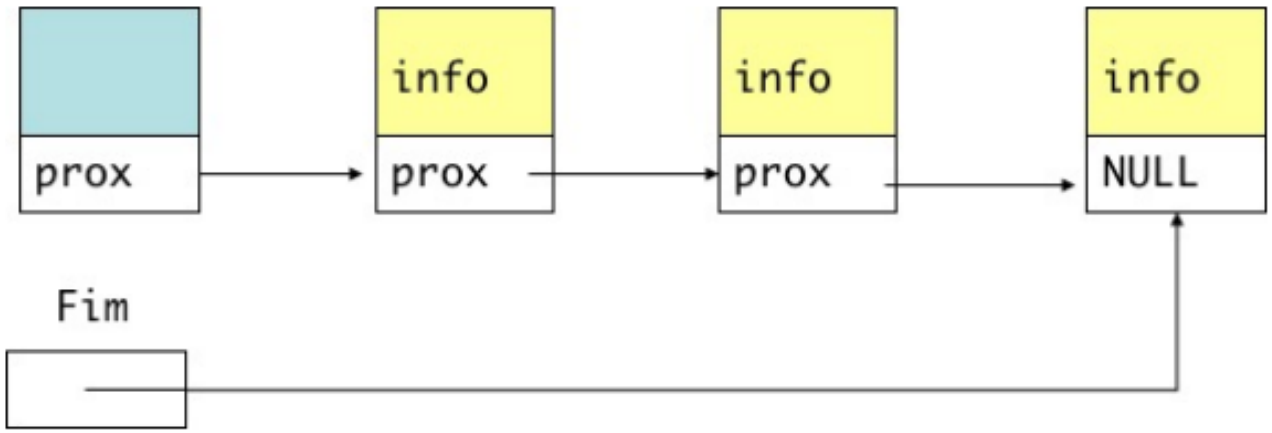
- tamanho(): Retorna o número de objetos na fila.
- vazia(): Retorna um *boolean* indicando se a fila está vazia.

Implementação por referência

- A fila é implementada por meio de células, tal que cada célula contém um item da fila e um apontador para a próxima célula.

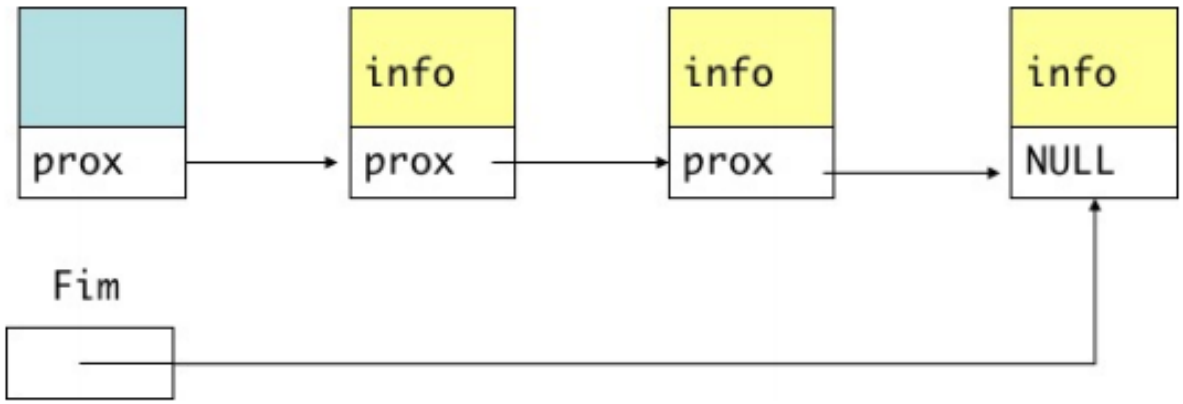
Implementação por referência

- A fila é implementada por meio de células, tal que cada célula contém um item da fila e um apontador para a próxima célula.
- A estrutura contém um apontador para a frente da fila (célula cabeça) e um apontador para a parte de trás da fila (fim).

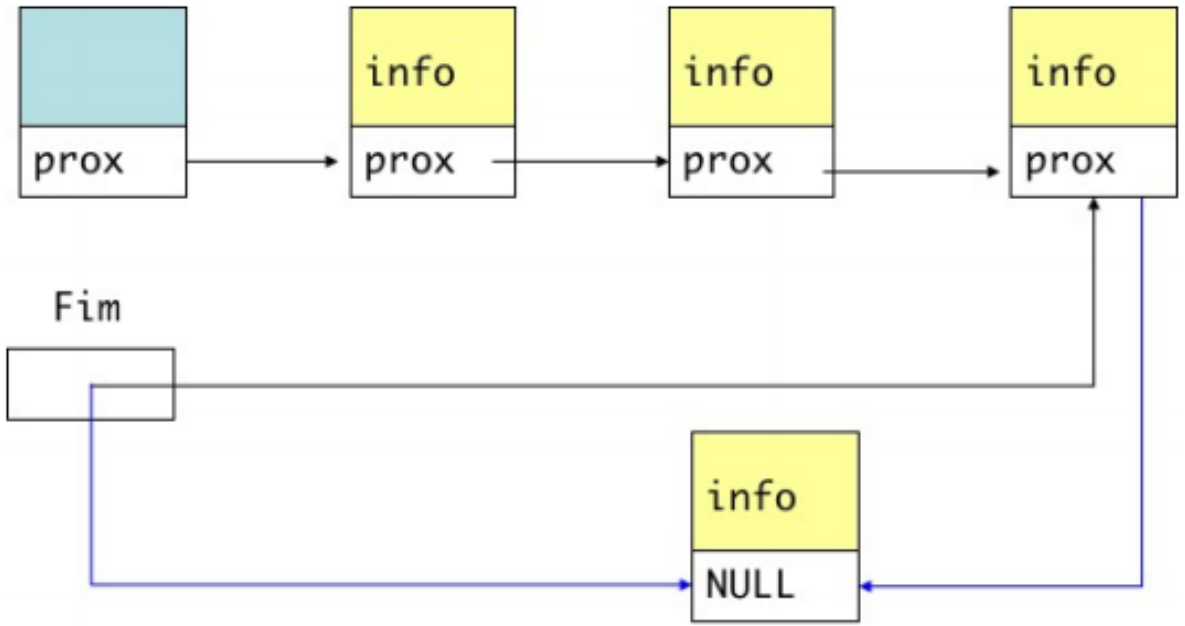


Implementação por referência - Inserção

De acordo com a política **FIFO**, há apenas uma opção de posição onde podemos inserir elementos: o fim da fila (ou seja, a última posição).

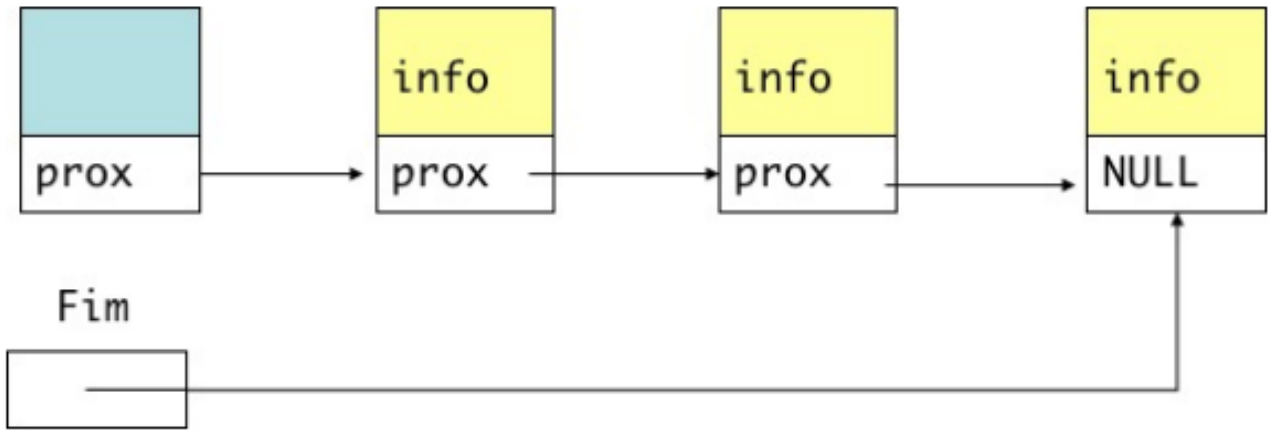


Implementação por referência - Inserção

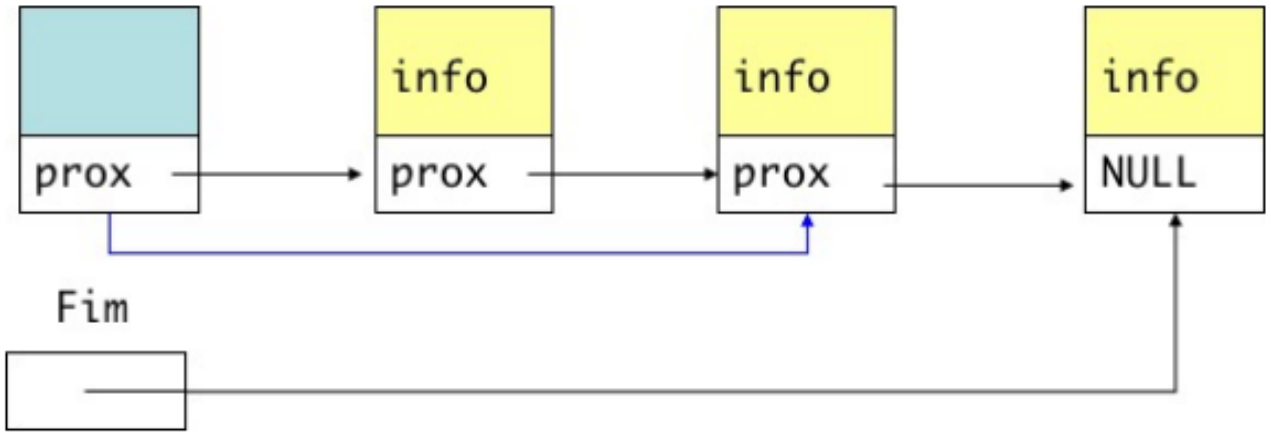


Implementação por referência - Remoção

De acordo com a política **FIFO**, há apenas uma opção de posição onde podemos remover elementos: o início da fila (ou seja, primeira posição).



Implementação por referência - Remoção



Complexidade

A complexidade de todas as operações é mantida da implementação de Lista:

- Enfileirar: $\theta(1)$
- Desenfileirar: $\theta(1)$

Perguntas?

Baixe o exemplo aqui

Exercícios

Implemente o exemplo sem o uso de *generics*. Você terá que fazer uma fila para pacientes e outra para professores.

Exercícios

Considere uma pilha P vazia e uma fila F não vazia. Utilizando apenas os testes de fila e pilha vazias, as operações Enfileira, Desenfileira , Empilha (push), Desempilha (pop), e uma variável aux do tipo <elemento>, escreva uma função que inverta a ordem dos elementos da fila.

Exercícios

(GIT) Escreva um programa em Java que permita cadastrar Strings em uma fila *q* (utilize Referência). Os cadastros (enfileiramento) acontecerão de dois em dois. O usuário, para preencher esta fila inicial, irá informar um nome e uma operação (A, B ou X) a cada operação de enfileiramento. Quando o usuário terminar o cadastro, os elementos da fila original serão separados em duas filas, *a* e *b*, de acordo com a operação.

- Nome A - Adiciona nome à fila a
- Nome B - Adiciona nome à fila b
- Nome X - Adiciona nome à fila que tenha menos elementos (a ou b). Se ambas as filas tiverem o mesmo número de elementos, o nome é descartado e não é adicionada a nenhuma.

Exercícios

Por exemplo, se a fila *q* for preenchida com [Luis, B, Pedro, A, Luisa, A, Joao, X, Jose, X, Miguel, B] quando for processada acontecerá:

1. Luis, B - Luis é adicionado à fila b;
2. Pedro, A - Pedro é adicionado à fila a;
3. Luisa, A - Luisa é adicionada à fila a;
4. Joao, X - Joao é adicionado à fila b, que tem apenas 1 elemento (Luis) contra os dois da fila b (Pedro e Luisa);
5. Jose, X - José é descartado (ambas as filas têm 2 elementos);
6. Miguel, B - Miguel é adicionado à fila b;

No final, a fila *a* ficaria com *[Pedro, Luisa]* e a fila *b* ficaria com *[Luis, Joao, Miguel].* A fila q deve ficar vazia.

GitHub (Ao vivo)

O aeroporto de Congonhas recebe todos os dias uma média de 600 pousos e decolagens, ou cerca de 36 por hora. No último ano, foram exatamente 223.989 movimentos aéreos. Para organizar todo o fluxo de aviões que chegam a Congonhas e saem de lá, a torre de controle funciona o tempo inteiro com nível máximo de atenção. Para descartar qualquer possibilidade de erro humano o chefe do controle de tráfego aéreo de Congonhas contratou você para desenvolver um programa que organize automaticamente o fluxo de aviões no campo de pouso. Para isso, basta seguir o seguinte protocolo, os aviões que veem do lado Oeste da pista têm maior prioridade de serem colocados na fila, pois são aqueles que estão mais próximo do localizador (início da pista). Já os aviões que estão se aproximando pelo lado Norte e Sul, devem ser inseridos na fila 1 por vez, ou seja, insere-se 1 avião do lado Norte e em seguida 1 avião do lado Sul. Por último, insere-se o próximo avião que esteja se aproximando ao lado leste da pista.

Entrada

A entrada é composta por um número inteiro **P**, representando o ponto cardeal do avião que entrou no campo da pista (-4 <= **P** <= -1), onde (-4 representa o lado leste, -3 o lado norte, -2 lado sul e -1 lado oeste) . Em seguida é realizada a entrada dos respectivos aviões, compostos de um identificador começando com a letra "A" seguida de um número inteiro **I** (1 <= **I** <= 1000). A qualquer momento é permitido trocar o ponto cardeal, e inserir novas aeronaves, repetidamente até que o controlador finalize a sessão com o dígito 0.

Saída

A saída é composta de uma linha contendo as aeronaves enfileiradas pela ordem do protocolo estabelecido pelo aeroporto.

GitHub (Ao vivo)

Exemplos de Entrada	Exemplos de Saída
-4 A1 A26 A38 A23 -1 A80 A40 -2 A2 A16 A108 -3 A20 A44 0	A80 A20 A2 A1 A40 A44 A16 A26 A108 A38 A23

GitHub (Ao vivo)

```
-4
A12
A33
-3
A8
A33
-2
A77
A102
A866
-3
A21
A15
A9
-1
A2
0
```

```
A2 A8 A77 A12 A33 A102 A33 A21 A866 A15 A9
```


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
dziękuję
bedankt
hvala
mauruuru
go raibh maith agat
sukriya
kop khun krap
terima kasih
grazie
arigatô
dakujem
tapadh leat
mochchakkeram
спасибо
рахмат
Баярлалаа
faafetai lava
kiitos
dankie
dhanyavad
nandri
nanni
bayarlalaa
gracie
enkosi
koszonom
vinaka
спасибі
blagodaram
mersi
barka
wetalin
tack
spas
dank je
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
tenki
djiere dieuf
tau
akun
dankon
aciu
mamnun
дякую
murakoze
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpay
taiku
sobodi
dèkuji
sagolun
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
ありがとう
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
takk
trugarez
мерси
xiexie
감사합니다
ευχαριστώ

Referências

- CARVALHO, Marco Antonio Moreira de. **Projeto e análise de algoritmos**. 01 mar. 2018, 15 jun. 2018. Notas de Aula. PPGCC. UFOP
- GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de Dados & Algoritmos em Java.** Bookman Editora, 2013.
- ZIVIANI, Nivio. **Projeto de Algoritmos com implementações em Java e C++**, 2007.