Estrutura de Dados 1

FILA ESTÁTICA

FILA DINÂMICA

Profª Juliana Franciscani

Roteiro

01 O que é

02 Fila Estática
Conceito

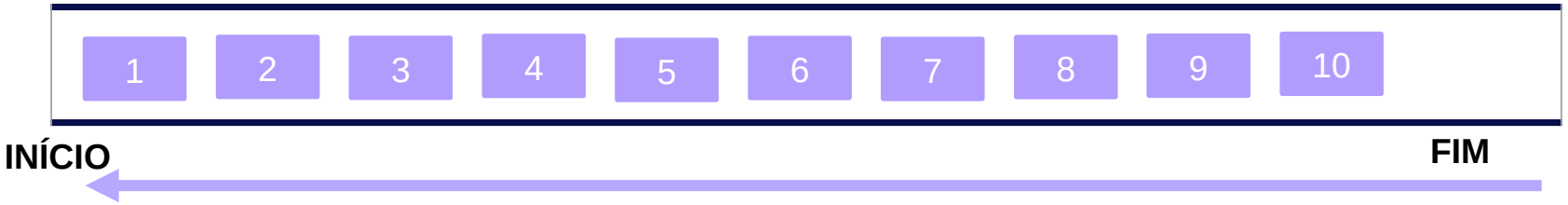
03 Fila Estática
Código

04 Fila dinâmica
Conceito

05 Fila dinâmica
Código

FILA

- Sequência de elementos de um mesmo tipo
- Tipo especial de lista
- É uma estrutura em que os elementos são inseridos sempre no final da fila , e a remoção sempre no início da fila.



FILA

FIFO (First In First Out) Primeiro elemento a entrar é o primeiro elemento a sair.
Inserção sempre no FIM da Fila
Remoção sempre no INÍCIO da Fila
Só é visível o início da FILA, apenas o primeiro elemento estará visível para ser manipulado.

APLICAÇÃO DE FILA

- Controle de fluxo em geral
- Administração de recursos compartilhados. (como buffer de impressora)

FILA Estática

- Espaço de memória é definido no momento de **compilação**
- Deve-se definir o tamanho máximo MAX do vetor a ser utilizado
- **Acesso é sequencial:** elementos consecutivos na memória
- Deve-se verificar se a FILA está cheia antes de cada inserção.

FILA Dinâmica

- Espaço de memória é definido no momento de **execução**
- A FILA cresce a cada elemento inserido e diminui a cada elemento removido.
- **Acesso é Encadeado:** cada elemento pode estar em uma área distinta da memória.
- Para acessar um elemento é preciso percorrer todos os seus antecessores

FILA

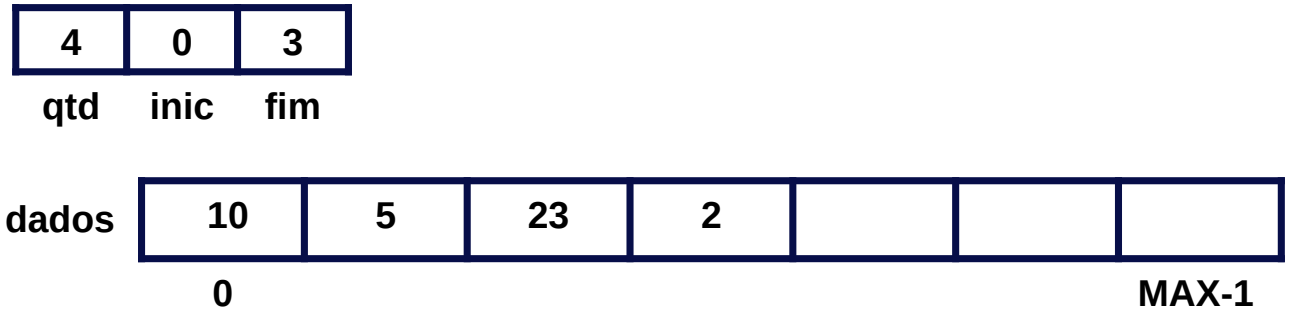
- Criar fila;
- Inserir item **no fim** da fila;
- Acessar um elemento **no início** da fila;
- Remover o elemento **do início** da fila;
- Contar número de itens;
- Verificar se a fila está vazia;
- Verificar se a fila está cheia (estática);
- Exibir os elementos da fila.

FILA ESTÁTICA

Explicação
Código em CPP

FILA ESTÁTICA

- É composta por informações como: quantidade de elementos, início e o final da Fila, além do vetor que armazenará os dados.
- Necessário informar o tamanho máximo (MAX) desse vetor.



qtd	inic	fim
4	0	3

dados

0					MAX-1
10	5	23	2		

Inserir 7, 3, 9
Remover
Remover
Inserir 5

Inserir 5
Se qtd!=max ok
Atualiza fim
Inserir no fim+1 (%MAX) ok
se fim==MAX-1 então atualiza p fim para 0
Inserir no fim
Atualizar o qtd.

Inserir 3
Se qtd!=max ok
Inserir no fim+1
Atualizar o qtd e o fim.

Inserir 7
Se qtd!=max ok
Inserir no fim+1
Atualizar o qtd e o fim.

Inserir 9
Se qtd!=max → não
Fila está cheia

Remover
Qtd != 0 ok
Remover elemento que está no início
Atualizar qtd e início

Remover
Qtd != 0 ok
Remover elemento que está no início
Atualizar qtd e inicio

qtd	inic	fim
5	2	0

dados

0	1	2	3	4	5
5		23	2		

```
#include <iostream>
#include "Fila Estatica.h"
#include<windows.h>

using namespace std;
char menuInicial();
char menuSaida();

int main(){
    SetConsoleCP(1252);
    SetConsoleOutputCP(1252);
    char menu;    int x;
    Fila *fila;
    Aluno alunoN;
    fila = criarFila();
    do{
        menu = menuInicial();
        switch(menu){
        case '1':
            if(fila->qtd==MAX)
                cout<<"Fila cheia, impossível inserir!\n";
            else{
                cadastrarAluno(&alunoN);
                x=inserirFila(fila,&alunoN);
            }
            break;
        case '2':
            x=removerFila(fila);
            break;
        case '3':
            exibirFila(fila);
            break;
        default:
            cout << "Opção Inválida!";
        }
        menu = menuSaida();
        system("cls");
    }while(menu!='S');
    apagarFila(fila);
    return 0;
}
```

```c
#ifndef FILA_ESTATICA_H_INCLUDED
#define FILA_ESTATICA_H_INCLUDED
#define MAX 3

struct ALUNO{
    char nome[100];
    int matricula;
    float nota;
};
typedef struct ALUNO Aluno;

struct FILA{
    int qtd, inicio, fim;
    ALUNO aluno[MAX];
};
typedef struct FILA Fila;

Fila* criarFila();
void apagarFila(Fila *fila);
int filaEhCheia(Fila *fila);
int filaEhVazia(Fila *fila);
int inserirFila(Fila *fila, Aluno *alunoN);
int removerFila(Fila *fila);
void exibirFila(Fila *fila);
void cadastrarAluno(Aluno *alunoN);
#endif // FILA_ESTATICA_H_INCLUDED
```

```cpp
#include<iostream>
#include "Fila Estatica.h"
using namespace std;
//criar lista
Fila* criarFila(){
    Fila *fila = new Fila;
    if(!fila){
        cout<<"Erro de alocação!!!";
        exit(1);
    }
    fila->qtd =0;
    fila->inicio =0;
    fila->fim =0;
    return fila;
}
// apagar lista
void apagarFila(Fila *fila){
    delete fila;
}
// verificar cheia
int filaEhCheia(Fila *fila){
    if(fila->qtd==MAX){
        cout << "Fila está cheia!\n";
        return 1;
    }
    else return 0; // 0 é para condição falsa
}
// verificar vazia
int filaEhVazia(Fila *fila){
    // return (fila->qtd==0);
    if(fila->qtd==0){
        cout << "Fila está vazia!\n";
        return 1;
    }
    else return 0;
}
//cadastro aluno
void cadastrarAluno(Aluno *alunoN){
    cout << "Cadastro Aluno:\n";
    cout <<"Nome: ";
    fflush(stdin);
    cin.getline(alunoN->nome, sizeof(alunoN->nome));
    cout << "Matrícula: ";
    cin >> alunoN->matricula;
    cout << "Nota: ";
    cin >> alunoN->nota;
}
```

```
48  // inserir elemento sempre no final
49  int inserirFila(Fila *fila, Aluno *alunoN){
50      if(fila == nullptr) return 0;
51      if(filaEhCheia(fila)==1) // if(filaEhCheia(fila))
52          return 0;
53      if(fila->fim==MAX){
54          fila->fim = 0;
55      }
56      fila->aluno[fila->fim]=*alunoN;
57      fila->fim++;
58      fila->qtd++;
59      cout << "Cadastro realizado com sucesso!\n";
60      return 1;
61  }
62  // remover elemento sempre no inicio
63  int removerFila(Fila *fila){
64      if(fila == nullptr) return 0;
65      if(fila->qtd==0) // if(filaEhVazia(fila)) return 0;
66          return 0;
67      fila->inicio++;
68      if(fila->inicio==MAX) fila->inicio=0;
69      fila->qtd--;
70      cout << "Remoção realizada com sucesso!\n";
71      return 1;
72  }
```

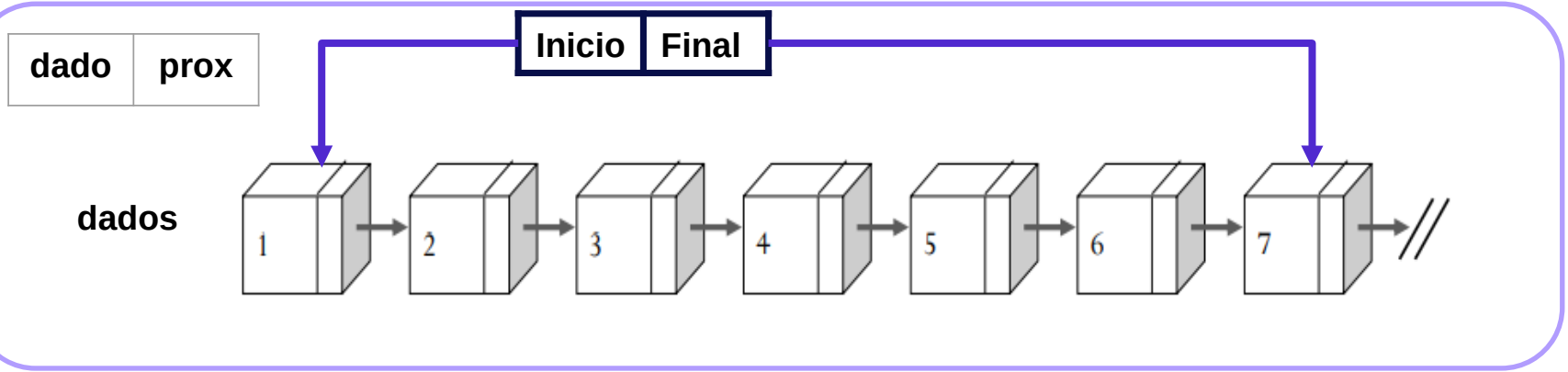
```
73  void exibirFila(Fila *fila){
74      int i,j;
75      if(fila->qtd ==0) cout << "Não há cadastro de aluno!\n";
76      else{
77          for(i=1,j=fila->inicio;i<=fila->qtd;i++){
78              cout<< i <<"º Cadastro:\n"
79                  << "Nome: " << fila->aluno[j].nome
80                  << "\nMatrícula: " << fila->aluno[j].matricula
81                  << "\nNota: " << fila->aluno[j].nota << "\n\n";
82              j++;
83              if(j==MAX) j=0;
84          }
85          cout<<"\n\n";
86      }
87  }
```

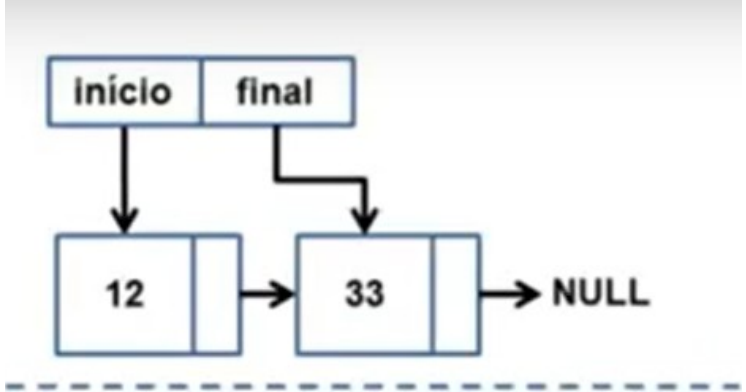
FILA DINÂMICA

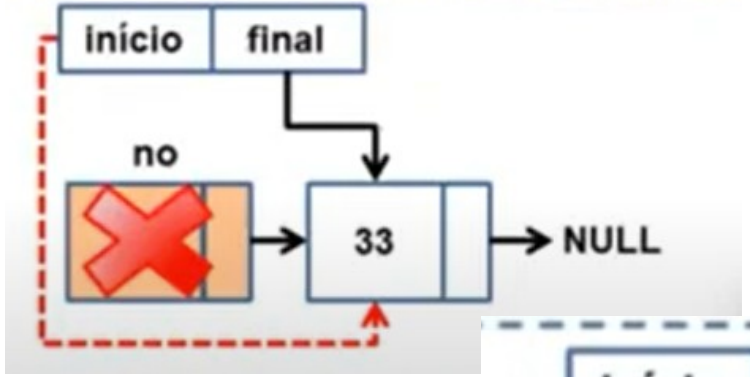
Explicação
Código em CPP

FILA DINÂMICA

- Alocação dinâmica para cada elemento um NEW
- Composta pelo dado, um ponteiro para o próximo elemento e indicadores (ponteiros) para o início e para o fim da fila

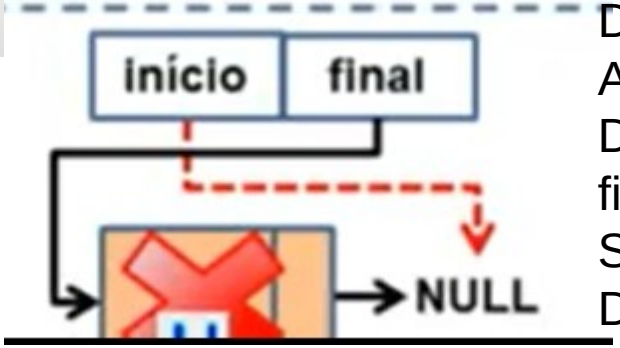






Remover elemento...

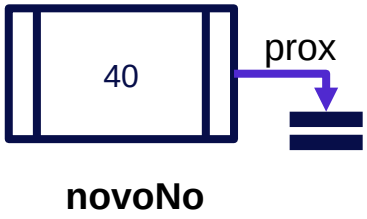
Verifica se existe elemento na fila
Remove sempre do inicio.
Declara um no.
Atribui ao no= fila->inicio
Depois atualiza o início
fila->inicio = fila->inicio->prox
Depois desaloca o no (delete no;)

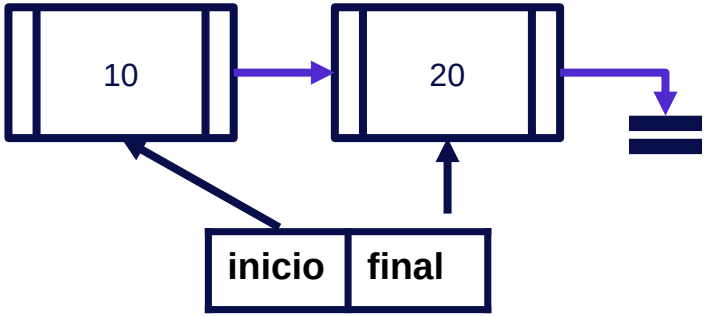


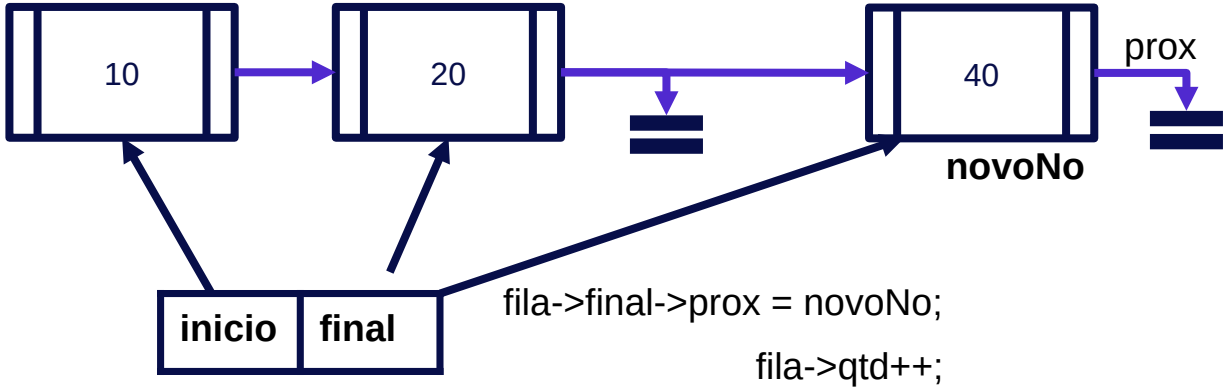
Remover elemento...

Verifica se existe elemento na fila
Remove sempre do inicio.
Declara um no.
Atribui ao no= fila->inicio
Depois atualiza o início
fila->inicio = fila->inicio->prox
Se o inicio for nulo, atribui nulo ao fim
Depois desaloca o no (delete no;)

➢ Inserção do elemento 40 em um fila que possui elementos

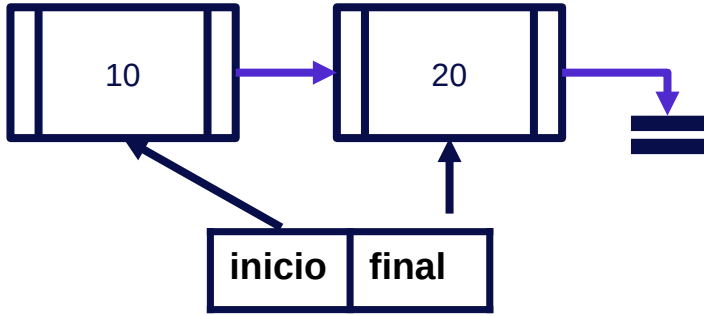


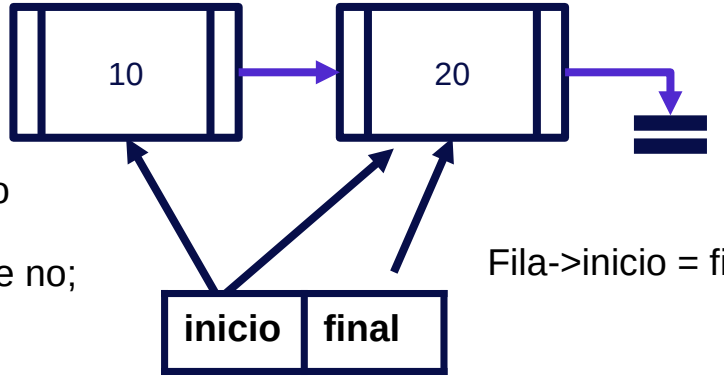





```
fila->final->prox = novoNo;
fila->qtd++;
```

➢ Remoção de um elemento – Sempre no Início





No= fila->inicio

Delete no;

Fila->inicio = fila->inicio->prox

```cpp
#include <iostream>
#include <windows.h>
#include "FilaDinamica.h"

using namespace std;
char menuInicial();
char menuSaida();

int main(){
    SetConsoleCP(1252);
    SetConsoleOutputCP(1252);
    Aluno alunoN;
    char menu;
    int x, tam;
    Fila* fila = criarFila();
    do{
        menu=menuInicial();
        switch(menu){
        case '1':
            cadastrarAluno(&alunoN);
            x=inserirFila (fila, &alunoN);
            break;
        case '2':
            x=removerFila(fila);
            break;
       case '3':
           imprimirFila(fila);
            break;
        case '4': consultarFila(fila,&alunoN);
            break;
        case '5': tam = tamanhoFila(fila);
            break;
        default: cout<< "Opção inválida!";
        }
        menu = menuSaida();
        system("clear||cls");
    }while(menu!='S');
    liberarFila(fila);
    return 0;
}
```

```
41  char menuInicial(){
42      char menu;
43      cout <<"\n --------- Menu --------\n"
44              "1 - para inserir aluno na fila\n"
45              "2 - para remover um aluno da fila\n"
46              "3 - exibir os alunos cadastrados\n"
47              "4 - consultar primeiro aluno.\n"
48              "5 - consultar número de aluno cadastrado.\n"
49              "--> ";
50      fflush(stdin);
51      cin>>menu;
52      return menu;
53  }
54
55  char menuSaida(){
56      char menu;
57      cout<<"\nDeseja sair do programa? S para sim "
58              "e qualquer tecla para continuar...\n--> ";
59      cin >> menu;
60      menu = toupper(menu);
61      return menu;
62  }
```

```c
#ifndef FILADINAMICA_H_INCLUDEI
#define FILADINAMICA_H_INCLUDEI
struct ALUNO{
    int matricula;
    char nome[30];
    float nota;
};
typedef struct ALUNO Aluno;
struct elemento{
    Aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
struct FILA{
    Elem *inicio;
    Elem *final;
    int qtd;
};
typedef struct FILA Fila;

Fila* criarFila();
void liberarFila(Fila* fila);
int consultarFila(Fila* fila, Aluno *alunoN);
int inserirFila(Fila* fila, Aluno *alunoN);
int removerFila(Fila* fila);
int tamanhoFila(Fila* fila);
int FilaEhVazia(Fila* fila);
int FilaEhCheia(Fila* fila);
int imprimirFila(Fila* fila);
void cadastrarAluno(Aluno *alunoN);
#endif // FILADINAMICA_H_INCLUDED
```

```cpp
#include <iostream>
#include "FilaDinamica.h" //inclui os Protótip
using namespace std;

Fila* criarFila(){
    Fila* fila = new Fila;
    if(fila != nullptr){
        fila->final = nullptr;
        fila->inicio = nullptr;
        fila->qtd = 0;
    }
    return fila;
}

void liberarFila(Fila *fila){
    if(fila != nullptr){
        Elem* no;
        while(fila->inicio != nullptr){
            no = fila->inicio;
            fila->inicio = fila->inicio->prox;
            delete no;
        }
        delete fila;
    }
}
```

```cpp
int consultarFila(Fila* fila, Aluno *alunoN){
    if(fila == nullptr){
        cout << "Fila não existe!\n";
        return 0;
    }
    if(fila->inicio == nullptr){//fila vazia
        cout << "Fila Vazia!\n";
        return 0;
    }
    *alunoN = fila->inicio->dados;
    return 1;
}
```

```
39
40  int inserirFila(Fila* fila, Aluno *alunoN){
41      if(fila == nullptr){
42          cout << "Fila não existe!\n";
43          return 0;
44      }
45      Elem *no = new Elem;
46      if(no == nullptr){
47          cout << "Espaço de memória não alocado para o nó!\n";
48          return 0;
49      }
50      no->dados = *alunoN;
51      no->prox = nullptr;
52      if(fila->final == nullptr)//fila vazia
53          fila->inicio = no;
54      else
55          fila->final->prox = no;
56      fila->final = no;
57      fila->qtd++;
58      return 1;
59  }
```

```
61  int removerFila(Fila* fila){
62       if(fila == nullptr){
63           cout << "Fila não existe!\n";
64           return 0;
65       }
66       if(fila->inicio == nullptr){//fila vazia
67           cout << "Fila Vazia!\n";
68           return 0;
69       }
70       Elem *no = fila->inicio;
71       fila->inicio = fila->inicio->prox;
72       if(fila->inicio == nullptr)//fila ficou vazia
73           fila->final = nullptr;
74       delete no;
75       fila->qtd--;
76       return 1;
77  }
78
```

```
78
79  int tamanhoFila(Fila* fila){
80      if(fila == nullptr){
81          cout << "Fila não existe!\n";
82          return 0;
83      }
84      cout << fila->qtd << " cadastro(s) na fila!\n";
85      return fila->qtd;
86  }
87
88  int FilaEhVazia(Fila* fila){
89       if(fila == nullptr){
90          cout << "Fila não existe!\n";
91          return 0;
92      }
93      if(fila->inicio == nullptr){//fila vazia
94          cout << "Fila Vazia!\n";
95          return 1;
96      }
97  }
98
```

```
98
99  int imprimirFila(Fila* fila){
100     if(fila == nullptr){
101         cout << "Fila não existe!\n";
102         return 0;
103     }
104     if(fila->inicio == nullptr){//fila vazia
105         cout << "Não há elementos na Fila!\n";
106         return 0;
107     }
108     Elem* no = fila->inicio;
109     while(no != nullptr){
110         cout << "Nome:" << no->dados.nome<< "\n";
111         cout << "Matricula: " << no->dados.matricula << "\n";
112         cout <<"Nota: " <<no->dados.nota << "\n\n\n";
113         no = no->prox;
114     }
115     return 1;
116 }
117
```

```
117
118 void cadastrarAluno(Aluno *alunoN){
119     cout << "Cadastro Aluno:\n";
120     cout <<"Nome: ";
121     fflush(stdin);
122     cin.getline(alunoN->nome, sizeof(alunoN->nome));
123     cout << "Matrícula: ";
124     cin >> alunoN->matricula;
125     cout << "Nota: ";
126     cin >> alunoN->nota;
127  }
128
```

Exercícios

Referências

EDELWEISS, N,; GALANTE, R.. Estruturas de dados. Porto Alegre: Bookman, 2009.

SZWARCFITER, J. L.; MARKENZON, L. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro: LTC, 2010.

ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C. 3. ed. rev. e ampl. São Paulo: Cengage Learning, 2011.

Aulas e vídeo aulas do professor André Backes: https://www.facom.ufu.br/~backes/