



Estruturas de Dados 1

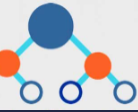
12 – Filas Estáticas e Dinâmicas

Antonio Angelo de Souza Tartaglia
angelot@ifsp.edu.br

Estrutura de Dados 1

Fila

- O conceito de Fila, é algo bastante comum, para nós humanos. Afinal, somos obrigados a enfrentar uma fila nas mais variadas situações, por exemplo, filas de bancos, cinema, caixas de supermercado, etc.
- Na computação, uma Fila nada mais é do que um conjunto finito de itens (de um mesmo tipo), aguardando por um serviço ou processamento. É um tipo de controle de fluxo muito comum na computação.
- Como exemplo de controle de fluxo utilizando as estruturas de dados Fila, temos o gerenciamento de documentos enviados para uma impressora, a **fila de impressão**.



Estrutura de Dados 1

Fila

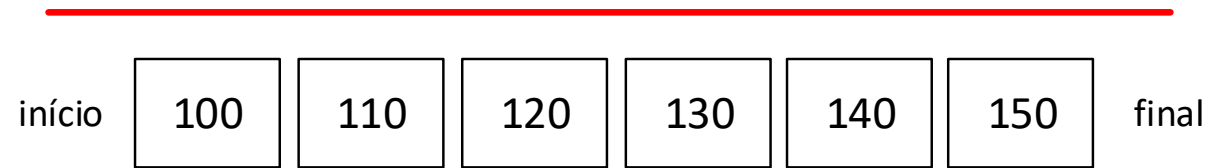
- Assim como as Listas (estáticas e dinâmicas), as Filas são uma sequência de elementos, mas diferentemente das Listas, os itens de uma Fila obedecem a uma ordem de entrada e saída.
- Um item somente pode ser retirado da Fila depois que todos os itens a sua frente também tenham sido retirados.
- Dessa forma, Filas são implementadas e se comportam de forma muito similar às Listas, e são consideradas muitas vezes, como um tipo especial de Lista, em que a inserção e remoção são realizadas sempre em extremidades distintas.
- Neste caso, a inserção de um item é feita de um lado da fila e a retirada dos itens é realizado do outro lado.



Estrutura de Dados 1

Fila

- Desse modo, se quisermos acessar determinado elemento da fila, é necessário remover todos que estiverem à sua frente. Por esse motivo, as filas são conhecidas como estruturas do tipo “**primeiro a entrar, primeiro a sair**”, ou FIFO (*First In, First Out*). Portanto, os elementos são removidos da fila na mesma ordem em que foram inseridos.



← Sentido da Fila

Se quisermos acessar o elemento 120, temos que remover antes os elementos 100 e 110...



Estrutura de Dados 1

Fila

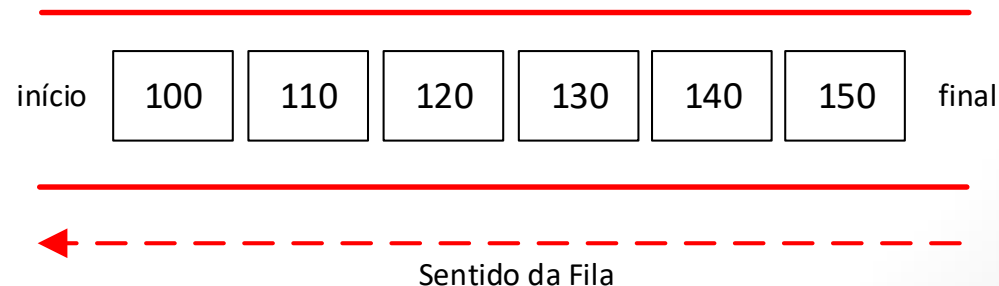
- Dessa forma, Fila é uma **estrutura de dados linear** utilizada para armazenar e controlar fluxo de dados em um computador.
- Uma estrutura do tipo Fila, é uma sequência de elementos do mesmo tipo. Seus elementos possuem estrutura interna abstraída, ou seja, sua complexidade é arbitrária e não afeta o seu funcionamento. Uma Fila pode possuir n (sendo $n \geq 0$) elementos ou itens, e, dependendo da aplicação, pode possuir elementos repetidos. Se $n == 0$, dizemos que a Fila está vazia.
- Aplicações
 - Qualquer aplicação onde se necessite de:
 - Controle de fluxo;
 - Recursos compartilhados (impressoras, transações de bancos de dados, etc);



Estrutura de Dados 1

Tipos de Filas

- Basicamente existem dois tipos de implementações para uma Fila. Essas implementações diferem entre si com relação ao tipo de alocação de memória utilizada e o tipo de acesso aos seus elementos. Uma Fila pode ser implementada usando:
 - Alocação estática com acesso sequencial, ou;
 - Alocação dinâmica com acesso encadeado



Fila

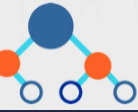
- Em uma Fila implementada utilizando o conceito de tipo abstrato de dado, podemos realizar as seguintes operações:
 - Criação da Fila;
 - Inserção de um elemento no final da Fila;
 - Remoção de um elemento no início da Fila;
 - Acesso ao elemento do início da Fila;
 - Destruição da Fila;
 - Além de informações como tamanho, se está cheia ou se está vazia.
- Essas operações também dependerão do tipo de alocação de memória que será utilizada, se alocação estática ou alocação dinâmica.



Estrutura de Dados 1

Fila

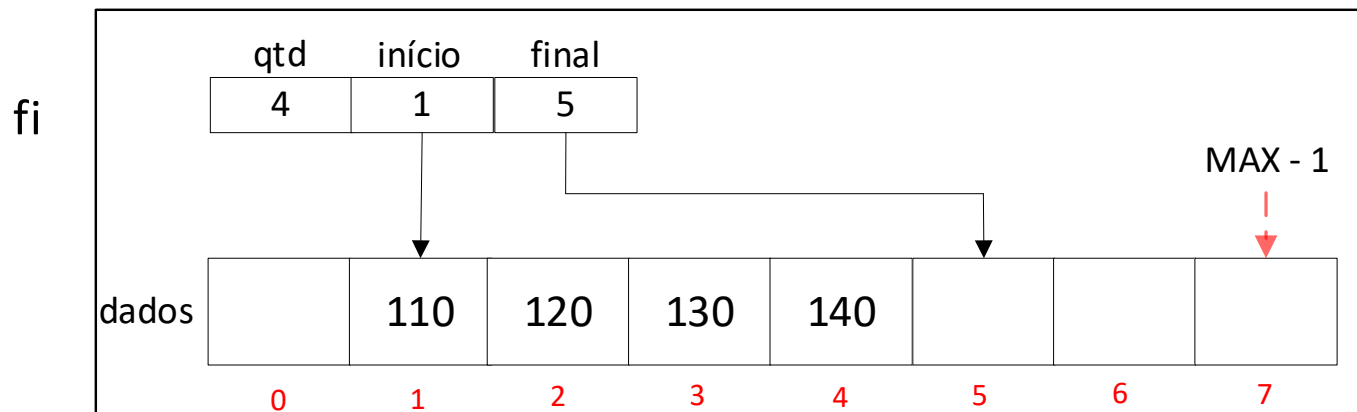
- Alocação Estática:
 - O Espaço de memória é alocado no momento da compilação do programa, ou seja, é necessário definir o número máximo de elementos que a Fila suportará.
 - Neste modo de alocação, os elementos são armazenados de forma consecutiva na memória, como em um *array* ou vetor, e a posição de um elemento pode ser facilmente obtida a partir do início da Fila, uma vez que todos os elementos estão armazenados de forma sequencial na memória;



Estrutura de Dados 1

Fila Sequencial Estática

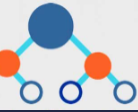
- Este tipo de Fila é definida utilizando-se a alocação estática e acesso sequencial dos elementos.
- Trata-se do tipo de fila mais simples possível de se implementar, e é definida utilizando-se um *array*, de modo que o sucessor de um elemento, ocupa a posição física em memória seguinte ao mesmo.
- Além do *array*, este tipo de Fila utiliza três campos adicionais para guardar o início, o final e a quantidade de elementos inseridos na Fila:



Estrutura de Dados 1

Fila Sequencial Estática

- Como a implementação deste tipo de Fila utiliza um tipo abstrato de dado, temos que o usuário/programador tem acesso apenas a um ponteiro através do módulo principal `main()`, o que torna o tipo de dado FILA, um tipo **opaco**.
- Isso o impede de ter conhecimento de como realmente a estrutura de dados Fila foi implementada, e limita o seu acesso aos dados nela armazenados, somente permitindo que se utilize das funções criadas especificamente para manipulação do início, final da Fila e consultas, completando assim o encapsulamento dos dados.



Estrutura de Dados 1

Fila Sequencial Estática

- A principal vantagem de se utilizar um *array* para a definição de uma Fila (neste caso sequencial estática), é a facilidade de criar e destruir esta estrutura de dados. Por outro lado, sua principal desvantagem é a necessidade de definir previamente o tamanho do *array*, e conseqüentemente o tamanho da Fila.
- Outro ponto, diz respeito à quantidade de memória alocada: será sempre fixa, estando a Fila cheia ou vazia, a mesma quantidade de memória sempre será utilizada.
- Considerando as vantagens e desvantagens, este tipo de Fila é ideal para utilização em pequenos conjuntos de dados, ou quando o tamanho máximo a ser suportado pela Fila está bem definido, e há garantias de que nunca será ultrapassado.



Estrutura de Dados 1

Fila Sequencial Estática



- Antes de se iniciar a implementação propriamente dita, é necessário definir qual o tipo de dado que será armazenado na Fila. Este tipo de estrutura de dados pode armazenar qualquer tipo de informação. Para isso, é necessário que em sua declaração, especifiquemos que tipo de informação, ou dado, ela armazenará.
- Como a Fila será implementada com o tipo de dado opaco, é necessário a sua definição. Este tipo, será então um ponteiro para a estrutura (struct) que define a Fila.
- Também será necessário definir o conjunto de funções que será visível para o usuário/programador que utilizará a biblioteca que estamos criando, completando assim o encapsulamento dos dados.

Estrutura de Dados 1

Fila Sequencial Estática

- Implementação das variáveis que serão utilizadas no programa:
 - A variável x será utilizada para armazenar as informações de execução devolvidas pelas funções de manipulação do tipo de dado Fila.
 - As variáveis al1, al2 e al3 comporão os dados de alunos inseridos na Fila, serão a massa de dados.
 - A variável al_consulta será utilizada para o retorno de dados da função consulta.

```
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "fila.h"

int main()
{
    int x; //para os codigos de erro
    ALUNO al_consulta, al1, al2, al3;
    al1.matricula = 100;
    al1.n1 = 8.3;
    al1.n2 = 8.4;
    al1.n3 = 8.5;

    al2.matricula = 110;
    al2.n1 = 7.3;
    al2.n2 = 7.4;
    al2.n3 = 7.5;

    al3.matricula = 120;
    al3.n1 = 6.3;
    al3.n2 = 6.4;
    al3.n3 = 6.5;
```



Fila Sequencial Estática

- No arquivo `fila.h`, por se tratar de uma Fila estática, será necessário definir:
 - O tamanho máximo do *array* utilizado na Fila. Este tamanho, será representado pela constante MAX;
 - O tipo de dados opaco que representará a Fila, será definido pela criação de uma estrutura chamada `struct fila`, e por sua vez por questões de padronização, terá seu nome redefinido através de um comando `typedef` para `FILA`;
 - Como nas outras estruturas de dados já vistas, serão armazenadas dentro da Fila estruturas (structs) do tipo `ALUNO`, que conterão quatro campos: *matrícula*, *n1*, *n2* e *n3*.
- No arquivo `fila.c`, será definido:
 - O tipo de dado `struct fila` – que será o dado opaco;
 - As funções para manipular o tipo de dado `struct fila`, completando assim o encapsulamento.

No arquivo `fila.c` será definido tudo aquilo que deve ficar oculto do usuário/programador que se utilizar desta biblioteca, o tipo de dado opaco e as funções que foram definidas para manipular seus dados.



Estrutura de Dados 1

Fila Sequencial Estática

```
//arquivo fila.c - FILA ESTÁTICA
#include <stdio.h>
#include <stdlib.h>
#include "fila.h"
```

```
//definição do tipo FILA
struct fila{
    int inicio, fim, qtd;
    ALUNO dados[MAX];
};
```

Por estar definido dentro do arquivo `fila.c`, os campos dessa estrutura, não são visíveis pelo `main()`, apenas seu outro nome `FILA` que foi definido no arquivo `fila.h`. `main()` pode apenas declarar ponteiros para este tipo de dado, que deve ser inicializado com `NULL` para a detecção de alocação

```
//arquivo fila.h
#define MAX 100

typedef struct aluno{
    int matricula;
    float n1, n2, n3;
}ALUNO;
```

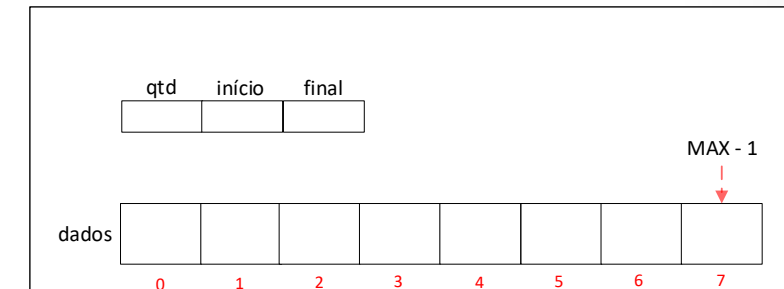
Número máximo de elementos suportados pela Fila

Estrutura `ALUNO`, visível pelo `main()` que será armazenada na Fila

```
typedef struct fila FILA;
```

Definição do tipo de dado `FILA`, que será visível pelo `main()`

```
//Arquivo main.c
//ponteiro para o no descritor
FILA *fi = NULL;
```



Estrutura de Dados 1

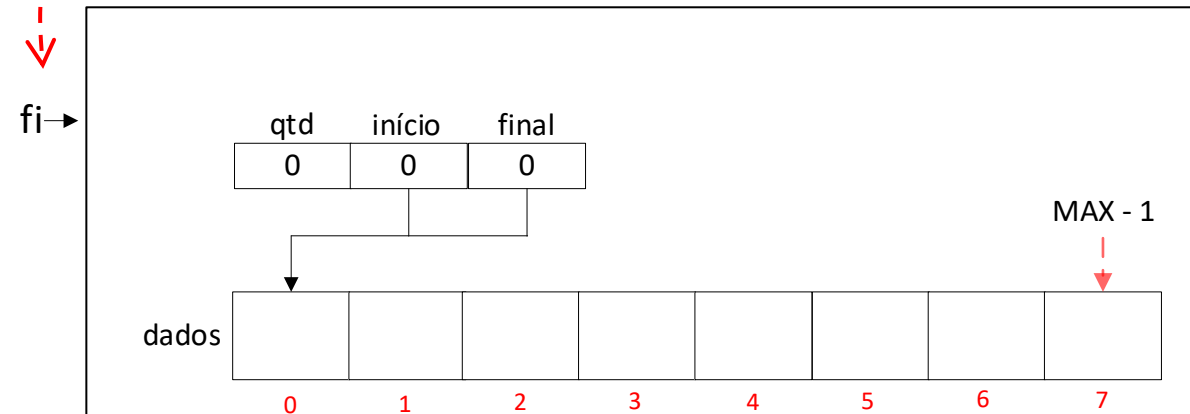
Fila Sequencial Estática

- Criando a Fila:

```
//arquivo fila.h
//cria a fila em memória e devolve
//endereço de sua localização
FILa *criaFila();
```

```
//arquivo fila.c - FILA ESTÁTICA
FILa *criaFila(){
    FILa *fi;
    fi = (FILa*) malloc(sizeof(FILa));
    if(fi != NULL){
        fi->inicio = 0;
        fi->fim = 0;
        fi->qtd = 0;
    }
}
```

```
//Arquivo main.c
fi = criaFila();
```



Estrutura de Dados 1

Fila Sequencial Estática

- Destruir uma Fila estática é bastante simples, basicamente o que tem que se fazer é liberar a memória alocada para toda a estrutura que representa a Fila. Isso pode ser feito com apenas uma chamada à função `free()`. Então por que criar uma função para destruir a fila, se podemos apenas chamar a dita função?
- Por questões de modularização. Destruir uma Fila estática é muito simples, mas destruir uma Fila alocada dinamicamente é uma tarefa mais complicada. Ao criar esta função, estamos escondendo a implementação dessa tarefa do usuário/programador, e ao mesmo tempo mantemos a mesma notação utilizada por uma Fila com alocação dinâmica.
- Dessa forma poderemos trocar os módulos (estático pelo dinâmico e vice e versa) entre si, e o funcionamento da Fila para o usuário/programador será sempre o mesmo, apesar terem as suas implementações completamente diferentes.

```
//arquivo fila.h
//libera memória alocada para a fila
void liberaFila(FILA *fi);
```

```
//arquivo fila.c - FILA ESTÁTICA
void liberaFila(FILA *fi){
    free(fi);
}
```

```
//Arquivo main.c
liberaFila(fi);
```



Estrutura de Dados 1

Fila Sequencial Estática

- Abortando o programa em caso de erro de alocação:

```
//arquivo fila.c - FILA ESTÁTICA
void abortaPrograma(){
    printf("ERRO! Fila nao foi alocada, ");
    printf("programa sera encerrado...\n\n\n");
    system("pause");
    exit(1);
}
```

- Em todas as funções que manipulam e acessam os dados encapsulados, deve-se testar antes da manipulação, se a Fila foi realmente alocada.
- Caso não tenha sido alocada por qualquer motivo, não teremos uma Fila válida para trabalhar. O programa então deverá ser abortado, antes porém, informando ao usuário sobre a falha na alocação, que portanto, impede o correto funcionamento do programa.



Estrutura de Dados 1

Fila Sequencial Estática



- Coletando informações básicas sobre a Fila:
- As operações de inserção, remoção e consulta são consideradas as principais operações de uma Fila.
- Apesar disso, para realizar estas operações, torna-se necessário ter algumas outras informações mais básicas sobre a Fila. Por exemplo, não é possível a remoção de um elemento da Fila se a mesma estiver vazia, ou se há uma tentativa de inserção em uma Fila que está cheia.
- Também útil, a informação de quantos elementos estão enfileirados nos dá uma ideia de quanto espaço ainda temos, antes que a Fila fique cheia impossibilitando a inserção de novos elementos.

Estrutura de Dados 1

Fila Sequencial Estática

- Verificando o tamanho da Fila:
- Basicamente, verificar o tamanho de uma Fila estática, consiste em acessar e retornar o valor de seu campo qtd, que armazena a quantidade de elementos que estão inseridos na Fila:

```
//Arquivo main.c
x = tamanhoFila(fi);
printf("\nO tamanho da Fila e: %d", x);
```

```
//arquivo fila.h
//devolve o quantos elementos estão na fila
int tamanhoFila(FILA *fi);
```

```
//arquivo fila.c - FILA ESTÁTICA
int tamanhoFila(FILA *fi){
    if(fi == NULL){
        abortaPrograma();
    }
    return fi->qtd;
}
```



Fila Sequencial Estática

- Verificando se a Fila está cheia:
- De forma similar, basicamente a verificação de Fila cheia, consiste em acessar o campo qtd e testar se seu valor é igual a constante MAX, que define o número máximo de elementos que a Fila suporta:

```
//arquivo fila.h
//verifica se fila cheia
int filaCheia(FILA *fi);
```

```
//Arquivo main.c
x = filaCheia(fi);
if(x){
    printf("\nA Fila esta cheia!");
}else{
    printf("\nA Fila nao esta cheia.");
}
```

```
//arquivo fila.c - FILA ESTÁTICA
int filaCheia(FILA *fi){
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->qtd == MAX){
        return 1;
    }else{
        return 0;
    }
}
```



Estrutura de Dados 1

Fila Sequencial Estática

- Verificando se a Fila está vazia:
- Também de forma similar, a verificação de Fila vazia, consiste em acessar o campo qtd e testar se seu valor é igual a 0 (zero), indicando então que não há elementos na Fila:

```
//arquivo fila.h
//verifica se fila vazia
int filaVazia(FILA *fi);
```

```
//Arquivo main.c
x = filaVazia(fi);
if(x){
    printf("\nA Fila esta vazia!");
}else{
    printf("\nA Fila nao esta vazia.");
}
```

```
//arquivo fila.c - FILA ESTÁTICA
int filaVazia(FILA *fi){
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->qtd == 0){
        return 1;
    }else{
        return 0;
    }
}
```



Estrutura de Dados 1

Fila Sequencial Estática

- Inserindo elementos na Fila:
- A inserção em uma Fila, ocorre sempre em seu final. Dessa forma, a estrutura ALUNO passada à função de inserção, é inserida na posição apontada pelo campo fim, incrementando-o, que então passa a indicar a próxima posição vaga disponível na Fila:

Utilizamos o resto da divisão no cálculo do novo final da Fila. Isso é feito para simular uma Fila circular. Dessa forma, ao chegar na posição MAX (que não existe no *array*), o final da fila será colocado na posição ZERO, de modo que as posições no começo do *array* que ficarem vagas, a medida que elementos são inseridos e removidos da fila, poderão também ser utilizadas.



```
//arquivo fila.h
//insere na fila, recebe estrutura a inserir
int insereFila(FILA *fi, ALUNO al);
```

```
//Arquivo main.c
x = insereFila(fi, al);
if(x) {
    printf("\nElemento %d inserido com sucesso!", x);
} else {
    printf("\nErro, elemento nao inserido.");
}
```

```
//arquivo fila.c - FILA ESTÁTICA
int insereFila(FILA *fi, ALUNO al) {
    if(fi == NULL) {
        abortaPrograma();
    }
    if(fi->qtd == MAX) {
        return 0;
    }
    fi->dados[fi->fim] = al;
    fi->fim = (fi->fim + 1) % MAX;
    fi->qtd++;
    return al.matricula;
}
```



Fila Sequencial Estática

- Removendo elementos da Fila:
- A remoção em uma Fila, ocorre sempre em seu início. Dessa forma, basta incrementar em uma unidade o campo `inicio`, tornando a posição que ele apontava anteriormente, disponível para novos elementos.

Utilizamos o resto da divisão no cálculo do novo início da Fila. Isso é feito para simular uma Fila circular. Dessa forma, ao chegar na posição MAX (que não existe no *array*), o início da fila será colocado na posição ZERO, de modo que as posições no começo do *array* que ficarem vagas, a medida que elementos são inseridos e removidos da fila, poderão também ser utilizadas.

```
//arquivo fila.h
//remove um elemento da fila
int removeFila(FILA *fi);
```

```
//Arquivo main.c
x = removeFila(fi);
if(x){
    printf("\nElemento %d removido com sucesso!", x);
}else{
    printf("\nErro, elemento nao removido.");
}
```

```
//arquivo fila.c - FILA ESTÁTICA
int removeFila(FILA *fi){
    int matricula;
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->qtd == 0){
        return 0;
    }
    matricula = fi->dados[fi->inicio].matricula;
    fi->inicio = (fi->inicio + 1) % MAX;
    fi->qtd--;
    return matricula;
}
```


Estrutura de Dados 1

Fila Sequencial Estática

- Consultando elementos da Fila:
- Nas Listas (sequencial de dinâmica) que vimos anteriormente, podemos acessar qualquer um de seus elementos. Porém, em uma Fila, somente pode ser acessado o elemento que está em seu início:

```
//arquivo fila.h
//consulta elem na fila, recebe end. estrutura
int consultaFila(FILA *fi, ALUNO *al);
```

```
//Arquivo main.c
x = consultaFila(fi, &al_consulta);
if(x){
    printf("\nConsulta realizada com sucesso:");
    printf("\nMatricula: %d", al_consulta.matricula);
    printf("\nNota 1:      %.2f", al_consulta.n1);
    printf("\nNota 2:      %.2f", al_consulta.n2);
    printf("\nNota 3:      %.2f", al_consulta.n3);
}else{
    printf("\nErro, consulta nao realizada.");
}
```

```
//arquivo fila.c - FILA ESTÁTICA
int consultaFila(FILA *fi, ALUNO *al){
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->qtd == 0){
        return 0;
    }
    *al = fi->dados[fi->inicio];
    return 1;
}
```

```
Consulta realizada com sucesso:
Matricula: 100
Nota 1:      8.30
Nota 2:      8.40
Nota 3:      8.50
```



Estrutura de Dados 1

Fila Sequencial Estática



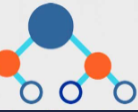
- Atividade 1 – Fila estática:
- Monte o programa Fila estática, posicionando as chamadas das funções no `main()`, para manipulação da Fila, de forma que apresentem o funcionamento e as mensagens iguais ao exemplo ao lado. Entregue no Moodle todos os arquivos zipados.

"C:\Users\angelot\Documents\Aulas\EDA\

```
O tamanho da Fila e: 0
A Fila nao esta cheia.
A Fila esta vazia!
Elemento 100 inserido com sucesso!
Elemento 110 inserido com sucesso!
Elemento 120 inserido com sucesso!
O tamanho da Fila e: 3
Consulta realizada com sucesso:
Matricula: 100
Nota 1:      8.30
Nota 2:      8.40
Nota 3:      8.50
Elemento 100 removido com sucesso!
Consulta realizada com sucesso:
Matricula: 110
Nota 1:      7.30
Nota 2:      7.40
Nota 3:      7.50
```

Fila Sequencial Dinâmica

- Alocação Dinâmica:
 - O espaço de memória é alocado em tempo de execução, ou seja, a fila cresce a medida que novos elementos são armazenados, e diminui a medida que elementos são removidos.
 - Nesse tipo de implementação, cada elemento pode estar em uma área distinta da memória, que pode ou não ser consecutiva. Por esse motivo é absolutamente necessário que cada elemento da Fila armazene, além de sua informação, o endereço de memória onde se encontra o próximo elemento.
 - Assim, o acesso aos seus elementos é encadeado, onde cada elemento pode estar em uma área distinta da memória. Para se acessar um elemento, é preciso percorrer todos os seus antecessores. Porém, como é uma Fila, só temos acesso ao seu primeiro elemento, e isso não chega a ser um problema.



Estrutura de Dados 1

Fila Sequencial Dinâmica

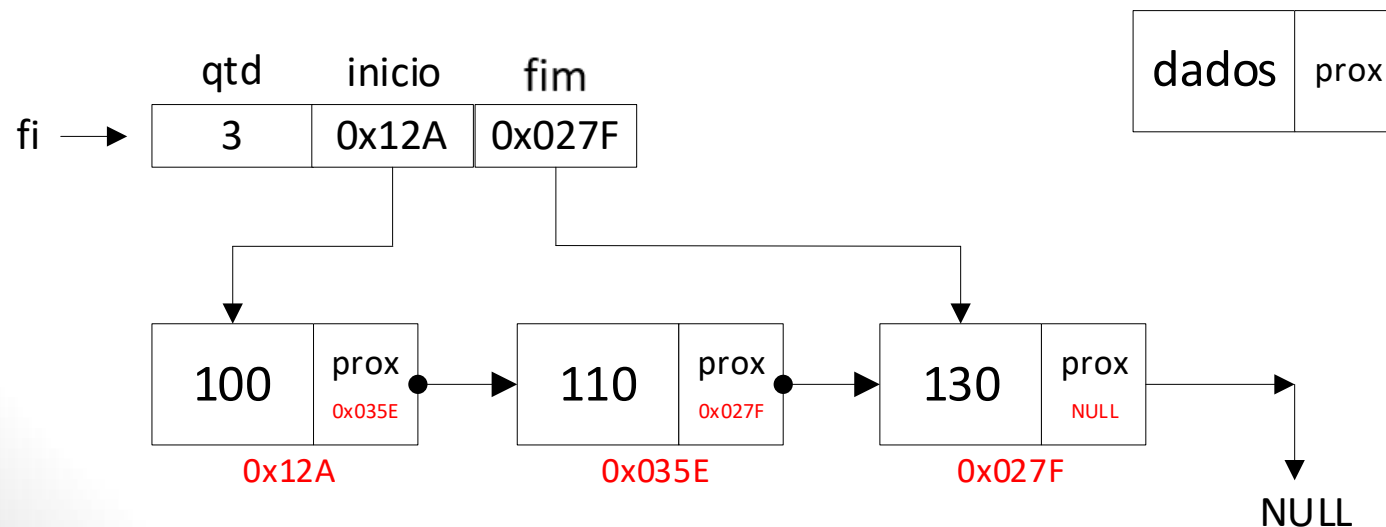
- O módulo dinâmico deste programa, foi construído para ser compatível função a função com a Fila estática, bastando para isso, substituir o módulo `fila.c` (estático) pelo módulo que veremos agora, o `fila.c` (dinâmico), trocando um arquivo pelo outro.
- Observe que desse modo, a implementação em módulos impede que o usuário/programador que utilizar esta biblioteca, tenha conhecimento do código interno e qual tipo de Fila está usando (estática ou dinâmica), uma vez que em uma distribuição real, os módulos `fila.c` são distribuídos pré-compilados, não permitindo o acesso e leitura de seu código.
- Essa é a grande vantagem da modularização e da utilização dos tipos opacos e encapsulamento: mudar a maneira como a Fila foi implementada não altera nem interfere no funcionamento do programa que a utiliza.



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Implementação:
- Em uma Fila Dinâmica cada elemento aponta para o seu sucessor na Fila;
- Este tipo de Fila também usa um nó “descritor” para representar a quantidade de elementos, o início e o final da fila, além de uma indicação especial de final de Fila:



A principal vantagem de se utilizar uma abordagem dinâmica e encadeada na definição da Fila é a melhor utilização dos recursos de memória, retirando a necessidade de definir previamente o tamanho de Fila. Já a sua principal desvantagem é a necessidade de percorrer toda a fila para destruí-la.



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Utilizaremos os arquivos `main()` e `fila.h`, do projeto anterior, e teremos no arquivo `fila.h`, por se tratar de uma Fila dinâmica, as definições:
 - O tipo de dados `opaco` que representará a Fila, definido pela criação de uma estrutura chamada `struct fila`, que será composta por três campos: `qtd`, `inicio` e `final`. Por questões de padronização com o projeto anterior, terá seu nome redefinido através de um comando `typedef` para `FILA`;
 - Como nas outras estruturas de dados já vistas, serão armazenadas dentro da Fila, estruturas do tipo `ALUNO`, que conterão quatro campos: `matrícula`, `n1`, `n2` e `n3`.
- No arquivo `fila.c`, será definido:
 - O tipo de dado `struct fila`, que será o dado `opaco`;
 - O tipo de dado `struct elemento`, que somente será visível dentro do módulo `fila.c`, uma vez que não é exportado pelo arquivo `fila.h`. Sua função será armazenar os dados e encadear a Fila, e a fim de facilitar a codificação, terá seu nome modificado para `ELEM`, por meio do comando `typedef`.
 - As funções para manipulação dos tipo de dados `struct fila` e `struct elemento`, completando assim o encapsulamento.



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Implementação:

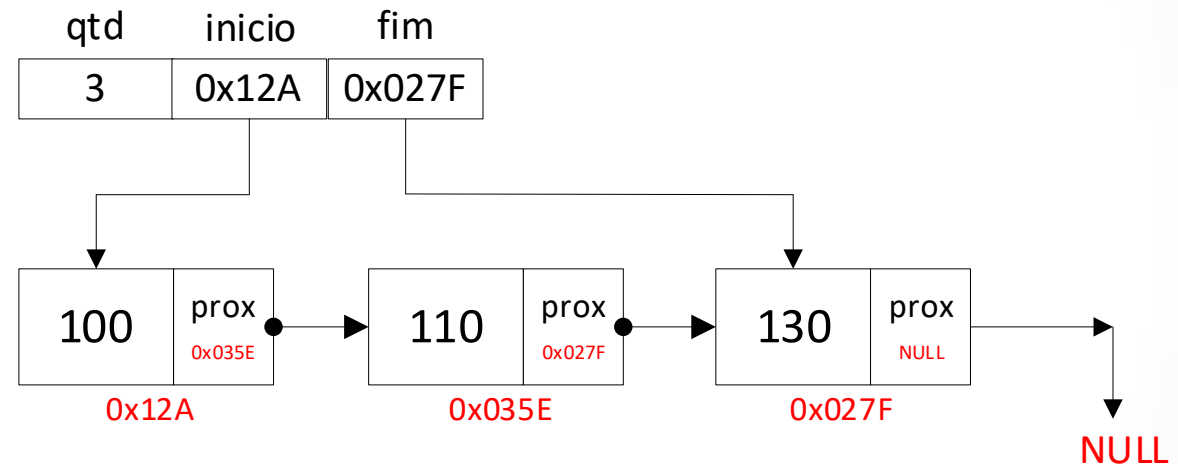
```
//Arquivo fila.c - FILA DINÂMICA
#include <stdio.h>
#include <stdlib.h>
#include "fila.h"
```

```
//definição do nó descritor
```

```
struct fila{
    struct elemento *inicio;
    struct elemento *fim;
    int qtd;
};
```

```
struct elemento{
    ALUNO dados;
    struct elemento *prox;
};
```

```
typedef struct elemento Elem;
```



Apenas para não
digitar muito a todo
instante...



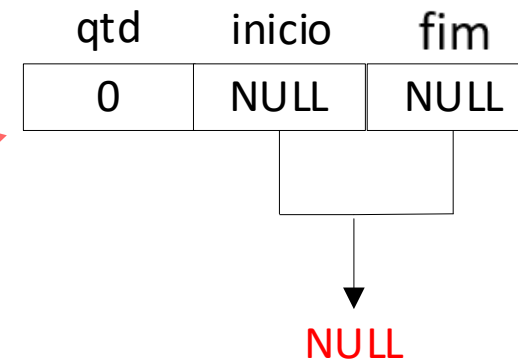
Estrutura de Dados 1

Fila Sequencial Dinâmica

- Criando a Fila:

```
//Arquivo fila.c - FILA DINÂMICA
```

```
FILA *criaFila(){  
    FILA *fi = (FILA*) malloc(sizeof(FILA));  
    if(fi != NULL){  
        fi->fim = NULL;  
        fi->inicio = NULL;  
        fi->qtd = 0;  
    }  
    return fi;  
}
```



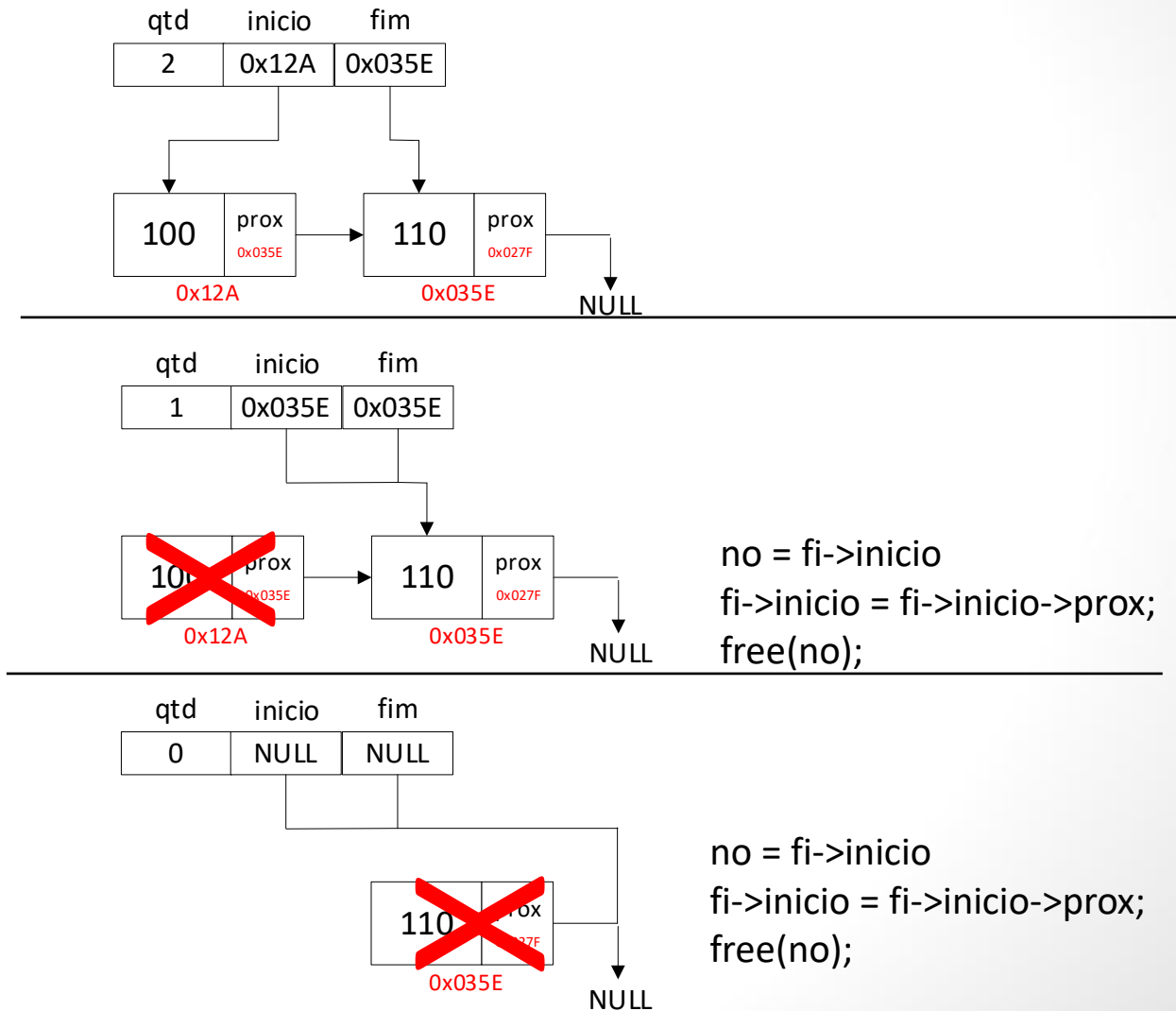
Estrutura de Dados 1

Fila Sequencial Dinâmica

- Destruindo a Fila

//Arquivo fila.c - FILA DINÂMICA

```
void liberaFila(FILA *fi){  
    if(fi != NULL){  
        Elem *no;  
        while(fi->inicio != NULL){  
            no = fi->inicio;  
            fi->inicio = fi->inicio->prox;  
            free(no);  
        }  
        free(fi);  
    }  
}
```



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Abortando o programa em caso de erro de alocação:

```
//arquivo fila.c - FILA ESTÁTICA
void abortaPrograma() {
    printf("ERRO! Fila nao foi alocada, ");
    printf("programa sera encerrado...\n\n\n");
    system("pause");
    exit(1);
}
```

- Também nesta versão da Fila, em todas as funções que manipulam e acessam os dados encapsulados, deve-se testar antes da manipulação, se a Fila foi realmente alocada.
- Caso não tenha sido alocada por qualquer motivo, não teremos uma Fila válida para trabalhar. O programa então deverá ser abortado, antes porém, informando ao usuário sobre a falha na alocação, que portanto, impede o correto funcionamento do programa.

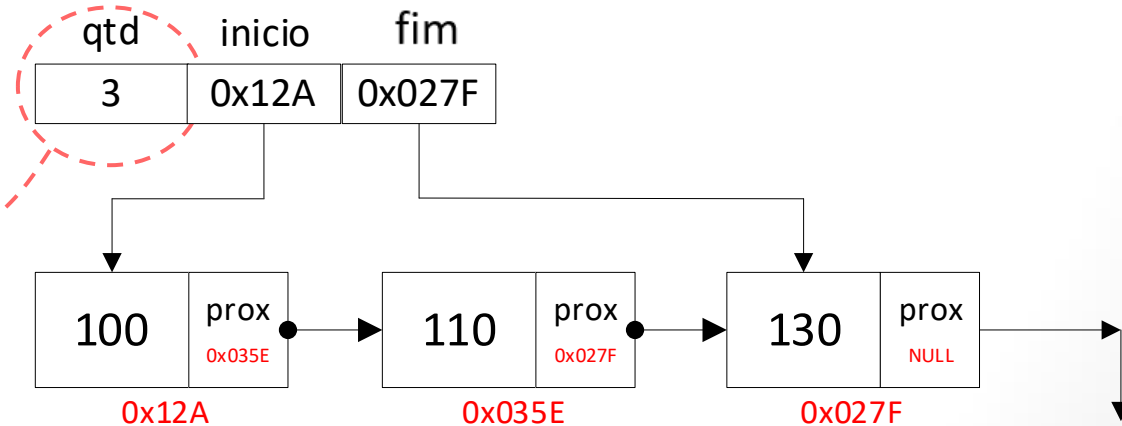


Estrutura de Dados 1

Fila Sequencial Dinâmica

- Informações básicas sobre a Fila, retornando o tamanho:

```
//Arquivo fila.c - FILA DINÂMICA
int tamanhoFila(FILA *fi){
    if(fi == NULL){
        abortaPrograma();
    }
    return fi->qtd;
}
```



Estrutura de Dados 1

Fila Sequencial Dinâmica



- Informações básicas sobre a Fila, retornando se cheia:

```
//Arquivo fila.c - FILA DINÂMICA
int filaCheia(FILA *fi){
    return 0;
}
```

- Em estruturas alocadas dinamicamente não faz sentido o conceito de “estruturas cheias”. Mantêm-se a função por uma questão de padronização e compatibilidade com as Filas estáticas.

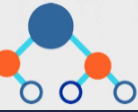
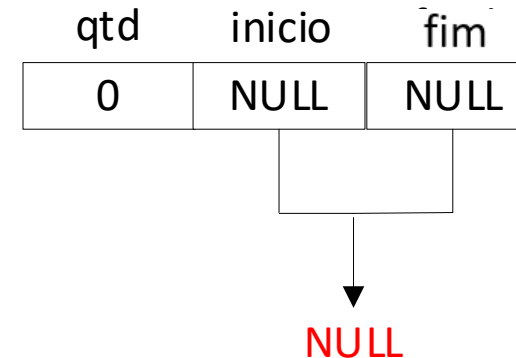
Estrutura de Dados 1

Fila Sequencial Dinâmica

- Informações básicas sobre a Fila, retornando se vazia:

```
//Arquivo fila.c - FILA DINÂMICA
int filaVazia(FILA *fi){
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->inicio == NULL){
        return 1;
    }
    return 0;
}
```

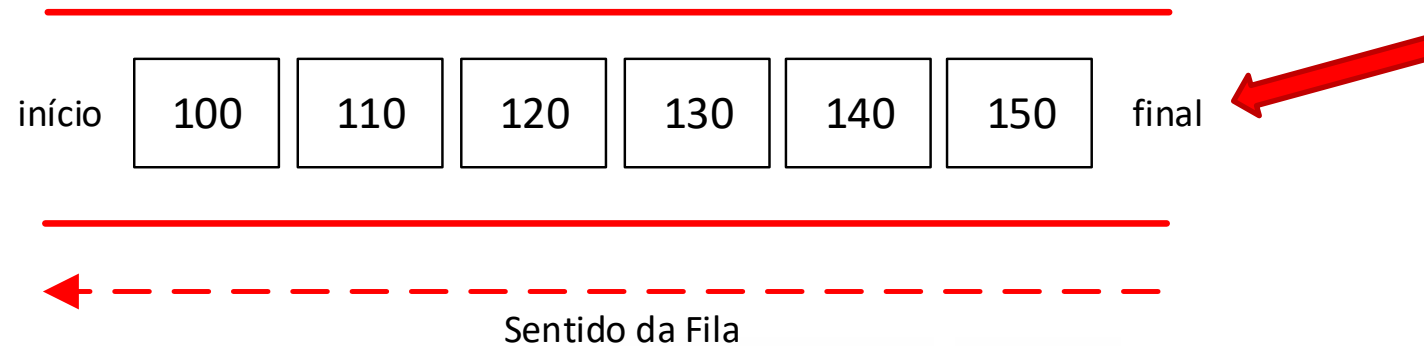
Não é necessário
verificar o final da lista



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Em uma Fila dinâmica a inserção também é sempre no seu final;



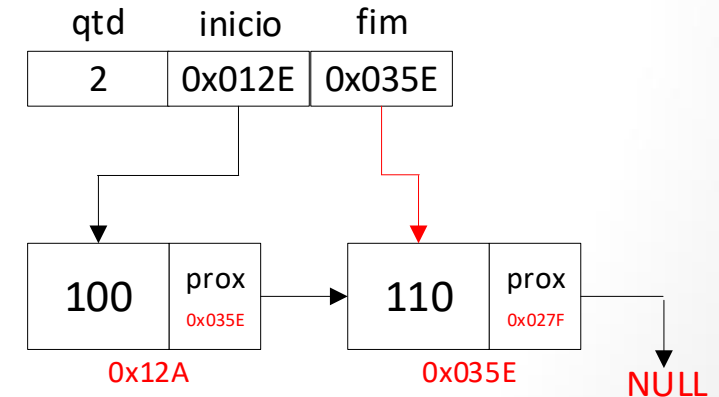
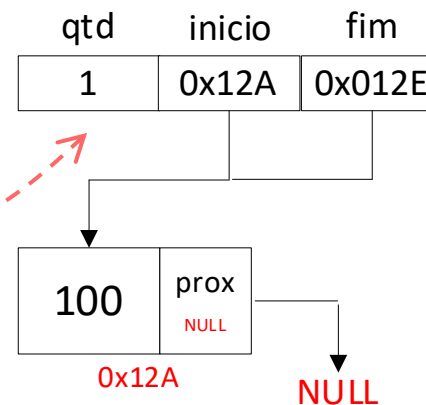
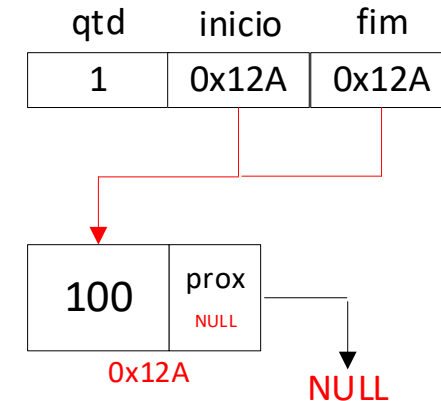
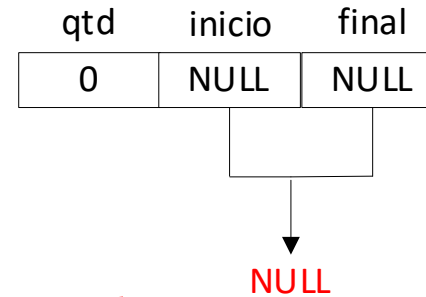
Estrutura de Dados 1

Fila Sequencial Dinâmica



//Arquivo fila.c - FILA DINÂMICA

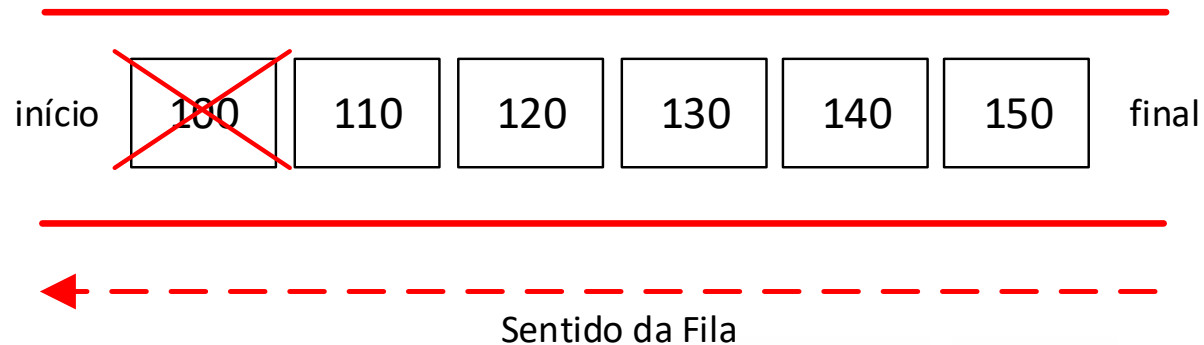
```
int insereFila(FILA *fi, ALUNO al){
    if(fi == NULL){
        abortaPrograma();
    }
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){
        return 0;
    }
    no->dados = al;
    no->prox = NULL;
    if(fi->fim == NULL){ //fila vazia
        fi->inicio = no;
    } else { //insere no final da fila
        fi->fim->prox = no;
    }
    fi->fim = no; //no passa a ser novo final da fila
    fi->qtd++;
    return al.matricula;
}
```



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Assim como na Fila estática, em uma Fila dinâmica a remoção é em seu início, sempre tendo o cuidado de não se tentar remover elementos de uma Fila vazia:



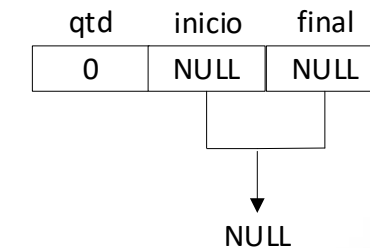
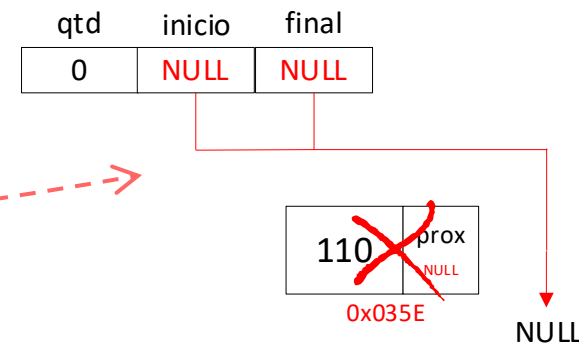
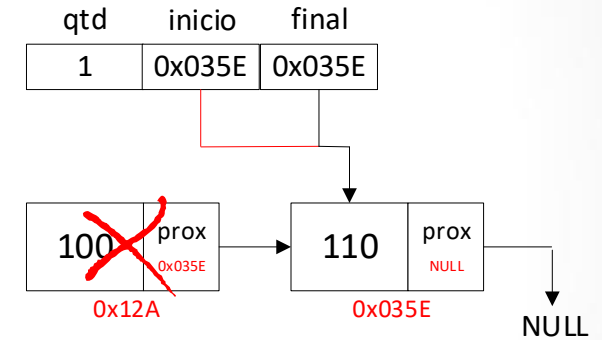
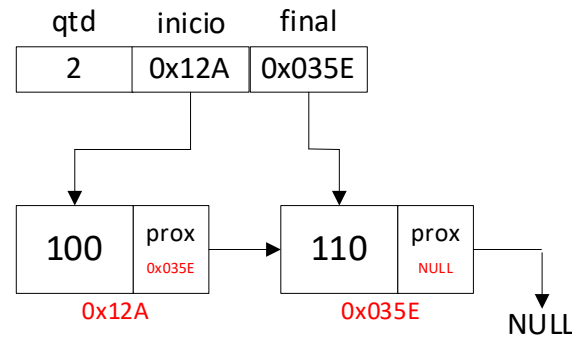
Estrutura de Dados 1

Fila Sequencial Dinâmica



//Arquivo fila.c - FILA DINÂMICA

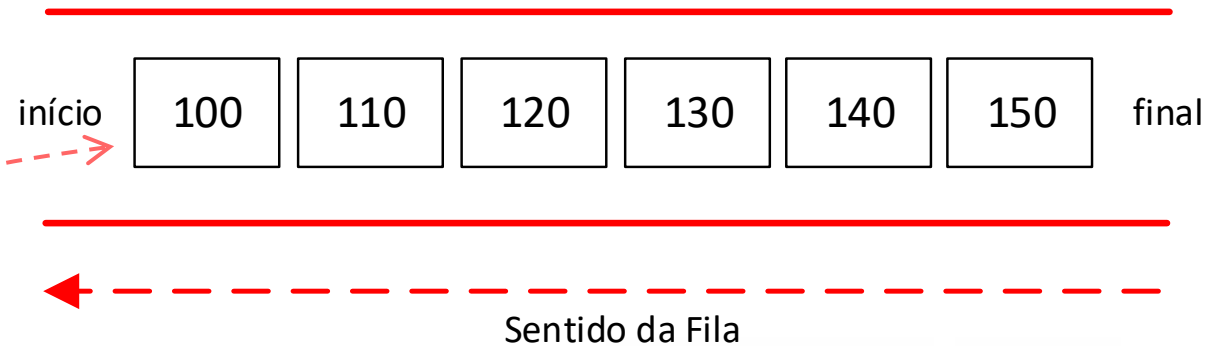
```
int removeFila(FILA *fi){
    int matricula;
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->inicio == NULL){ //fila vazia
        return 0;
    }
    matricula = fi->inicio->dados.matricula;
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL){ //fila ficou vazia
        fi->fim = NULL;
    }
    free(no);
    fi->qtd--;
    return matricula;
}
```



Estrutura de Dados 1

Fila Sequencial Dinâmica

- Em uma Fila a consulta se dá apenas ao elemento que está no seu início:



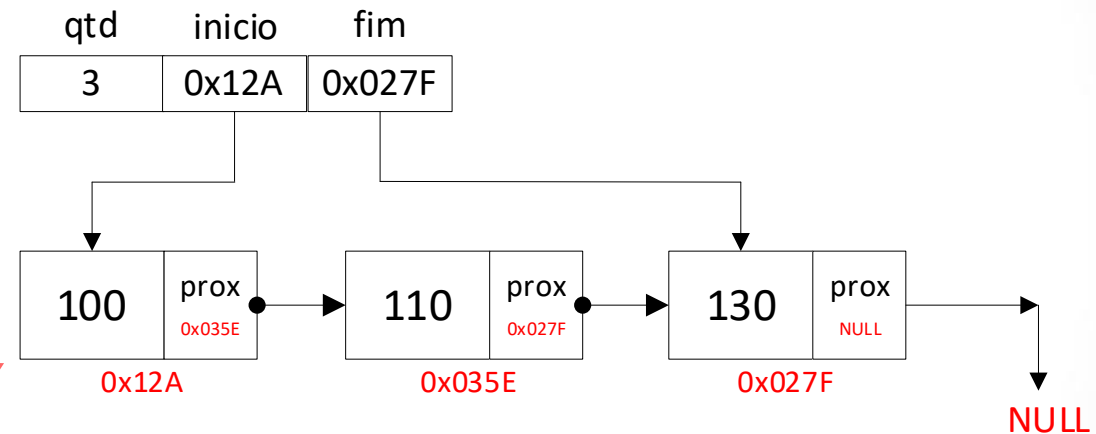
Só se acessa para consulta este elemento, senão, não seria uma Fila... Seria uma Lista.



Estrutura de Dados 1

Fila Sequencial Dinâmica

```
//Arquivo fila.c - FILA DINÂMICA
int consultaFila(FILA *fi, ALUNO *al){
    if(fi == NULL){
        abortaPrograma();
    }
    if(fi->inicio == NULL){ //fila vazia
        return 0;
    }
    *al = fi->inicio->dados;
    return 1;
}
```



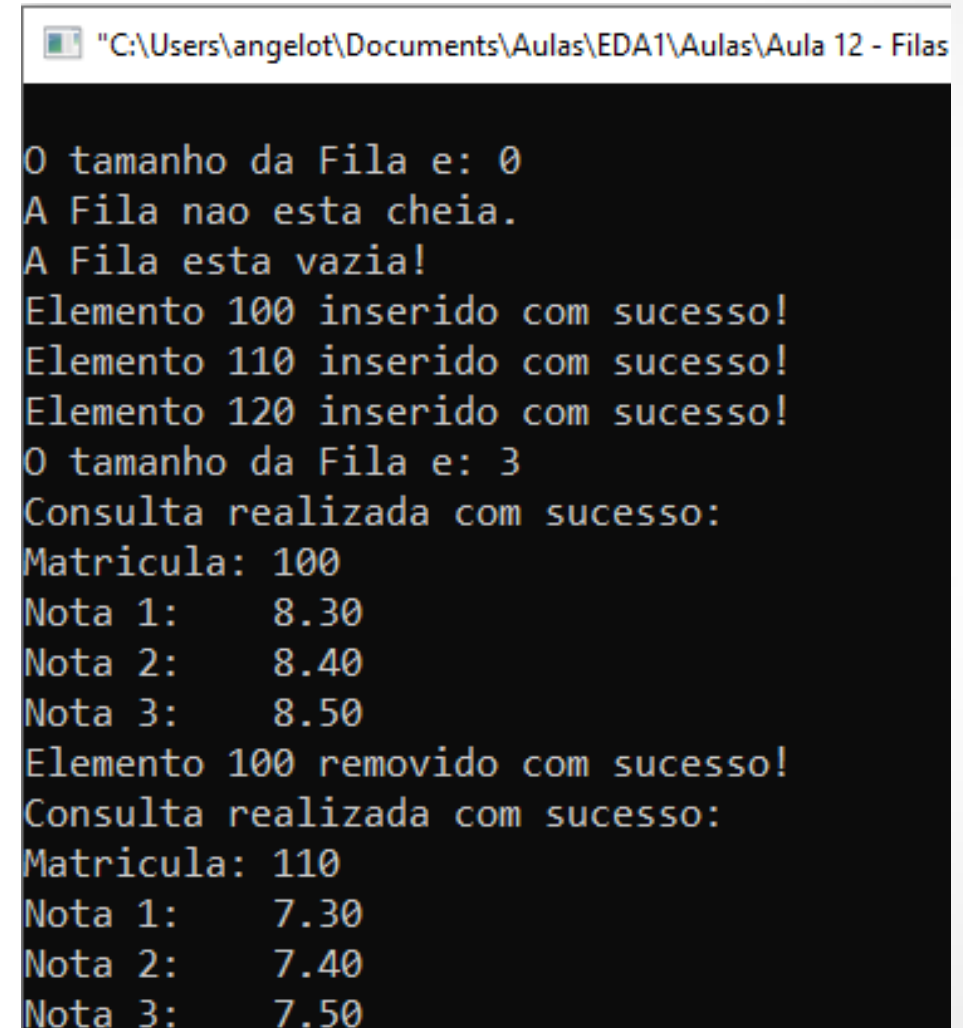
```
Consulta realizada com sucesso:
Matricula: 100
Nota 1:    8.30
Nota 2:    8.40
Nota 3:    8.50
```



Estrutura de Dados 1

Atividade 2

- Monte o programa Fila dinâmica, posicionando as chamadas das funções no `main()`, para manipulação da Pilha, de forma que apresentem o funcionamento e as mensagens iguais ao exemplo ao lado. Entregue no Moodle todos os arquivos zipados.



```
"C:\Users\angelot\Documents\Aulas\EDA1\Aulas\Aula 12 - Filas"

O tamanho da Fila e: 0
A Fila nao esta cheia.
A Fila esta vazia!
Elemento 100 inserido com sucesso!
Elemento 110 inserido com sucesso!
Elemento 120 inserido com sucesso!
O tamanho da Fila e: 3
Consulta realizada com sucesso:
Matricula: 100
Nota 1:      8.30
Nota 2:      8.40
Nota 3:      8.50
Elemento 100 removido com sucesso!
Consulta realizada com sucesso:
Matricula: 110
Nota 1:      7.30
Nota 2:      7.40
Nota 3:      7.50
```

