

# Estrutura de Dados - I

## Fila

Prof. MSc. Rafael Staiger Bressan  
[rafael.bressan@unicesumar.edu.br](mailto:rafael.bressan@unicesumar.edu.br)

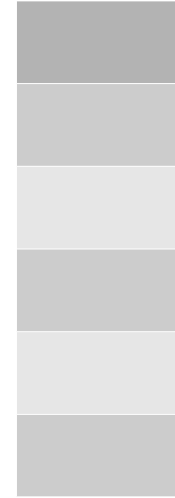


# Fila

- Estruturas do tipo **FIFO** (*First In First Out*)
  - primeiro elemento a ser inserido, será o primeiro a ser retirado.
- Assim como as pilhas, uma fila também pode ser implementada por meio de um vetor ou de uma lista encadeada

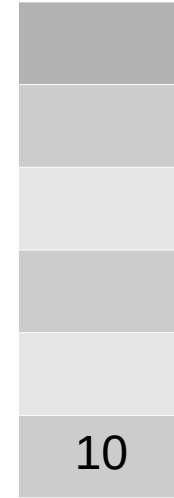
# Fila

- Fila Vazia



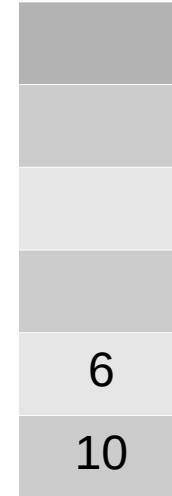
# Fila

- Fila Vazia
- insert(10)



# Fila

- Fila Vazia
- insert(10)
- insert(6)



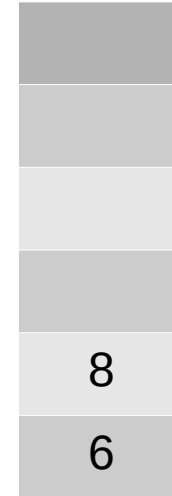
# Fila

- Fila Vazia
- insert(10)
- insert(6)
- insert(8)



# Fila

- Fila Vazia
- insert(10)
- insert(6)
- insert(8)
- remove()



# Fila

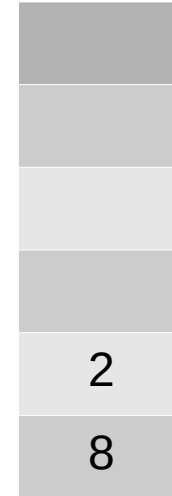
- Fila Vazia
- insert(10)
- insert(6)
- insert(8)
- remove()
- insert(2)





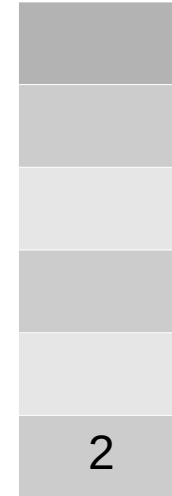
# Fila

- Fila Vazia
- insert(10)
- insert(6)
- insert(8)
- remove()
- insert(2)
- remove()



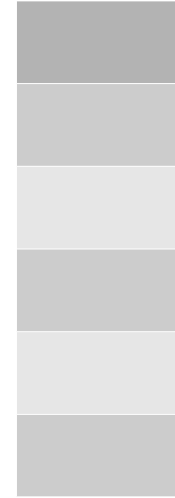
# Fila

- Fila Vazia
- insert(10)
- insert(6)
- insert(8)
- remove()
- insert(2)
- remove()
- remove()



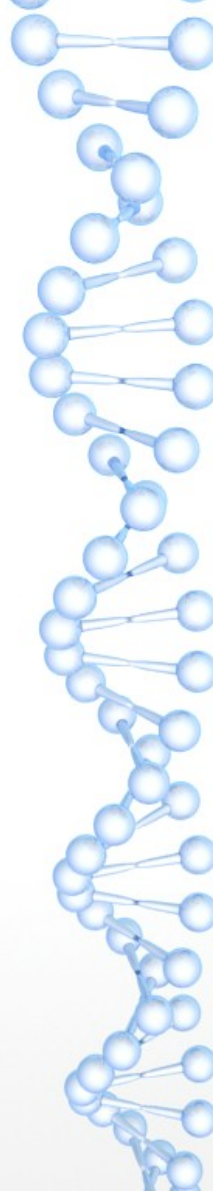
# Fila

- Fila Vazia
- insert(10)
- insert(6)
- insert(8)
- remove()
- insert(2)
- remove()
- remove()
- remove()
- Fila Vazia

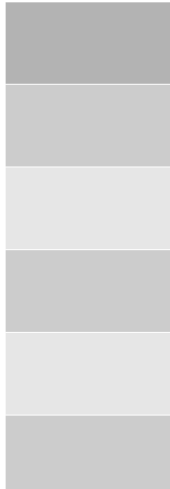


# Fila

## Inserção



```
10 void insereFila(int conteudo, celula *ini) {  
11     if (ini->prox == NULL){  
12         celula *nova; nova = malloc(sizeof(celula));  
13         nova->conteudo = conteudo;  
14         nova->prox = NULL;  
15         ini->prox = nova;  
16     }else{  
17         insereFila(conteudo, ini->prox);  
18     }  
19 }
```



# Fila

## Inserção



insereFila(10, ini);

```
10 void insereFila(int conteudo, celula *ini) {  
11     if (ini->prox == NULL){  
12         celula *nova; nova = malloc(sizeof(celula));  
13         nova->conteudo = conteudo;  
14         nova->prox = NULL;  
15         ini->prox = nova;  
16     }else{  
17         insereFila(conteudo, ini->prox);  
18     }  
19 }
```

10

# Fila

## Inserção



insereFila(5, ini);

```
10 void insereFila(int conteudo, celula *ini) {  
11     if (ini->prox == NULL){  
12         celula *nova; nova = malloc(sizeof(celula));  
13         nova->conteudo = conteudo;  
14         nova->prox = NULL;  
15         ini->prox = nova;  
16     }else{  
17         insereFila(conteudo, ini->prox);  
18     }  
19 }
```

5

10

# Fila

## Inserção



insereFila(8, ini);

```
10 void insereFila(int conteudo, celula *ini) {  
11     if (ini->prox == NULL){  
12         celula *nova; nova = malloc(sizeof(celula));  
13         nova->conteudo = conteudo;  
14         nova->prox = NULL;  
15         ini->prox = nova;  
16     }else{  
17         insereFila(conteudo, ini->prox);  
18     }  
19 }
```

8

5

10

# Fila

## Inserção



insereFila(16, ini);

```
10 void insereFila(int conteudo, celula *ini) {  
11     if (ini->prox == NULL){  
12         celula *nova; nova = malloc(sizeof(celula));  
13         nova->conteudo = conteudo;  
14         nova->prox = NULL;  
15         ini->prox = nova;  
16     }else{  
17         insereFila(conteudo, ini->prox);  
18     }  
19 }
```

16
8
5
10



# Fila

## Remoção

```
22 void removeFila(celula *ini) {  
23     if (ini->prox == NULL){  
24         printf("Lista Vazia!\n");  
25     }else{  
26         celula *lixo;  
27         lixo = ini->prox;  
28         ini->prox = ini->prox->prox;  
29         free(lixo);  
30     }  
31 }
```

16

8

5

10

# Fila

## Remoção



removeFila(ini);

```
22  [ ] void removeFila(celula *ini) {  
23  [ ]     if (ini->prox == NULL){  
24  [ ]         printf("Lista Vazia!\n");  
25  [ ]     }else{  
26  [ ]         celula *lixo;  
27  [ ]         lixo = ini->prox;  
28  [ ]         ini->prox = ini->prox->prox;  
29  [ ]         free(lixo);  
30  [ ]     }  
31  [ ] }
```

16

8

5

# Fila

## Remoção



removeFila(ini);

```
22  [ ] void removeFila(celula *ini) {  
23  [ ]     if (ini->prox == NULL){  
24  [ ]         printf("Lista Vazia!\n");  
25  [ ]     }else{  
26  [ ]         celula *lixo;  
27  [ ]         lixo = ini->prox;  
28  [ ]         ini->prox = ini->prox->prox;  
29  [ ]         free(lixo);  
30  [ ]     }  
31  [ ] }
```

16

8

# Fila

## Remoção



removeFila(ini);

```
22  [ ] void removeFila(celula *ini) {  
23  [ ]     if (ini->prox == NULL){  
24  [ ]         printf("Lista Vazia!\n");  
25  [ ]     }else{  
26  [ ]         celula *lixo;  
27  [ ]         lixo = ini->prox;  
28  [ ]         ini->prox = ini->prox->prox;  
29  [ ]         free(lixo);  
30  [ ]     }  
31  [ ] }
```

16

# Fila

## Remoção



removeFila(ini);

```
22  [ ] void removeFila(celula *ini) {  
23  [ ]     if (ini->prox == NULL){  
24  [ ]         printf("Lista Vazia!\n");  
25  [ ]     }else{  
26  [ ]         celula *lixo;  
27  [ ]         lixo = ini->prox;  
28  [ ]         ini->prox = ini->prox->prox;  
29  [ ]         free(lixo);  
30  [ ]     }  
31  [ ] }
```



# Atividade

- Desenvolva um programa que contenha as funções `insereFila()` e `removeFila()` apresentadas no slide para manipulação de uma fila.
- Seu programa deve conter as opções:
  - `insereFila`
  - `removeFila`
  - Fila vazia
  - Exibir dados