INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ESTRUTURA DE DADOS APLICADA

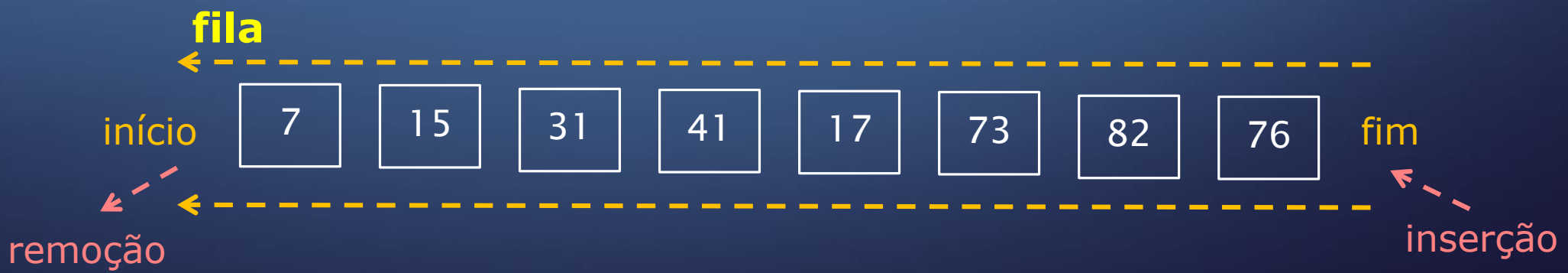
FILAS

IMPLEMENTAÇÃO DE FILAS COMO LISTA DINÂMICA ENCADEADA E NÓ DESCRITOR

PROFESSORA RESPONSÁVEL:
José Arnaldo Mascagni de Holanda
CONTATOS: arnaldomh@ifsp.edu.br

FILA


sai da fila
entra na fila


fila
início
7
15
31
41
17
73
82
76
fim
remoção
inserção

FILAS – CARACTERÍSTICAS

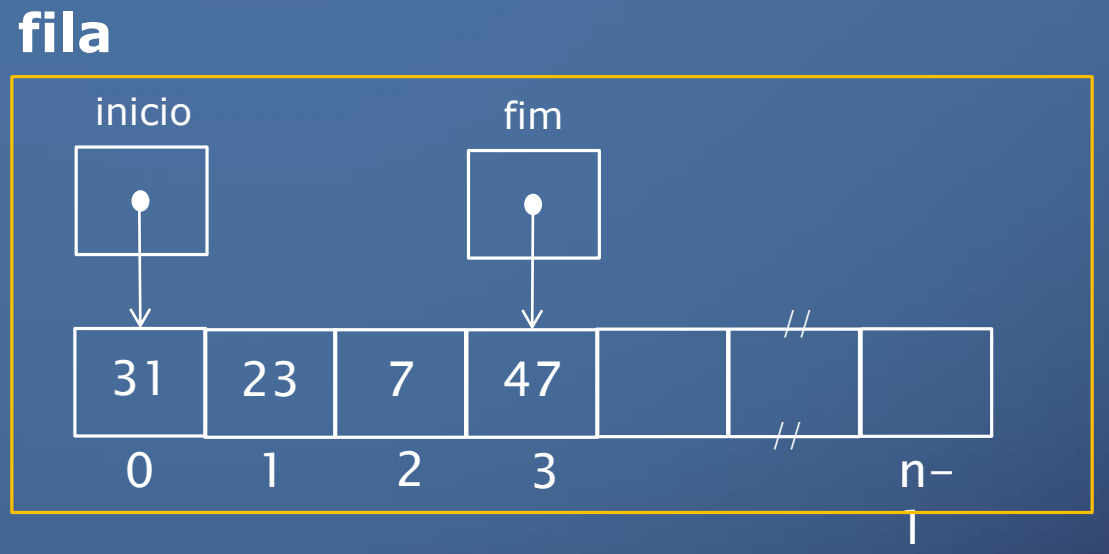
- Fila é uma estrutura de dados que utiliza o algoritmo *first in, first out* – FIFO.
  - ✓A inserção de novos elementos ocorre sempre no final.
  - ✓A remoção de elementos ocorre sempre no início da fila.

- Exemplos de aplicação:
  - Gerenciamento de recursos compartilhados (com mesma prioridade): senha de espera em restaurante, impressora, transações de bancos de dados etc.

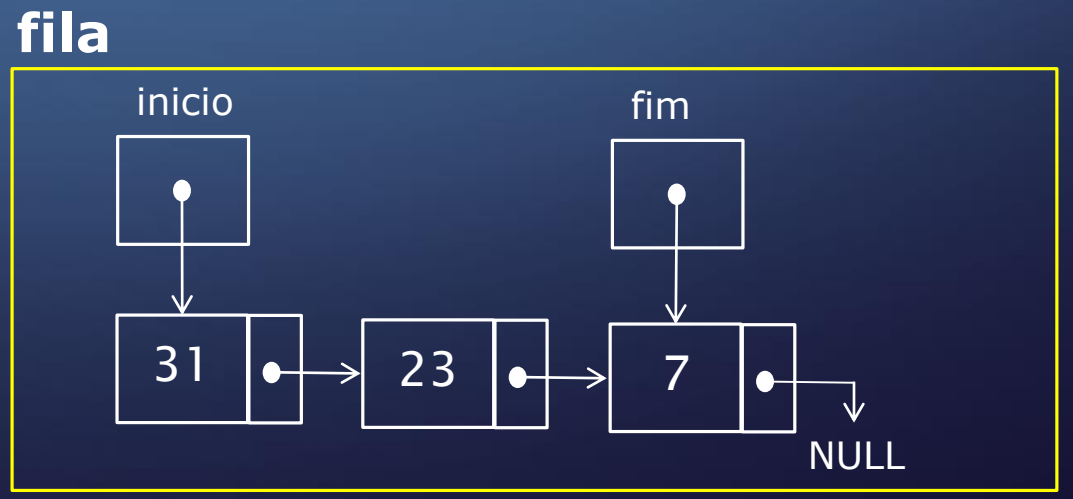
- Operações básicas (limitadas): inserção no final; remoção no início; acesso ao início.

IMPLEMENTAÇÃO DE FILA

- Com uso de lista estática (vetor):



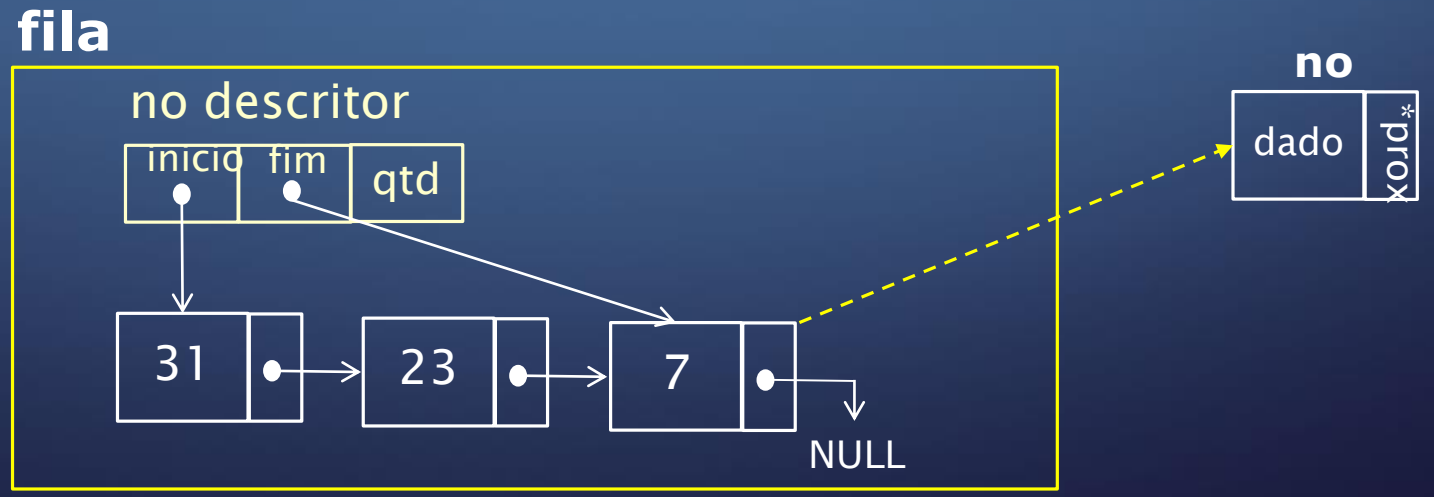
- Com uso de lista dinâmica:



IMPLEMENTAÇÃO DE FILA USANDO LISTA E NÓ DESCRITOR

- Nó descritor é um nó especial para armazenar informações sobre a lista, tais como:
  - Ponteiro para início da lista;
  - Ponteiro para fim da lista;
  - Quantidade de elementos na lista;
  - Cálculos diversos etc.

- Nó descritor substitui o apontador lista* (**)



IMPLEMENTAÇÃO: ESTRUTURAS

main.c | FilaDinamica.c | FilaDinamica.h

```c
#include <stdio.h>
#include <stdlib.h>
#include "filaDinamica.h"

struct tipoNo{
    struct elemento dado;
    struct tipoNo *prox;
};
typedef struct tipoNo noFila;

//Nó Descritor da Fila com 3 campos
struct tipoFila{
    struct tipoNo *inicio;
    struct tipoNo *fim;
    int qtd;
};
```

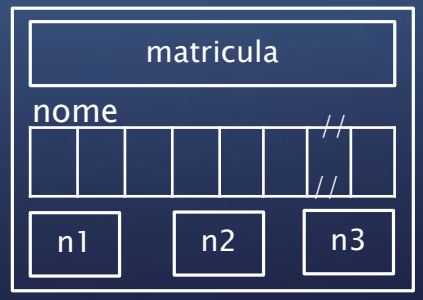
tipoNo ou noFila

main.c | FilaDinamica.c | FilaDinamica.h

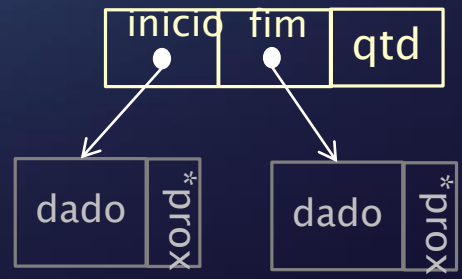
```c
struct elemento{
    int matricula;
    char nome[40];
    float n1, n2, n3;
};

typedef struct tipoFila Fila; //NÃO É PONTEIRO P/ PONTEIRO **
```

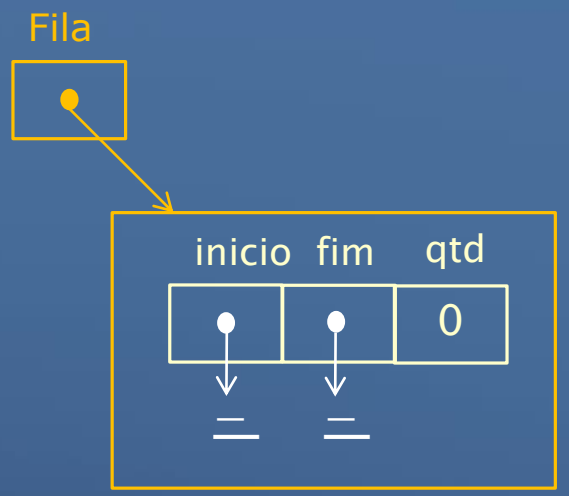
elemento



Fila



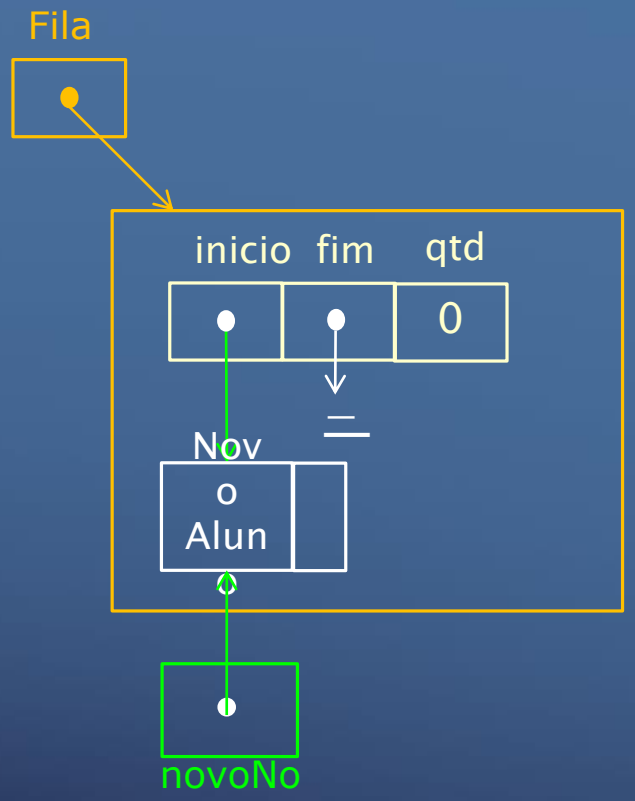
IMPLEMENTAÇÃO: CRIAÇÃO DA FILA



1. Criar a estrutura fila dinamicamente (*malloc*);
2. Inicializar os campos do nó construtor.

```
Fila* fila = (Fila*) malloc(sizeof(Fila));
if(fila != NULL){
    fila->fim = NULL;
    fila->inicio = NULL;
    fila->qtd = 0;
}
```

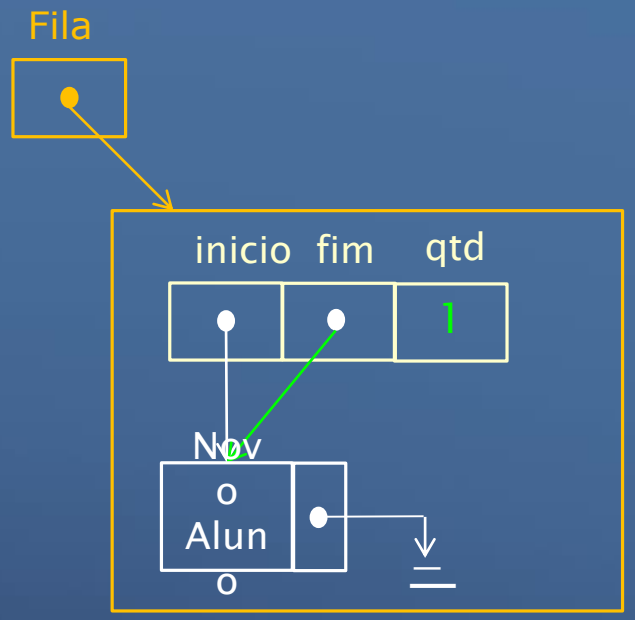
IMPLEMENTAÇÃO: INSERÇÃO NA FILA VAZIA (PASSOS 1 E 2)



1. Criar novoNo e preencher seus campos.
2. Atualizar fila->início que apontará para novoNo.
3. Atualizar demais campos do nó descritor (fila->fim) e qtd.

```
novoNo->dado = aluno;
novoNo->prox = NULL;
if(fila->fim == NULL)//fila vazia
    fila->inicio = novoNo;
```

IMPLEMENTAÇÃO: INSERÇÃO NA FILA VAZIA (PASSO 3)

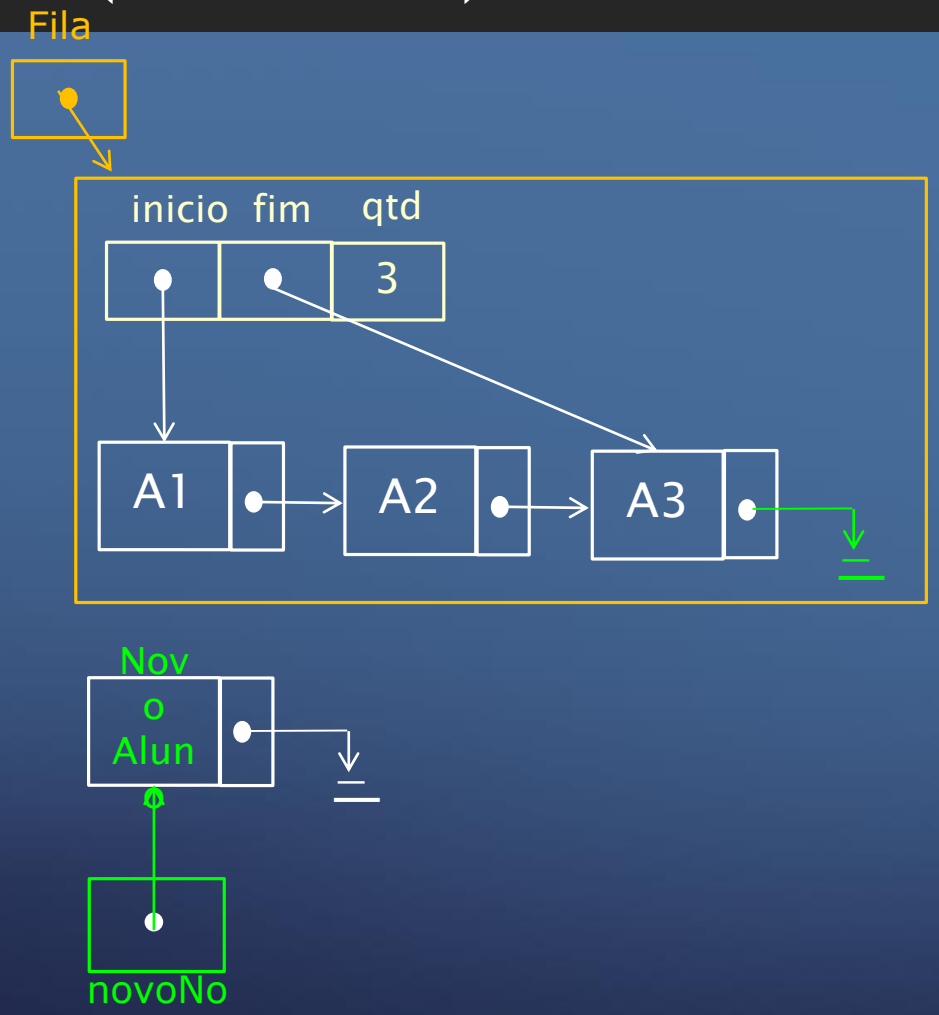


1. Criar novoNo e preencher seus campos.
2. Atualizar fila->início que apontará para novoNo.
3. Atualizar demais campos do nó descritor (fila->fim) e qtd.

```
novoNo->dado = aluno;
novoNo->prox = NULL;
if(fila->fim == NULL)//fila vazia
    fila->inicio = novoNo;
```

```
fila->fim = novoNo;
fila->qtd++;
```

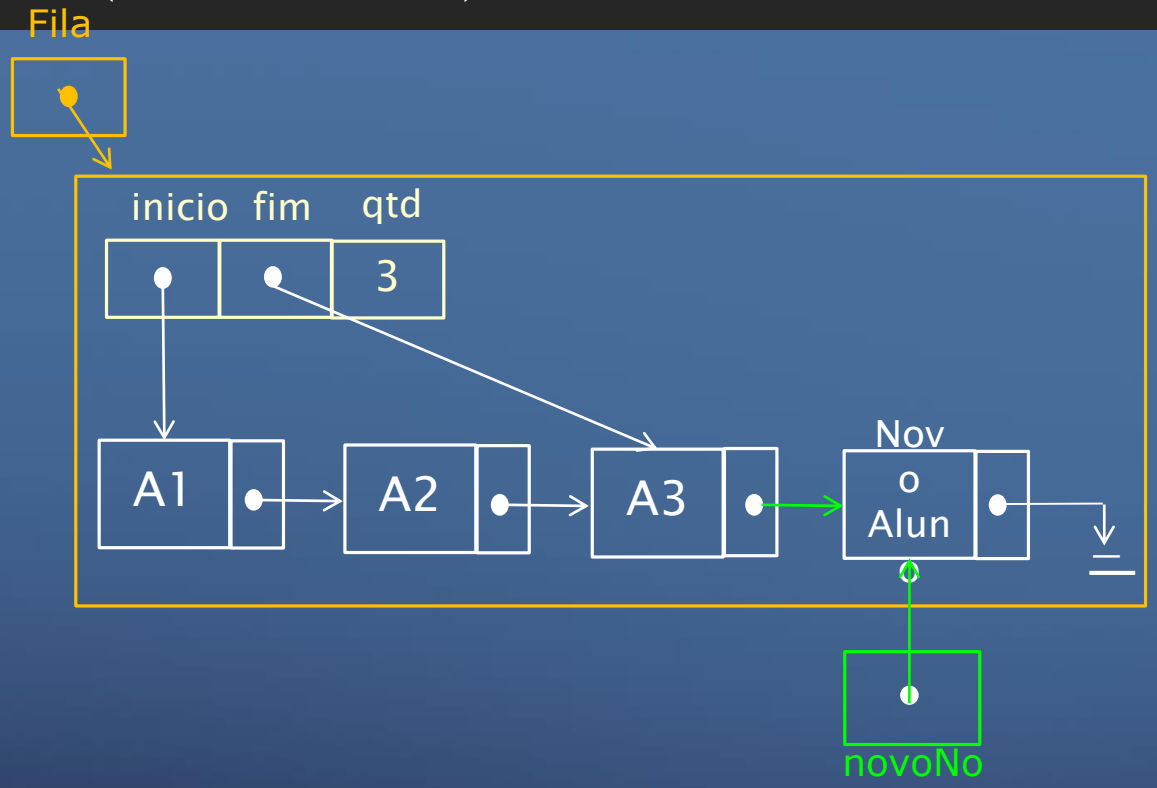
IMPLEMENTAÇÃO: INSERÇÃO NA FILA NÃO VAZIA (PASSO 1)



1. Criar novoNo e preencher seus campos.
2. Atualizar "fim" que apontará para novoNo.
3. Atualizar demais campos do nó descritor.

```
novoNo->dado = aluno;
novoNo->prox = NULL;
```

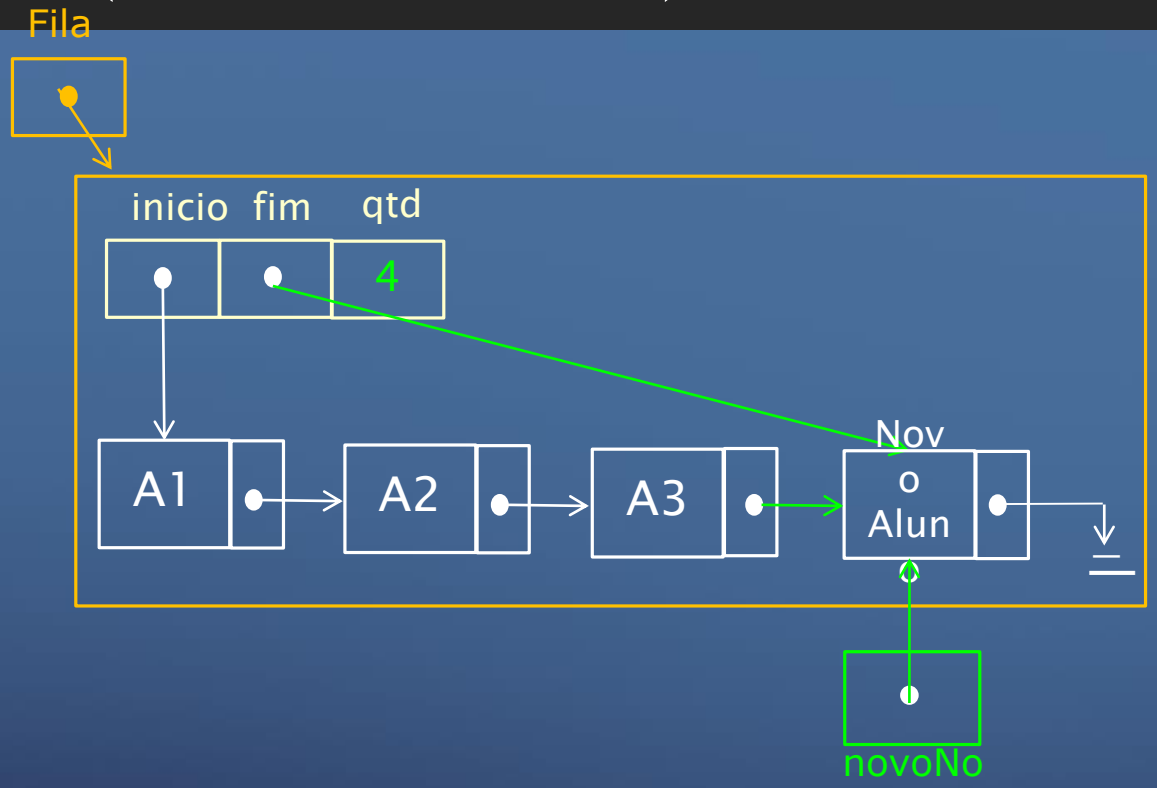
IMPLEMENTAÇÃO: INSERÇÃO NA FILA NÃO VAZIA (PASSO 2)



1. Criar novoNo e preencher seus campos.
2. Atualizar o último nó (apontado por "fim") que apontará para novoNo.
3. Atualizar demais campos do nó descritor.

```
else
    fila->fim->prox = novoNo;
fila->fim = novoNo;
fila->qtd++;
```

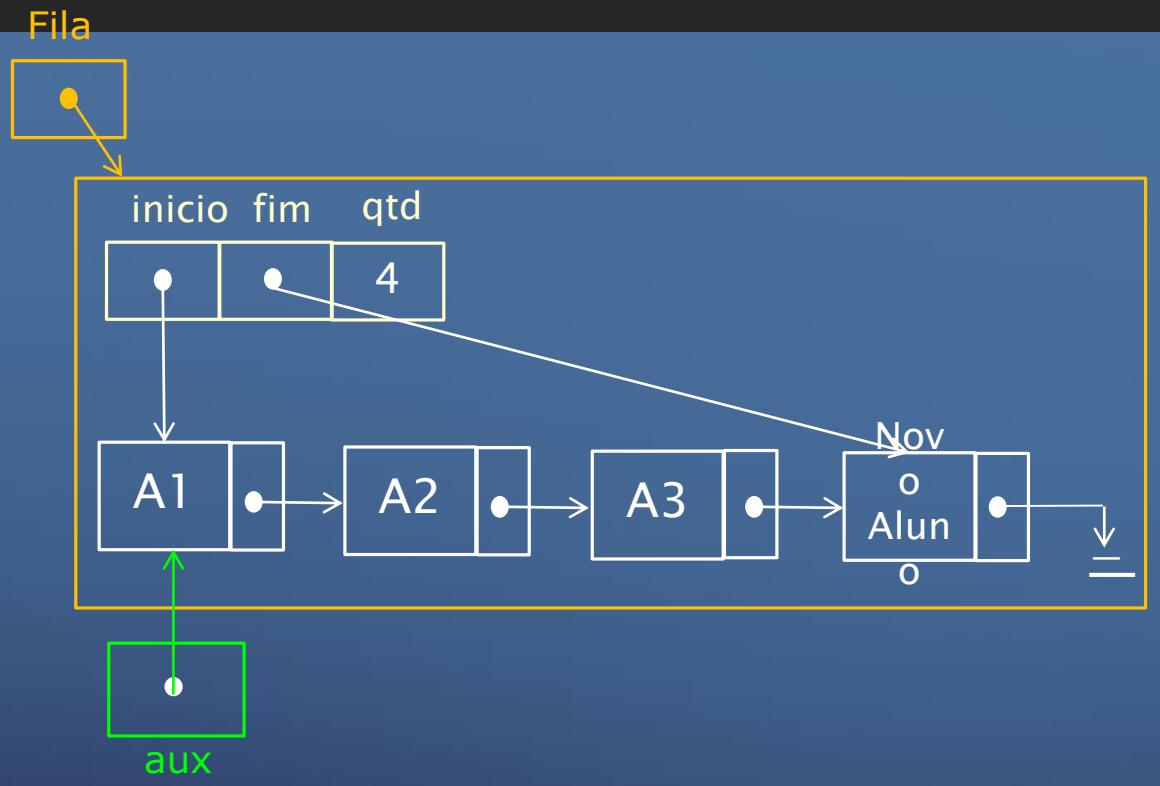
IMPLEMENTAÇÃO: INSERÇÃO NA FILA NÃO VAZIA (PASSOS 2 E 3)



1. Criar novoNo e preencher seus campos.
2. Atualizar o último nó (apontado por "fim") que apontará para novoNo.
3. Atualizar demais campos do nó descritor.

```
else
    fila->fim->prox = novoNo;
fila->fim = novoNo;
fila->qtd++;
```

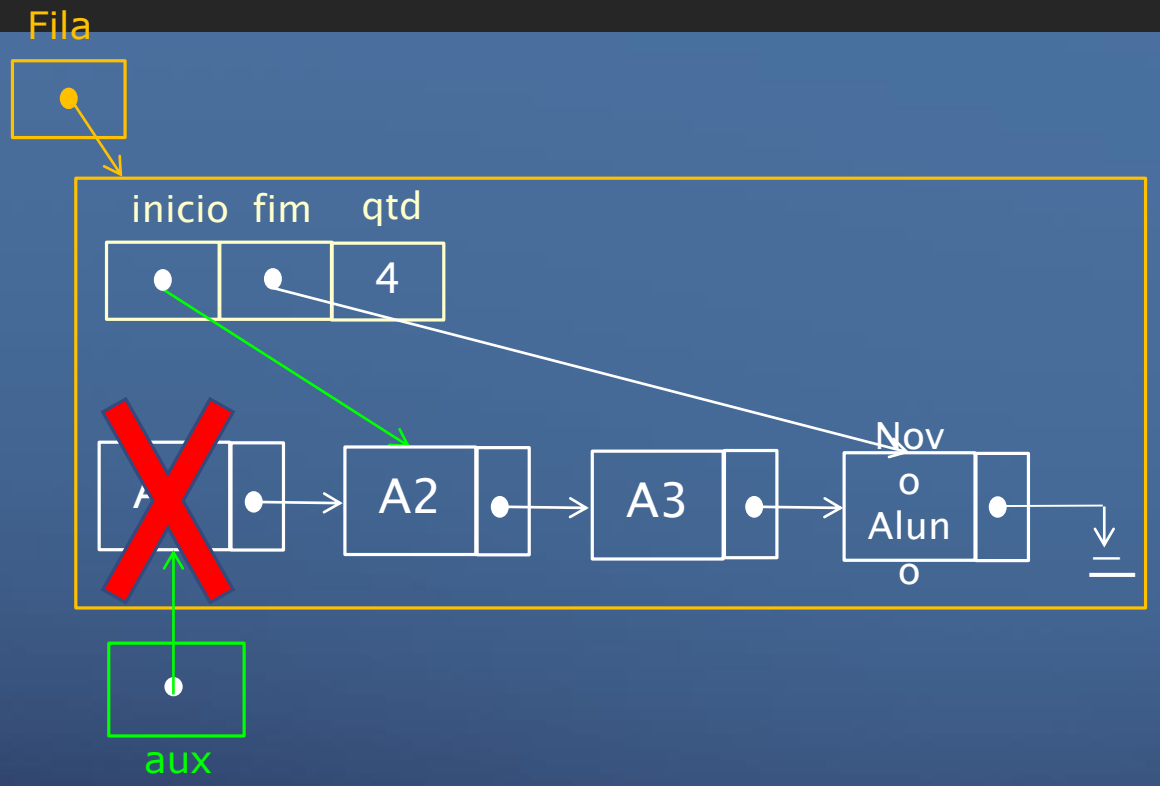
IMPLEMENTAÇÃO: REMOÇÃO DA FILA (PASSO 2 E 3)



1. Não remover de fila vazia.
2. Criar nó auxiliar e apontar para início.
3. Atualizar inicio para apontar para o segundo elemento.
4. Liberar nó auxiliar. Verificar antes se foi apagado último elemento da fila e, neste caso, atualizar fim = NULL.
5. Atualiza qtd de elementos no nó descritor.

```
noFila *aux = fila->inicio;
fila->inicio = fila->inicio->prox;
```

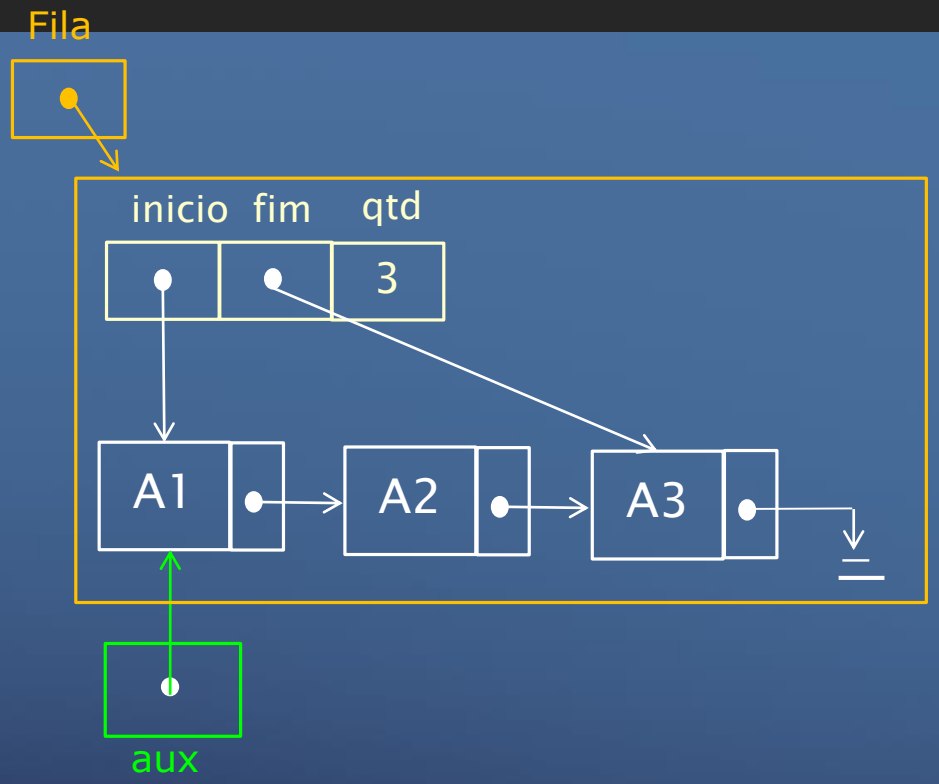
IMPLEMENTAÇÃO: REMOÇÃO DA FILA (PASSO 3 E 4)



1. Não remover de fila vazia.
2. Criar nó auxiliar e apontar para início.
3. Atualizar inicio para apontar para o segundo elemento.
4. Liberar nó auxiliar. Verificar antes se foi apagado último elemento da fila e, neste caso, atualizar fim = NULL.
5. Atualiza qtd de elementos no nó descritor.

```
if(fila->inicio == NULL)//fila ficou vazia
    fila->fim = NULL;
free(aux);
```

IMPLEMENTAÇÃO: APAGAR A FILA



1. Enquanto houver elementos na fila:
   i. Nó auxiliar aponta para o 1º elemento.
   ii. Alterar "inicio" para o segundo elemento.
   iii. Liberar primeiro nó da fila (aux).
2. Liberar a cabeça da fila.

```
noFila* aux;
while(fila->inicio != NULL){
    aux = fila->inicio;
    fila->inicio = fila->inicio->prox;
    free(aux);
}
free(fila);
```

DÚVIDAS?

VOCÊ FAZ....

Dado os protótipos, implemente todas as operações de manipulação de fila dinâmica encadeada com uso de nó descritor.

```
 9   Fila* criaFila();
10
11   void liberaFila(Fila* fila);
12
13   //consulta apenas 1º nó fila->inicio->dado;
14   //não faz sentido consultar/retornar outros nós em fila
15   int consultaFila(Fila* fila, struct elemento *aluno);
16
17   int insereFila(Fila* fila, struct elemento aluno);
18
19   void imprimeFila(Fila* fila);
20
21   int tamanhoFila(Fila* fila);
22
23   int filaVazia(Fila* fila);
24
25   int removeFila(Fila* fila);
```

EXERCÍCIO PARA FIXAÇÃO

Implemente um código para controle de pouso de aviões em um aeroporto. Utilize o conceito de fila dinâmica e considere uma característica adicional em relação à fila comum: prioridade. Faça o controle de prioridade de acordo com o combustível disponível nos aviões que aguardam na fila.

Observe que as operações de inserção, remoção e consulta deverão considerar a prioridade do elemento para ordenação.

Implemente utilizando lista dinâmica encadeada.