

# Estrutura de Dados

## Aula 8 : Filas (Queue)

**Prof. MSc. Fausto Sampaio**

[https://github.com/Fausto14/estrutura\\_de\\_dados](https://github.com/Fausto14/estrutura_de_dados)

Centro Universitário UniFanor - Wyden

19 de novembro de 2019

## 1 Filas

- Definição
- Aplicações
- Operações

## 2 Fila Estática

- Definição
- Implementação

## 3 Fila Dinâmica

- Definição
- Implementação

## 4 Exemplos

## 5 Referências

# Filas

# Definição

- Uma estrutura do tipo **Fila** é uma sequência de elementos do mesmo tipo, como as listas;
- Seus elementos possuem estrutura interna abstraída, ou seja, sua complexidade é arbitrária e não afeta o seu funcionamento.



# Definição

- Uma **Fila** é um tipo especial de lista on inserções e exclusões de elementos ocorrem nas extremidades da lista;
- FIFO (First In First Out): Primeiro que entra, primeiro que sai;
- LIFO (Last In Last Out): Último que entra, último que sai;



- Controle de fluxo;
- Recursos compartilhados: impressoras, transações de banco de dados;
- Atendimento de processos requisitados ao um sistema operacional;
- Buffer para gravação de dados em mídia;
- Processos de comunicação em redes de computadores.
- etc;

- Em uma Fila podemos realizar as seguintes operações básicas:
  - criação da fila;
  - inserção de um elemento no final;
  - inserção de um elemento no início;
  - acesso ao elemento do início;
  - destruição da fila
- Essas operações dependem do tipo de alocação de memória usada:
  - estática;
  - dinâmica;

- O espaço de memória é alocado no momento da compilação;
- Exige a definição do número máximo de elementos da Fila;
- Acesso sequencial: elementos consecutivos na memória;



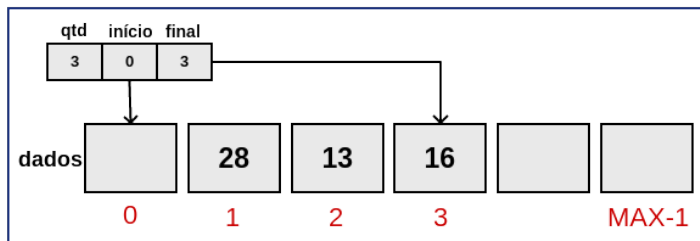
- O espaço de memória é alocado em tempo de execução;
- A Fila cresce à medida que novos elementos são armazenados, e diminui à medida que elementos são removidos;
- Acesso encadeado: cada elemento pode estar em uma área distinta da memória;
- Para acessar um elemento, é preciso percorrer todos os antecessores na Fila.
- O acesso é através de ponteiros! Não tem índices!

# Fila Estática

# Definição

- **Fila Estática** : Tipo de Fila onde o sucessor de um elemento ocupa a posição física seguinte do mesmo;
- Uso de vetores (array).

## Fila - Estática



# Implementação em C

## FilaEstatica.h

- os protótipos das funções;
- o tipo de dado armazenado na fila;
- o ponteiro "fila".
- tamanho do vetor usado na fila.

## FilaEstatica.c

- o tipo de dados "fila";
- implementar as suas funções.

## Observação

- A implementada também contempla a circularidade da Fila estática;
- Sempre que os índices de início ou de fim chegam no final do vetor, estes são redefinidos na primeira posição do vetor;
- **Cuidado:** No percurso sobre a estrutura, o fim do vetor precisa ser determinado logicamente (quantidade de elementos na fila), e não mais fisicamente pelo fim da estrutura.

## FilaEstatica.h

```
2  #define MAX 100
3  struct aluno{
4      int matricula;
5      char nome[30];
6      float n1,n2,n3;
7  };
8
9  typedef struct fila Fila;
10
11  Fila* cria_Fila();
12  void libera_Fila(Fila* fi);
13  int consulta_Fila(Fila* fi, struct aluno *al);
14  int insere_Fila(Fila* fi, struct aluno al);
15  int remove_Fila(Fila* fi);
16  int tamanho_Fila(Fila* fi);
17  int Fila_vazia(Fila* fi);
18  int Fila_cheia(Fila* fi);
19  void imprime_Fila(Fila* fi);
20
```




# Definição - Fila Estática

## FilaEstatica.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "FilaEstatica.h" //inclui os Protótipos
4
5  //Definição do tipo Fila
6  struct fila{
7      int inicio, final, qtd;
8      struct aluno dados[MAX];
9  };
```

# Criar e Liberar - Fila Estática

## FilaEstatica.c

```
11  Fila* cria_Fila(){
12     Fila *fi;
13     fi = (Fila*) malloc(sizeof(struct fila));
14      if(fi != NULL){
15         fi->inicio = 0;
16         fi->final = 0;
17         fi->qtd = 0;
18     }
19     return fi;
20 }
21
22  void libera_Fila(Fila* fi){
23     free(fi);
24 }
```

# Informações da Fila Estática

```
52 □ int tamanho_Fila(Fila* fi){  
53     if(fi == NULL)  
54         return -1;  
55     return fi->qtd;  
56 }
```

(a) Tamanho da Fila

```
58 □ int Fila_cheia(Fila* fi){  
59     if(fi == NULL)  
60         return -1;  
61     if (fi->qtd == MAX)  
62         return 1;  
63     else  
64         return 0;  
65 }
```

(b) Fila Cheia

```
67 □ int Fila_vazia(Fila* fi){  
68     if(fi == NULL)  
69         return -1;  
70     if (fi->qtd == 0)  
71         return 1;  
72     else  
73         return 0;  
74 }
```

(c) Fila Vazia

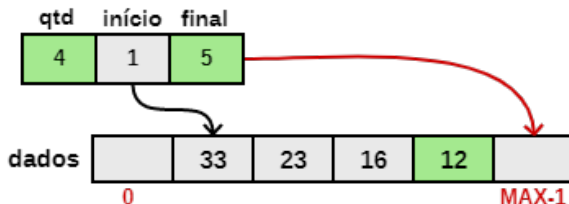
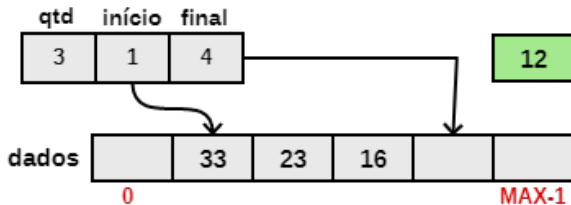


# Inserir Elemento

- Enfileirar - enqueue;
- Em uma fila a inserção é sempre no seu final;
- Cuidado: não se pode inserir numa fila cheia;

```
33 int insere_Fila(Fila* fi, struct aluno al){
34     if(fi == NULL)
35         return 0;
36     if(Fila_cheia(fi))
37         return 0;
38     fi->dados[fi->final] = al;
39     fi->final = (fi->final+1)%MAX;
40     fi->qtd++;
41     return 1;
42 }
```

# Inserir Elemento

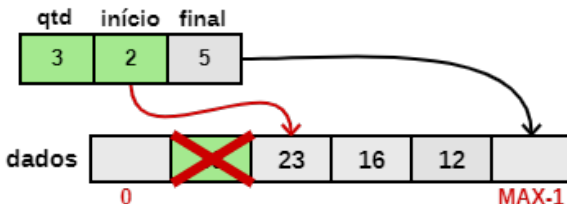
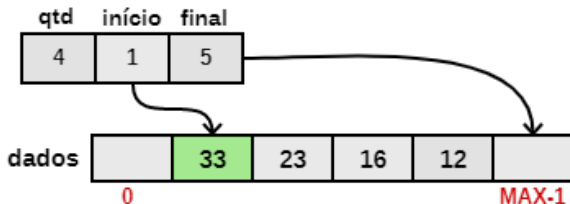


# Remover Elemento

- Desenfileirar - dequeue;
- Em uma fila a remoção é sempre no seu início;
- Cuidado: não se pode remover de uma fila vazia;

```
44 int remove_Fila(Fila* fi){  
45     if(fi == NULL || Fila_vazia(fi))  
46         return 0;  
47     fi->inicio = (fi->inicio+1)%MAX;  
48     fi->qtd--;  
49     return 1;  
50 }
```

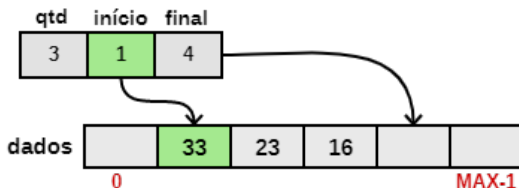
# Remover Elemento



# Consultar Elemento

- Em uma fila a consulta se dá apenas ao elemento que está no seu início;

```
26 int consulta_Fila(Fila* fi, struct aluno *al){  
27     if(fi == NULL || Fila_vazia(fi))  
28         return 0;  
29     *al = fi->dados[fi->inicio];  
30     return 1;  
31 }
```

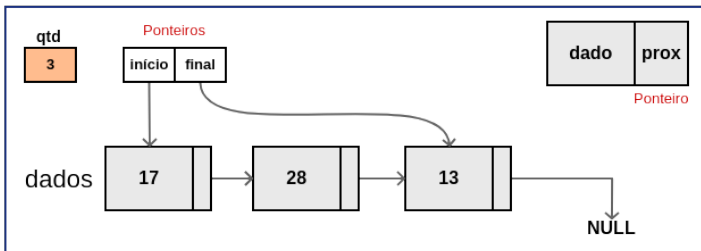


# Fila Dinâmica

# Definição

- **Fila Dinâmica** : Tipo de Fila onde cada elemento aponta para o seu sucessor na Fila;
- Usa um **nó descritor** para representar o início e final da Fila, assim como a sua quantidade de elementos.

## Fila - Dinâmica



## FilaDin.h

- os protótipos das funções;
- o tipo de dado armazenado na fila;
- o ponteiro "fila".

## FilaDin.c

- o tipo de dados "fila";
- implementar as suas funções.



## FilaDin.h

```
2  #define MAX 100
3  struct aluno{
4      int matricula;
5      char nome[30];
6      float n1,n2,n3;
7  };
8
9  typedef struct fila Fila;
10
11  Fila* cria_Fila();
12  void libera_Fila(Fila* fi);
13  int consulta_Fila(Fila* fi, struct aluno *al);
14  int insere_Fila(Fila* fi, struct aluno al);
15  int remove_Fila(Fila* fi);
16  int tamanho_Fila(Fila* fi);
17  int Fila_vazia(Fila* fi);
18  int Fila_cheia(Fila* fi);
19  void imprime_Fila(Fila* fi);
20
```

# Definição - Fila Dinâmica

## FilaDin.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "FilaDin.h" //inclui os Protótipos
4  //Definição do tipo Fila
5  struct elemento{
6      struct aluno dados;
7      struct elemento *prox;
8  };
9  typedef struct elemento Elem;
10 //Definição do Nó Descritor da Fila
11 struct fila{
12     struct elemento *inicio;
13     struct elemento *final;
14     int qtd;
15 };
```

# Criar e Liberar - Fila Dinâmica

## FilaDin.c

```
17 Fila* cria_Fila(){
18     Fila* fi = (Fila*) malloc(sizeof(Fila));
19     if(fi != NULL){
20         fi->final = NULL;
21         fi->inicio = NULL;
22         fi->qtd = 0;
23     }
24     return fi;
25 }
```

(a) Criar Fila Dinâmica

```
27 void libera_Fila(Fila* fi){
28     if(fi != NULL){
29         Elem* no;
30         while(fi->inicio != NULL){
31             no = fi->inicio;
32             fi->inicio = fi->inicio->prox;
33             free(no);
34         }
35         free(fi);
36     }
37 }
```

(b) Liberar Fila Dinâmica

# Informações da Fila Dinâmica

```
79 int tamanho_Fila(Fila* fi){  
80     if(fi == NULL)  
81         return 0;  
82     return fi->qtd;  
83 }
```

(a) Tamanho da Fila

```
93 int Fila_cheia(Fila* fi){  
94     return 0;  
95 }
```

(b) Fila Cheia

```
85 int Fila_vazia(Fila* fi){  
86     if(fi == NULL)  
87         return 1;  
88     if(fi->inicio == NULL)  
89         return 1;  
90     return 0;  
91 }
```

(c) Fila Vazia

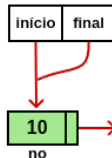
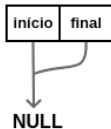
# Inserir Elemento - Fila Dinâmica

- Enfileirar - enqueue;
- Em uma fila a inserção é sempre no seu final;
- Cuidado: não se pode inserir numa fila cheia;

```
48 int insere_Fila(Fila* fi, struct aluno al){
49     if(fi == NULL)
50         return 0;
51     Elem *no = (Elem*) malloc(sizeof(Elem));
52     if(no == NULL)
53         return 0;
54     no->dados = al;
55     no->prox = NULL;
56     if(fi->final == NULL) //fila vazia
57         fi->inicio = no;
58     else
59         fi->final->prox = no;
60     fi->final = no;
61     fi->qtd++;
62     return 1;
63 }
```

# Inserir Elemento - Fila Dinâmica

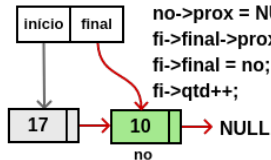
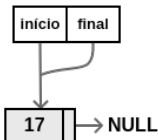
Inserção em Fila vazia



```
no->prox = NULL;  
fi->início = no;  
fi->final = no;  
fi->qtd++;
```



Inserção em Fila NÃO vazia



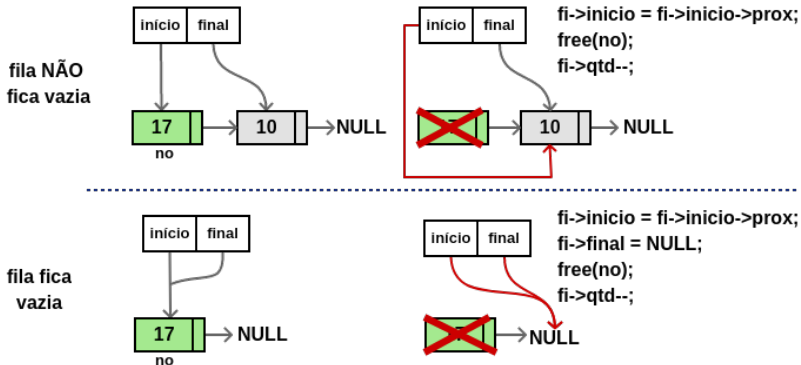
```
no->prox = NULL;  
fi->final->prox = no;  
fi->final = no;  
fi->qtd++;
```

# Remover Elemento - Fila Dinâmica

- Desenfileirar - dequeue;
- Em uma fila a remoção é sempre no seu início;
- Cuidado: não se pode remover de uma fila vazia;

```
65 int remove_Fila(Fila* fi){
66     if(fi == NULL)
67         return 0;
68     if(fi->inicio == NULL)//fila vazia
69         return 0;
70     Elem *no = fi->inicio;
71     fi->inicio = fi->inicio->prox;
72     if(fi->inicio == NULL)//fila ficou vazia
73         fi->final = NULL;
74     free(no);
75     fi->qtd--;
76     return 1;
77 }
```

# Remover Elemento - Fila Dinâmica

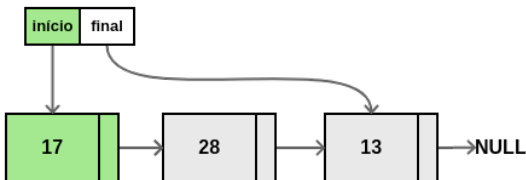




# Consultar Elemento - Fila Dinâmica

- Em uma fila a consulta se dá apenas ao elemento que está no seu início;

```
39 int consulta_Fila(Fila* fi, struct aluno *al){
40     if(fi == NULL)
41         return 0;
42     if(fi->inicio == NULL) //fila vazia
43         return 0;
44     *al = fi->inicio->dados;
45     return 1;
46 }
```



## Exemplos

- `https://portaldoprofessor.fct.unesp.br/projetos/cadilag/apps/structs/?list=card`

## Referências

- André Ricardo Backes, CAPÍTULO 6 - Filas, Editor(s): André Ricardo Backes, **Estrutura de Dados Descomplicada em Linguagem C**, Elsevier Editora Ltda., 2016, Pages 193-220, ISBN 9788535285239.

