



RAFAEL VIEIRA COELHO



FILAS

SUMÁRIO

- Estruturas de Dados
- Listas Simplesmente Encadeada
- Listas Duplamente Encadeada
- **Filas**
- Pilhas
- Grafos
- Árvores

ANDRÉ BACKES

Estrutura de dados descomplicada em linguagem

C

DEFINIÇÃO DE FILA

O conceito de fila, ou fila de espera, é algo bastante comum para as pessoas. Afinal, somos obrigados a enfrentar uma fila sempre que vamos ao banco, ao cinema etc.

Na computação, uma fila nada mais é do que um conjunto finito de itens (de um mesmo tipo) esperando por um serviço ou processamento.



Um exemplo bastante comum da aplicação de filas é o gerenciamento de documentos enviados para a impressora.

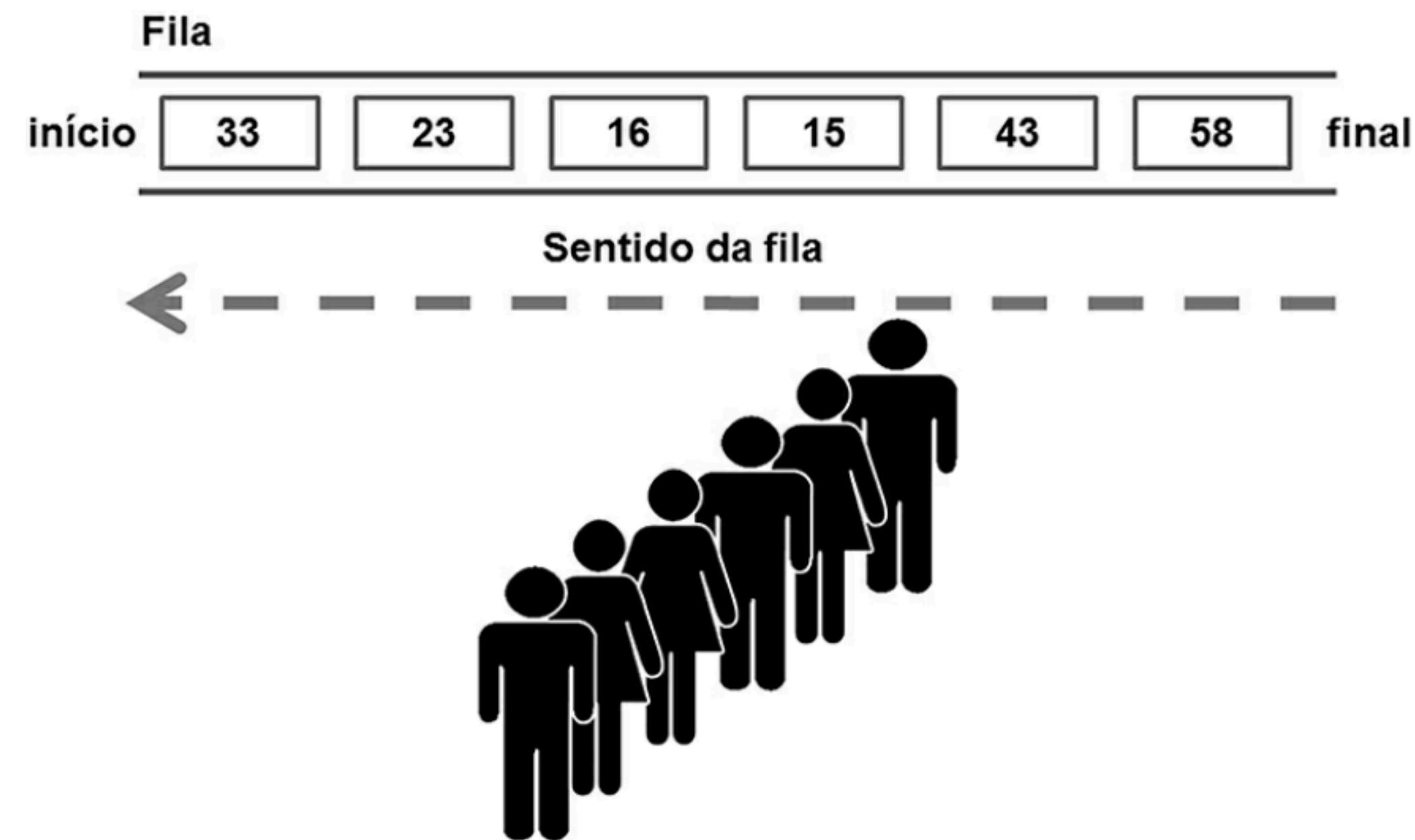


As filas são implementadas e se comportam de modo muito similar às listas, sendo, muitas vezes, consideradas um tipo especial de lista em que a inserção e a remoção são realizadas sempre em extremidades distintas.

DEFINIÇÃO DE FILA

A inserção de um item é feita de um lado da fila, enquanto a retirada é feita do outro lado. Desse modo, se quisermos acessar determinado elemento da fila, deveremos remover todos os que estiverem à frente dele.

Por esse motivo, as filas são conhecidas como estruturas do tipo primeiro a entrar, primeiro a sair ou **FIFO (First In First Out)**: os elementos são removidos da fila na mesma ordem em que foram inseridos.



OPERAÇÕES BÁSICAS DE UMA FILA

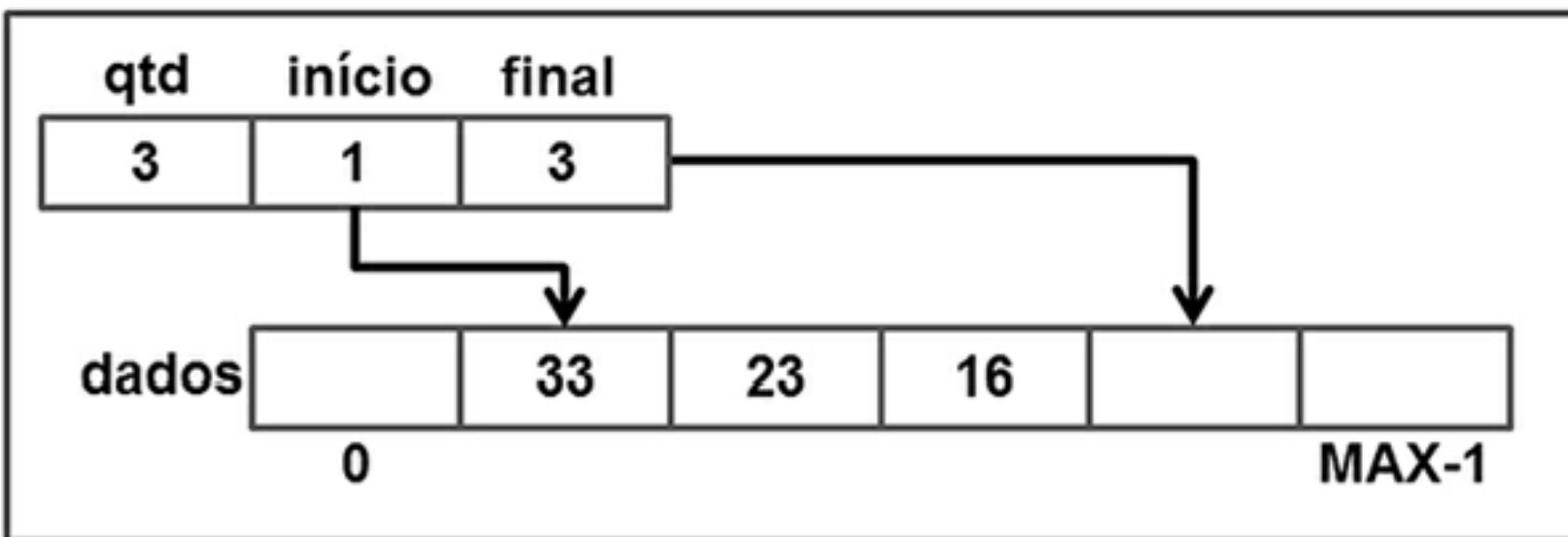
- Criação da fila.
- Inserção de um elemento no final da fila.
- Remoção de um elemento do início da fila.
- Acesso ao elemento do início da fila.
- Destruuição da fila.
- Além de informações com tamanho, se a fila está cheia ou vazia.

I) FILA SEQUENCIAL ESTÁTICA



Além do array, essa fila utiliza três campos adicionais para guardar o **início**, o **final** e a **quantidade** de elementos (**dados**) inseridos na fila.

```
Fila *fi;
```



A principal vantagem de se utilizar um array na definição de uma **fila sequencial estática** é a facilidade de criar e destruir a fila. Já a sua principal desvantagem é a necessidade de definir previamente o tamanho do array e, consequentemente, da fila.

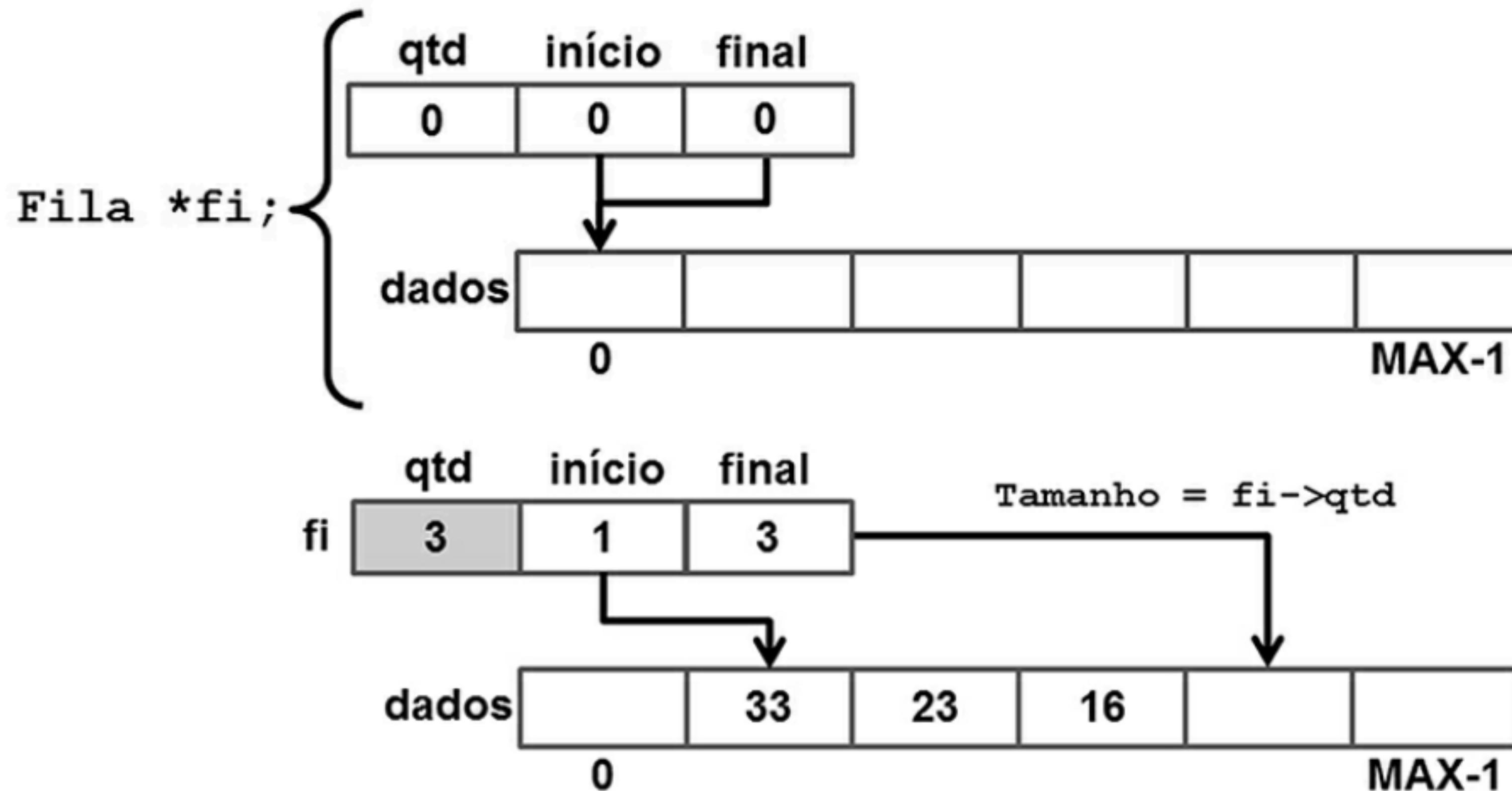
Arquivo FilaEstatica.h

```
01 #define MAX 100
02 struct aluno{
03     int matricula;
04     char nome[30];
05     float n1,n2,n3;
06 };
07 typedef struct fila Fila;
08
09 Fila* cria_Fila();
10 void libera_Fila(Fila* fi);
11 int consulta_Fila(Fila* fi, struct aluno *al);
12 int insere_Fila(Fila* fi, struct aluno al);
13 int remove_Fila(Fila* fi);
14 int tamanho_Fila(Fila* fi);
15 int Fila_vazia(Fila* fi);
16 int Fila_cheia(Fila* fi);
```

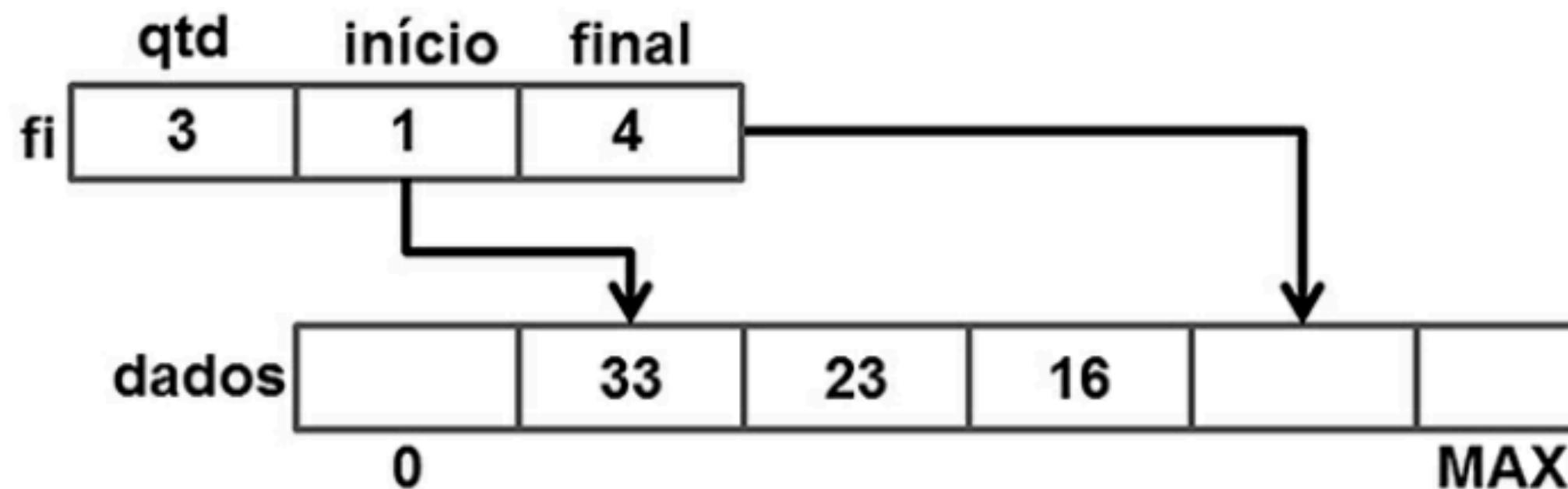
Arquivo FilaEstatica.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include "FilaEstatica.h" //inclui os protótipos
04 //Definição do tipo Fila
05 struct fila{
06     int inicio, final, qtd;
07     struct aluno dados[MAX];
08 };
```

CRIAÇÃO/TAMANHO DA FILA



INSERINDO UM ELEMENTO NA FILA



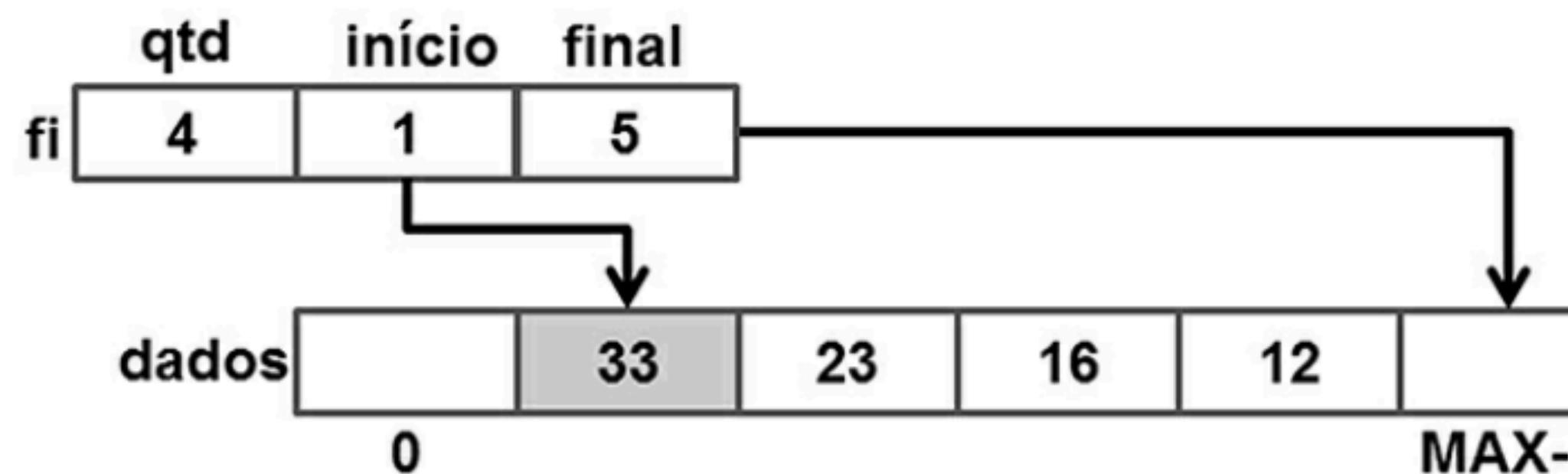
al 



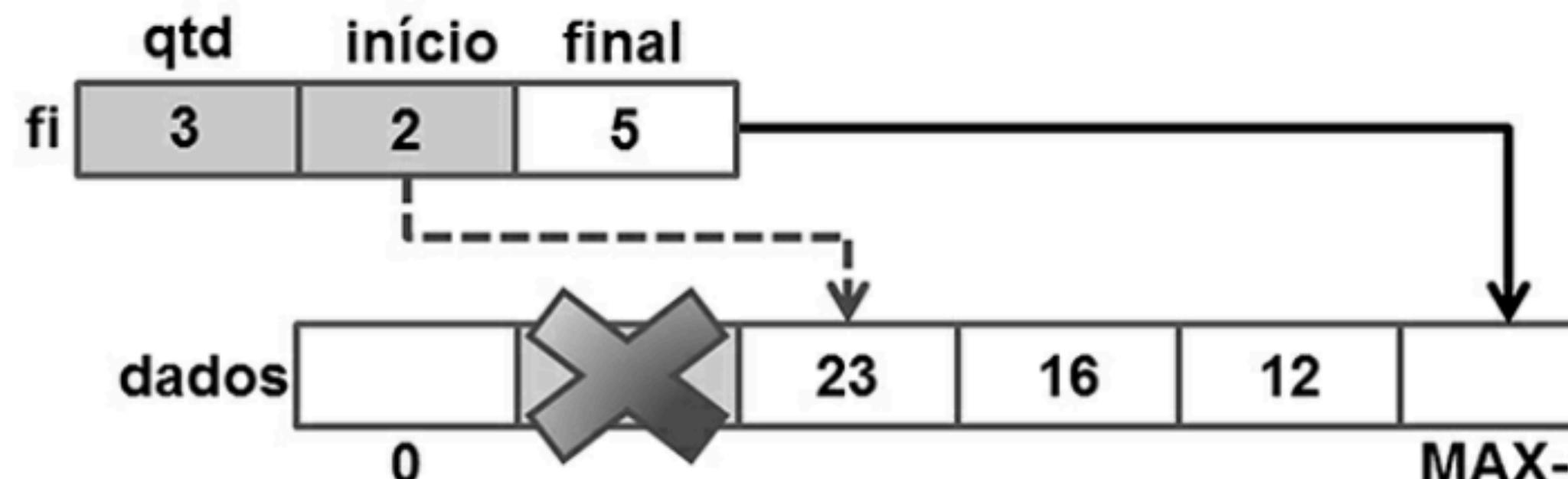
```
fi->dados[fi->final] = al;  
fi->final = (fi->final+1)%MAX;  
fi->qtd++;
```



REMOVENDO UM ELEMENTO DA FILA



fi->início = (fi->início+1) %MAX;
fi->qtd--;

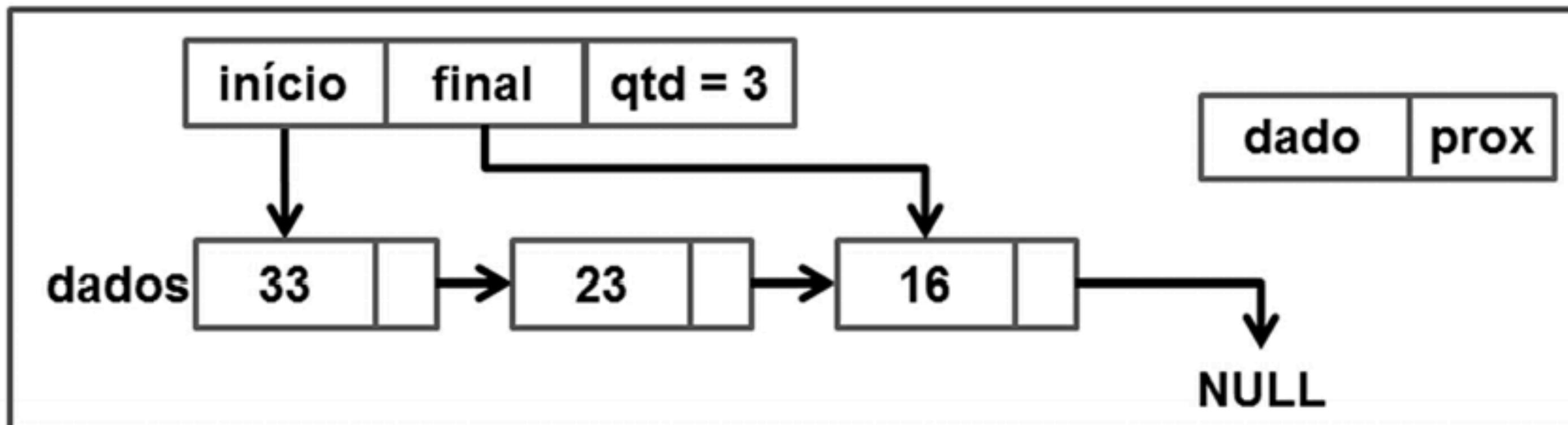


2) FILA DINÂMICA ENCADEADA



Além da estrutura que define seus elementos, essa fila utiliza um **nó descriptor** para guardar o **início**, o **final** e a **quantidade** de elementos (**dados**) inseridos na fila.

Fila *fi;



Arquivo FilaDin.h

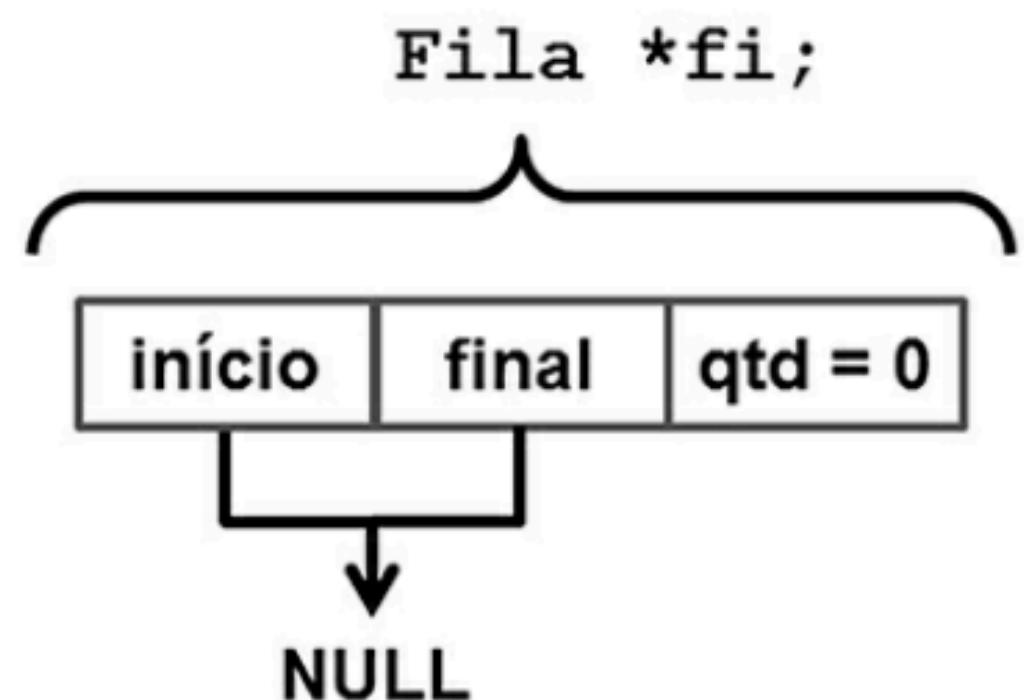
```
01 struct aluno{  
02     int matricula;  
03     char nome[30];  
04     float n1,n2,n3;  
05 };  
06 typedef struct fila Fila;  
07  
08 Fila* cria_Fila();  
09 void libera_Fila(Fila* fi);  
10 int consulta_Fila(Fila* fi, struct aluno *al);  
11 int insere_Fila(Fila* fi, struct aluno al);  
12 int remove_Fila(Fila* fi);  
13 int tamanho_Fila(Fila* fi);  
14 int Fila_vazia(Fila* fi);  
15 int Fila_cheia(Fila* fi);
```

Arquivo FilaDin.c

```
01 #include <stdio.h>  
02 #include <stdlib.h>  
03 #include "FilaDin.h" //inclui os Protótipos  
04 //Definição do tipo Fila  
05 struct elemento{  
06     struct aluno dados;  
07     struct elemento *prox;  
08 };  
09 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 };
```

Criando uma fila

```
01 Fila* cria_Fila(){
02     Fila* fi = (Fila*) malloc(sizeof(struct fila));
03     if(fi != NULL){
04         fi->final = NULL;
05         fi->inicio = NULL;
06         fi->qtd = 0;
07     }
08     return fi;
09 }
```

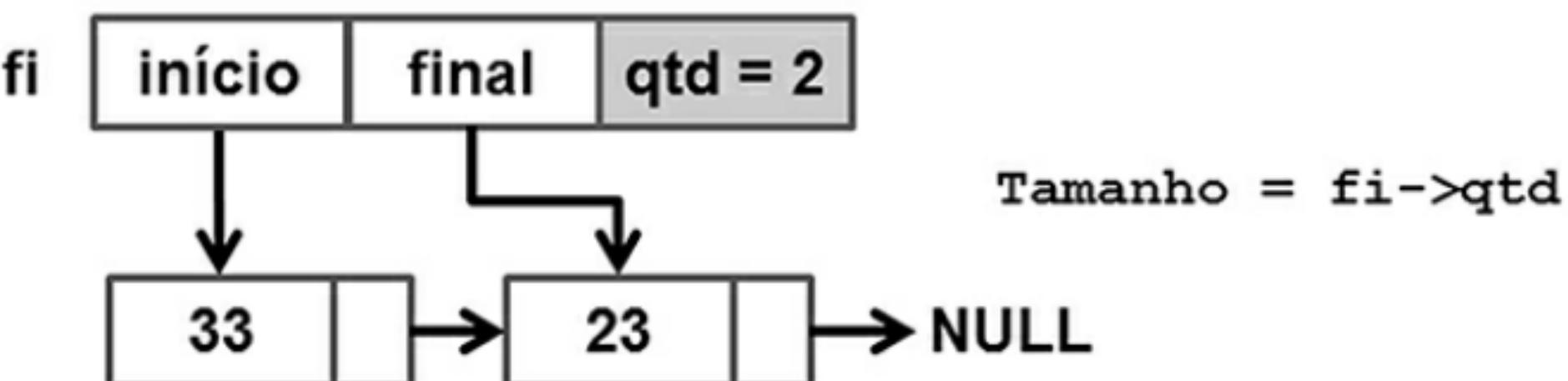


Destruindo uma fila

```
01 void libera_Fila(Fila* fi) {  
02     if(fi != NULL) {  
03         Elem* no;  
04         while(fi->inicio != NULL) {  
05             no = fi->inicio;  
06             fi->inicio = fi->inicio->prox;  
07             free(no);  
08         }  
09         free(fi);  
10     }  
11 }
```

Tamanho da fila

```
01 int tamanho_Fila(Fila* fi) {  
02     if(fi == NULL)  
03         return 0;  
04     return fi->qtd;  
05 }
```





Basicamente, retornar se uma **fila dinâmica encadeada** está vazia consiste em verificar se o valor do seu campo **inicio** é igual a **NULL**.



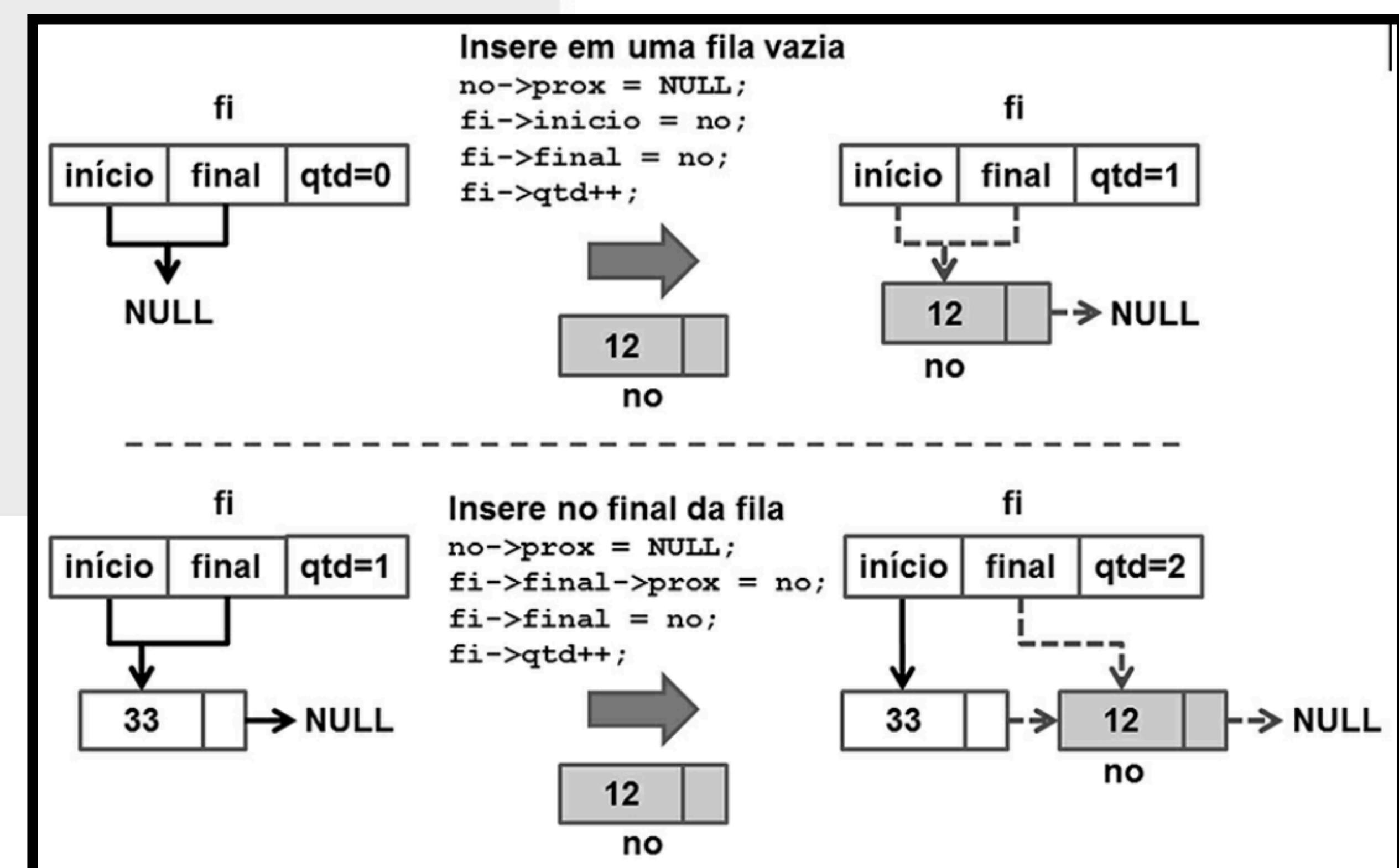
Outra maneira de saber se uma fila está vazia é verificando se o valor do campo **qtd** é igual a **ZERO**.

Retornando se a fila está vazia

```
01 int Fila_vazia(Fila* fi) {  
02     if(fi == NULL)  
03         return -1;  
04     if(fi->inicio == NULL)  
05         return 1;  
06     return 0;  
07 }
```

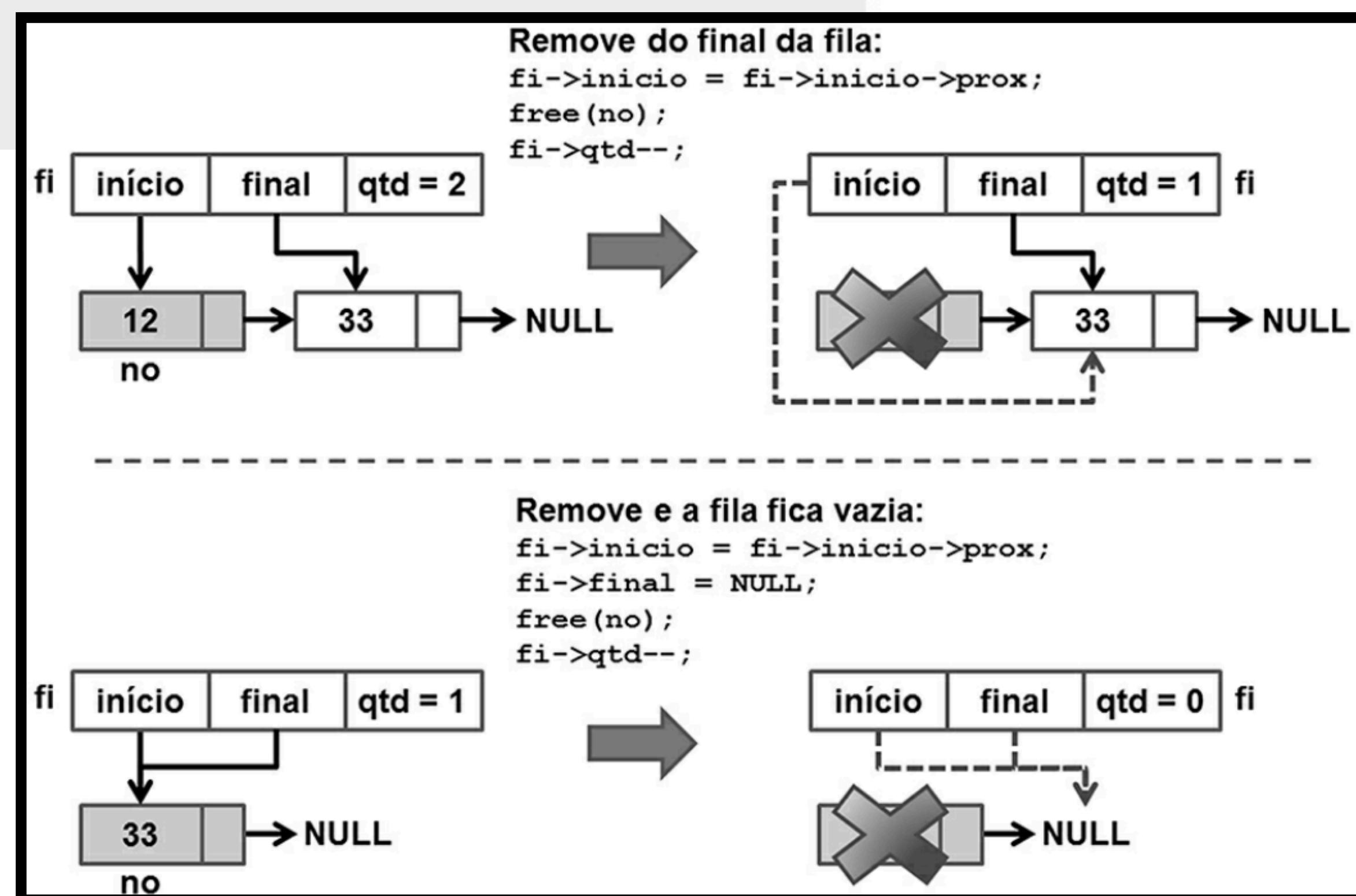
Inserindo um elemento na fila

```
01 int insere_Fila(Fila* fi, struct aluno al) {
02     if(fi == NULL)
03         return 0;
04     Elem *no = (ELEM*) malloc(sizeof(ELEM));
05     if(no == NULL)
06         return 0;
07     no->dados = al;
08     no->prox = NULL;
09     if(fi->final == NULL) //fila vazia
10         fi->inicio = no;
11     else
12         fi->final->prox = no;
13     fi->final = no;
14     fi->qtd++;
15     return 1;
16 }
```



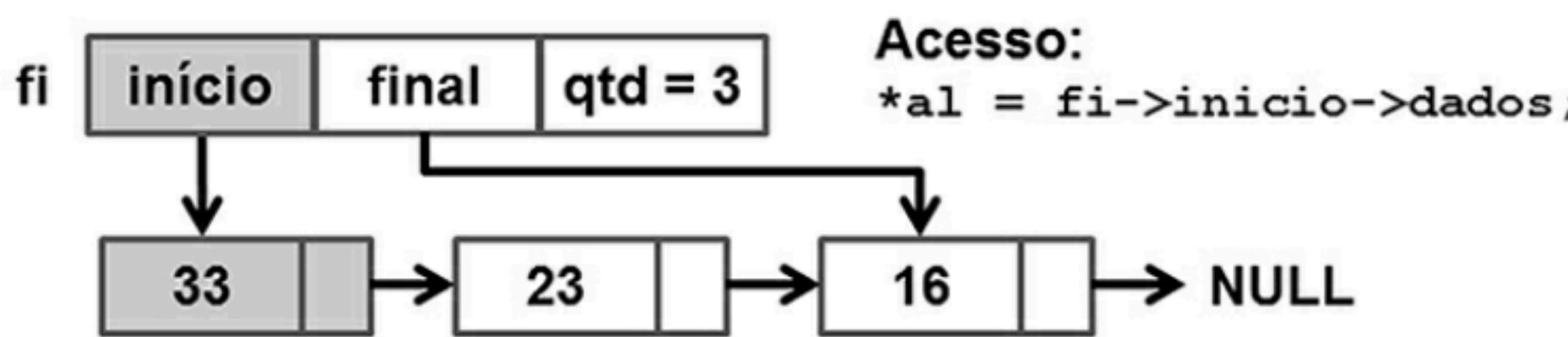
Removendo um elemento da fila

```
01 int remove_Fila(Fila* fi) {  
02     if(fi == NULL)  
03         return 0;  
04     if(fi->inicio == NULL)//fila vazia  
05         return 0;  
06     Elem *no = fi->inicio;  
07     fi->inicio = fi->inicio->prox;  
08     free(no);  
09     if(fi->inicio == NULL)//fila ficou vazia  
10         fi->final = NULL;  
11     fi->qtd--;  
12     return 1;  
13 }
```



Consultando a fila

```
01 int consulta_Fila(Fila* fi, struct aluno *al){  
02     if(fi == NULL)  
03         return 0;  
04     if(fi->inicio == NULL) //fila vazia  
05         return 0;  
06     *al = fi->inicio->dados;  
07     return 1;  
08 }
```



REUTILIZANDO O CÓDIGO

Arquivo FilaUsandoListaDinEncad.h

```
01 #include "ListaDinEncad.h"
02
03 typedef Lista Fila;
04
05 Fila* cria_Fila();
06 void libera_Fila(Fila* fi);
07 int consulta_Fila(Fila* fi, struct aluno *al);
08 int insere_Fila(Fila* fi, struct aluno al);
09 int remove_Fila(Fila* fi);
10 int tamanho_Fila(Fila* fi);
11 int Fila_vazia(Fila* fi);
12 int Fila_cheia(Fila* fi);
```

Arquivo FilaUsandoListaDinEncad.c

```
01 //inclui os Protótipos
02 #include "FilaUsandoListaDinEncad.h"
03 Fila* cria_Fila(){
04     return cria_lista();
05 }
06 void libera_Fila(Fila* fi){
07     libera_lista(fi);
08 }
09 int consulta_Fila(Fila* fi, struct aluno *al){
10     return consulta_lista_pos(fi,1,al);
11 }
12 int insere_Fila(Fila* fi, struct aluno al){
13     return insere_lista_final(fi,al);
14 }
15 int remove_Fila(Fila* fi){
16     return remove_lista_inicio(fi);
17 }
18 int tamanho_Fila(Fila* fi){
19     return tamanho_lista(fi);
20 }
21 int Fila_vazia(Fila* fi){
22     return lista_vazia(fi);
23 }
24 int Fila_cheia(Fila* fi){
25     return lista_cheia(fi);
26 }
```

TAREFAS

- 1) Implemente uma função para intercalar filas sequenciais estáticas. A função receberá duas filas (F_1 e F_2) e o resultado da intercalação deve ser colocado em F_1 . A fila F_2 deve ficar vazia.
- 2) Implemente uma função para unir filas sequenciais estáticas. A função receberá duas filas (F_1 e F_2) e o resultado da união deve ser colocado em F_1 , sem repetições. A fila F_2 deve ficar vazia.
- 3) Implemente uma função que copie os elementos de uma fila F_1 para uma fila F_2 (ambas sequenciais estáticas).



OBRIGADO!

RAFAELVC2@GMAIL.COM