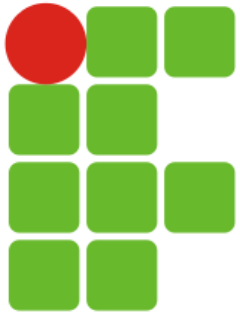


Estrutura de Dados 1

Aula 13 – Fila Dinâmica e Pilha Dinâmica

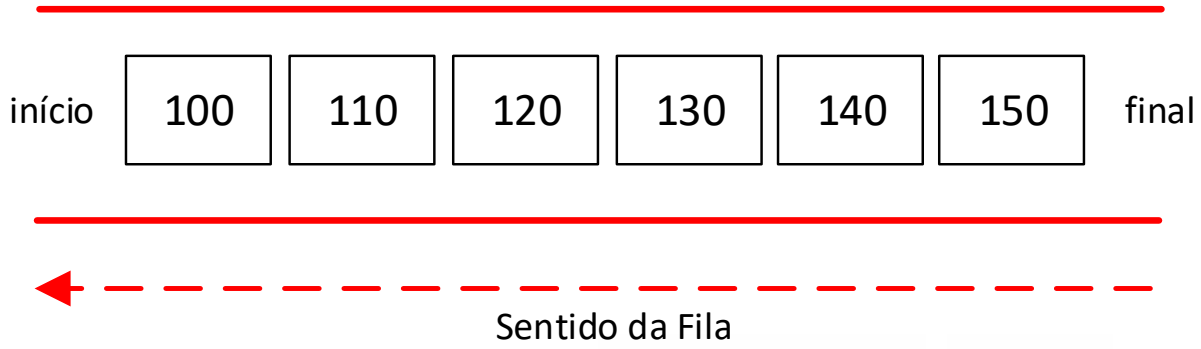
Antonio Angelo de Souza Tartaglia

angelot@ifsp.edu.br

Estrutura de Dados 1

Fila

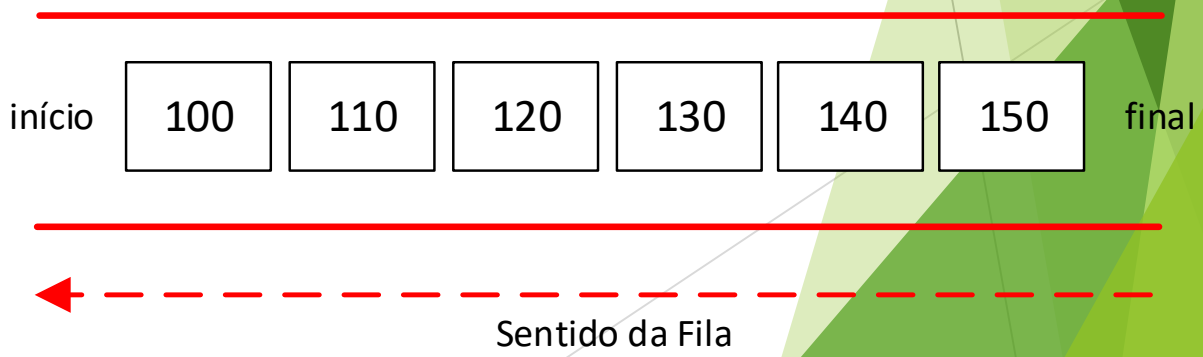
- Uma estrutura do tipo “Fila” é uma sequência de elementos do mesmo tipo, como as “Listas”. Seus elementos possuem estrutura interna abstraída, ou seja, sua complexidade é arbitrária e não afeta o seu funcionamento. São estruturas do tipo “FIFO” (first-in-first-out).



Estrutura de Dados 1

Fila

- Uma fila é um tipo especial de “Lista”:
  - Inserções e exclusões de elementos ocorrem nas extremidades.
- Aplicações
  - Qualquer aplicação onde se necessite de:
    - Controle de fluxo;
    - Recursos compartilhados (impressoras, transações de bancos de dados, etc);



Estrutura de Dados 1

Fila

- Em uma fila 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;
  - Etc.

- Essas operações dependem do tipo de alocação de memória utilizada:
  - Alocação estática;
  - Alocação Dinâmica.

Estrutura de Dados 1

Fila

- Alocação Estática:
  - O Espaço de memória é alocado no momento da compilação;
  - Exige a definição do número máximo de elementos da “Fila”;
  - Acesso sequencial: Elementos estão de forma consecutiva na memória.

Estrutura de Dados 1

Fila

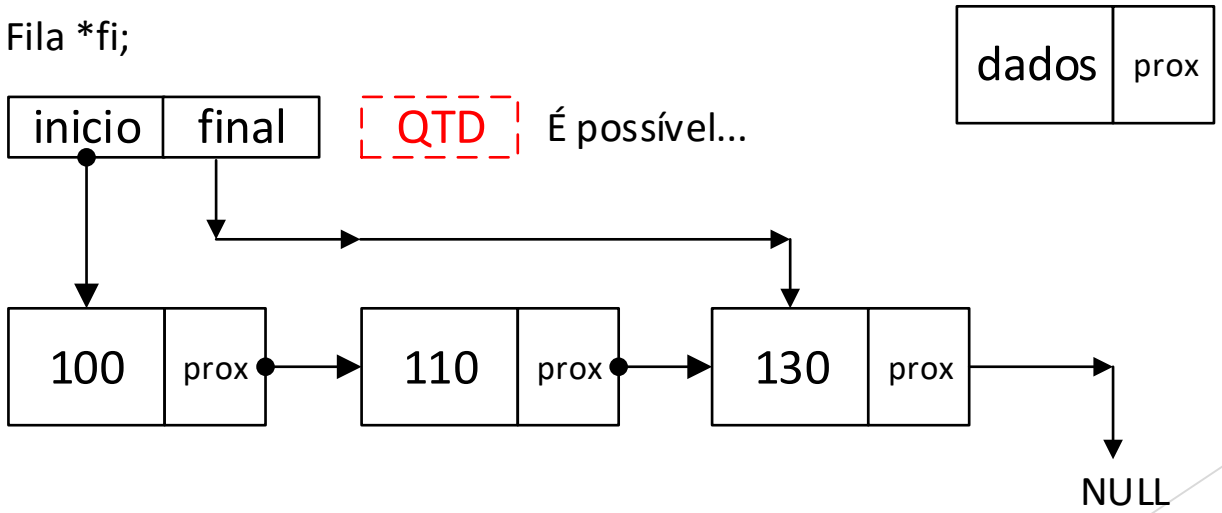
- Alocação Dinâmica:
  - O espaço de memória é alocado em tempo de execução;
  - A fila cresce a medida que novos elementos são armazenados, e diminui a medida que elementos são removidos;
  - Acesso encadeado: Cada elemento pode estar em uma área distinta da memória. Para acessar um elemento, é preciso percorrer todos os seus antecessores na “Fila”.

Estrutura de Dados 1

Fila Dinâmica - Implementação

- Em uma Fila Dinâmica cada elemento aponta para o seu sucessor na Fila;

- Este tipo de Fila usa um nó “descritor” para representar o início e o final da fila, e uma indicação especial de final de Fila:



Estrutura de Dados 1

Fila Dinâmica - Implementação

- **filaD.h**
  - Os protótipos das funções;
  - O tipo de dado armazenado na Fila;
  - O ponteiro Fila.
- **filaD.c**
  - O tipo de dados “Fila”;
  - Implementar suas funções.

Estrutura de Dados 1

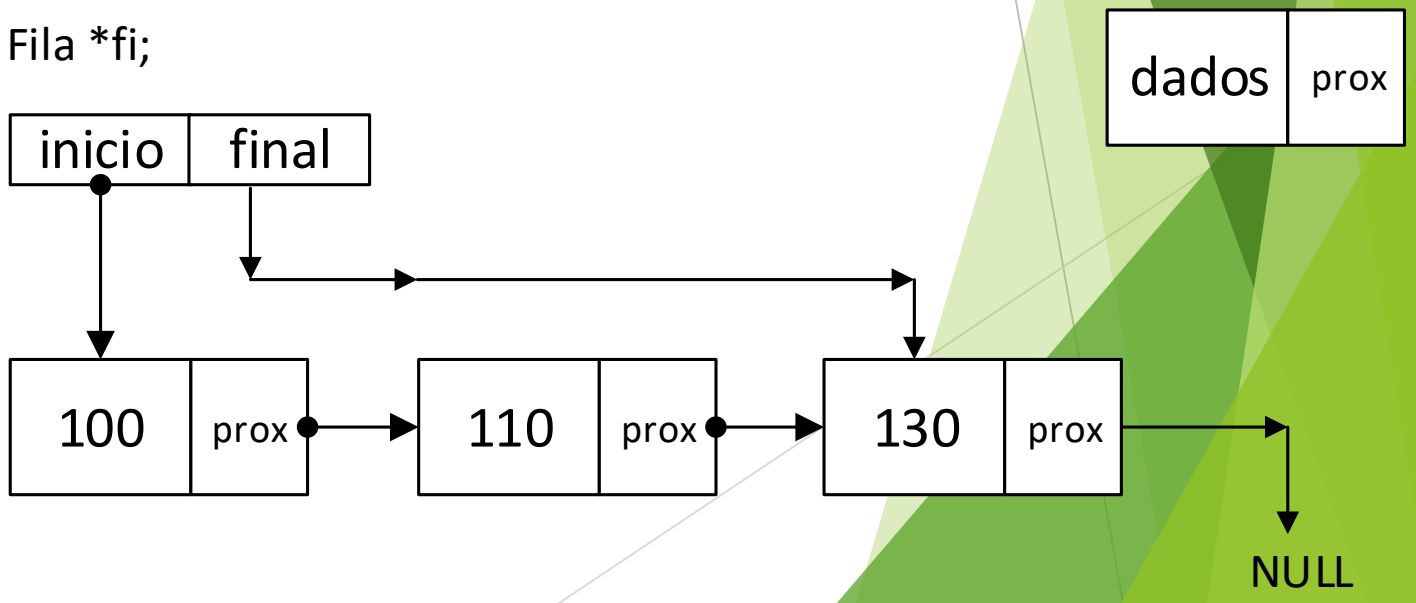
Fila Dinâmica - Implementação

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

struct fila{
    struct elemento *inicio;
    struct elemento *fim;
};

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

typedef struct elemento Elem;
```

Apenas para não digitar muito a todo instante...

```c
//Arquivo principal main()
Fila *fi; //ponteiro para o no descritor
```

```c
//Arquivo filaD.h
typedef struct aluno{
    int matricula;
    float n1, n2, n3;
}ALUNO;

typedef struct fila Fila;
```



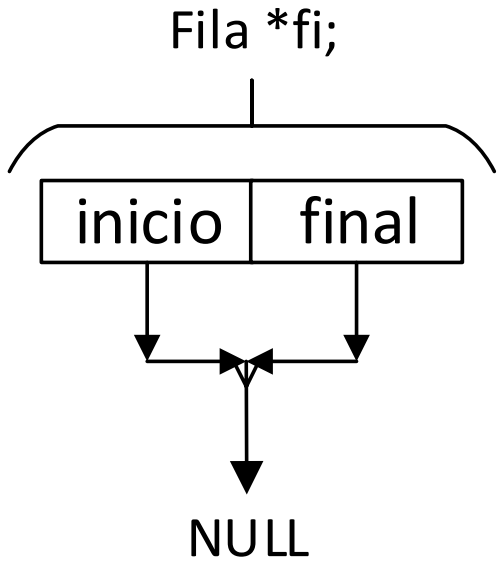
Estrutura de Dados 1

Fila Dinâmica - Criando a Fila

```c
//Arquivo filaD.h
Fila *cria_fila();
```

```c
//Arquivo filaD.c
Fila *cria_fila(){
    Fila *fi = (Fila*) malloc(sizeof(Fila));
    if(fi != NULL){
        fi->fim = NULL;
        fi->inicio = NULL;
    }
    return fi;
}
```

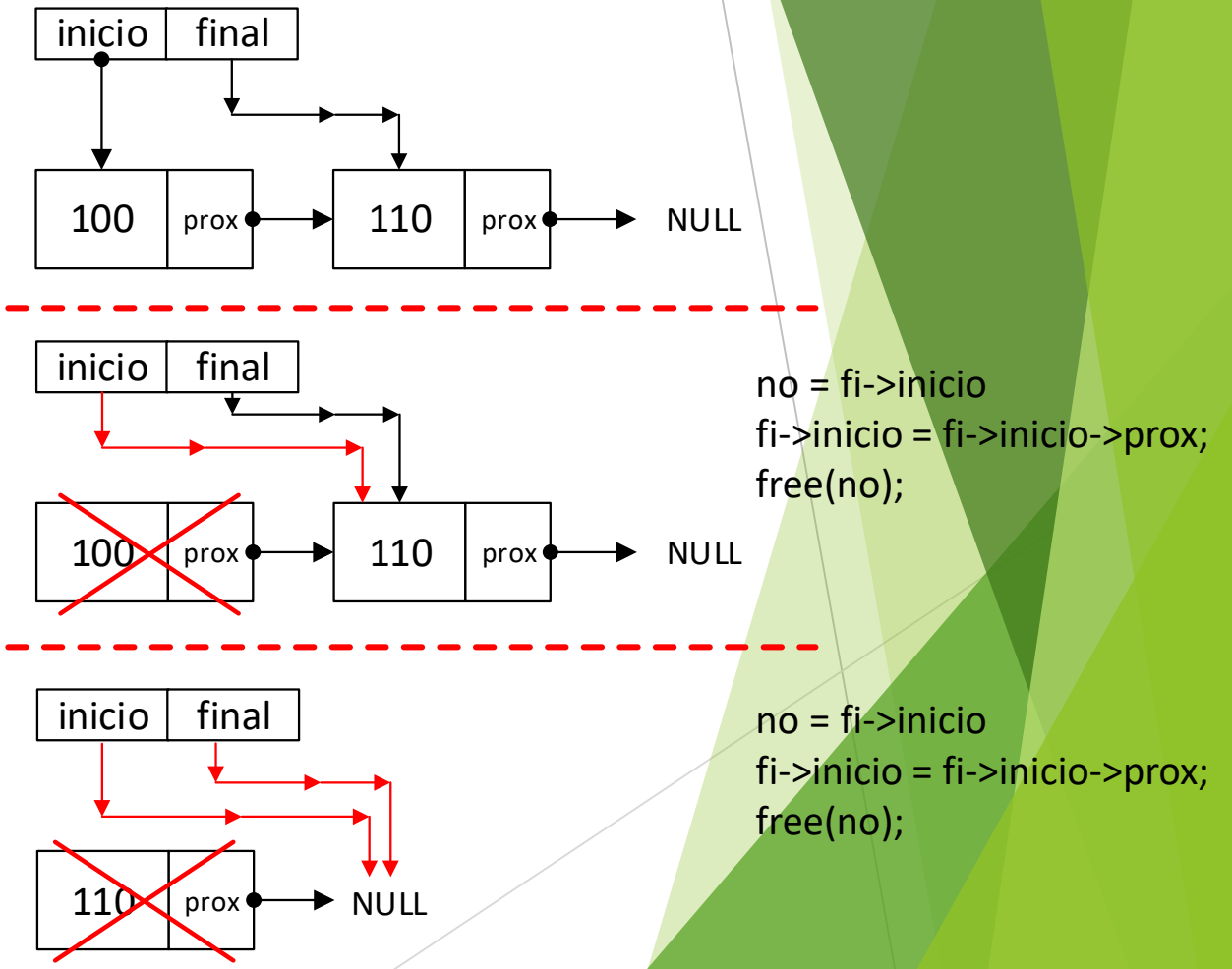
```c
//Arquivo principal main()
fi= cria_fila();
```



Estrutura de Dados 1

Fila Dinâmica - Destruindo a Fila

```
//Arquivo filaD.h
void destroi_fila(fi);

 //Arquivo filaD.c
void destroi_fila(Fila *fi){
    if(fi != NULL){ //testa se fila foi alocada
        Elem *no;
        while(fi->inicio != NULL){
            no = fi->inicio;
            fi->inicio = fi->inicio->prox;
            free(no);
        }
        free(fi);
    }
}

//Arquivo principal main()
destroi_fila(fi);
```



Estrutura de Dados 1

Fila Dinâmica Fila

- Informações básicas sobre a Fila:
  - Tamanho;
  - Se está cheia;
  - Se está vazia.

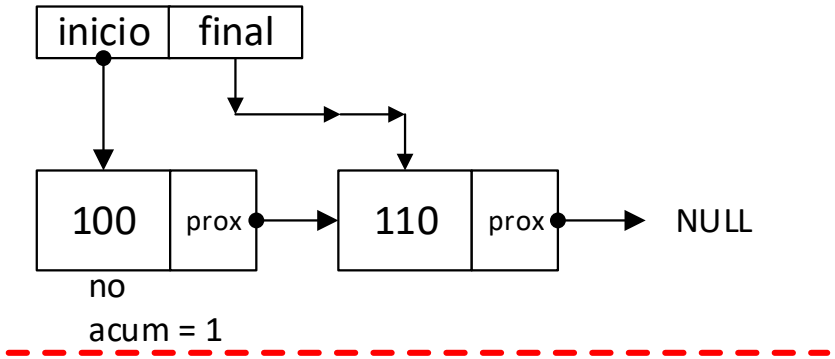
```c
//Arquivo filaD.h
int tamanho_fila(Fila *fi);
```

```c
//Arquivo principal main()
x = tamanho_fila(fi);
printf("\nO tamanho da fila e: %d", x);
```

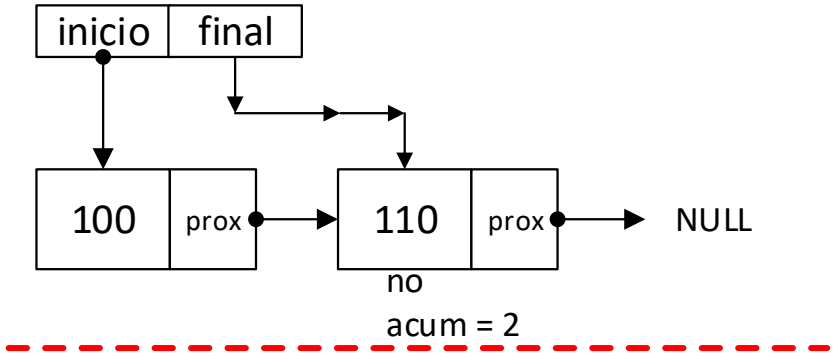
```c
//Arquivo filaD.c
int tamanho_fila(Fila *fi){
    if(fi == NULL){
        return 0;
    }
    int acum = 0;
    Elem *no = fi->inicio;
    while(no != NULL){
        acum++;
        no = no->prox;
    }
    return acum;
}
```

Estrutura de Dados 1

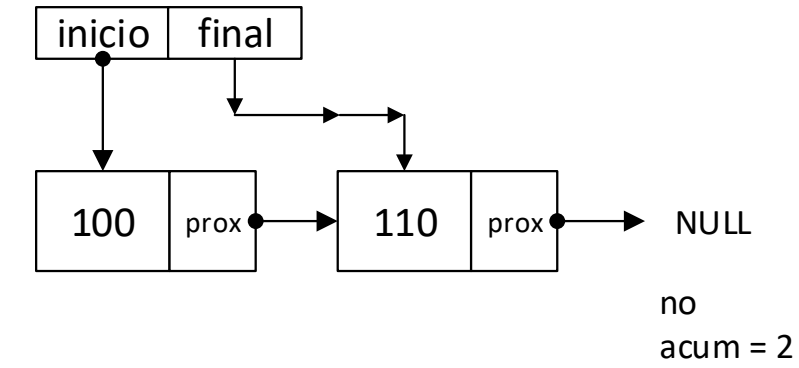
Fila Dinâmica - Tamanho



acum++;
no = no->prox;



acum++;
no = no->prox;



Fim:
No = NULL

Estrutura de Dados 1

Fila Dinâmica - Fila Cheia

- Novamente, em estruturas alocadas dinamicamente não faz sentido o conceito de “estruturas cheias”. Mantêm-se a função por uma questão de padronização.

```c
//Arquivo filaD.h
int fila_cheia(Fila *fi);
```

```c
//Arquivo filaD.c
int fila_cheia(Fila *fi){
    return 0;
}
```

```c
//Arquivo principal main()
x = fila_cheia(fi);
if(x){
    printf("\nA fila está cheia!");
}else{
    printf("\nA fila não está cheia.");
}
```

Estrutura de Dados 1

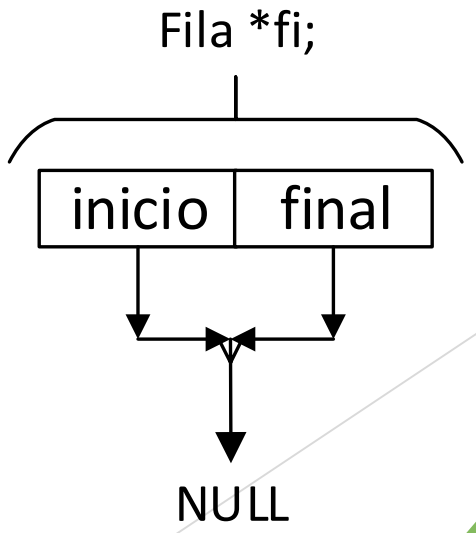
Fila Dinâmica - Fila Vazia

```c
//Arquivo filaD.h
int fila_vazia(Fila *fi);
```

```c
//Arquivo principal main()
x = fila_vazia(fi);
if(x){
    printf("\nA fila está vazia!");
}else{
    printf("\nA fila não está vazia.");
}
```

```c
//Arquivo filaD.c
int fila_vazia(Fila *fi){
    if(fi == NULL){
        return 1;
    }
    if(fi->inicio == NULL){
        return 1;
    }
    return 0;
}
```

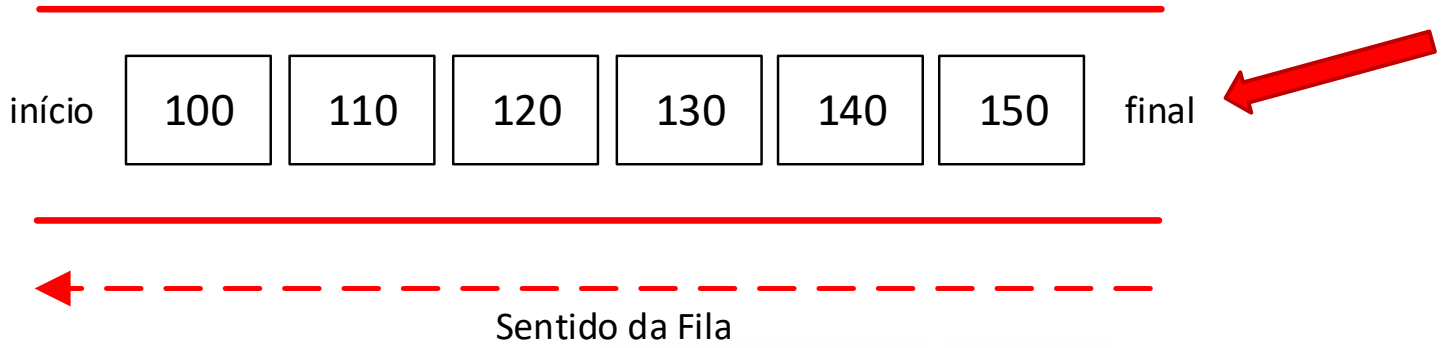
Não é necessário olhar para o final da lista



Estrutura de Dados 1

Fila Dinâmica - Inserção

- Em uma Fila a inserção é sempre no seu final;
- Também existe o caso onde a inserção é feita em uma fila que está vazia:



Estrutura de Dados 1

Fila Dinâmica - Inserção

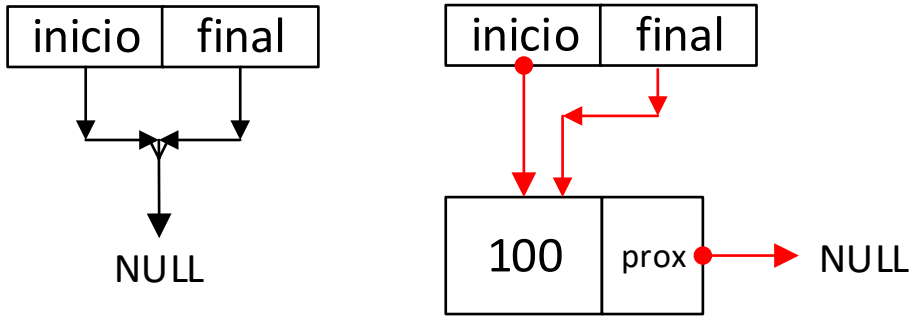
```c
//Arquivo filaD.c
int insere_fila(Fila *fi, ALUNO al){
    if(fi == NULL){//Testa se fila existe
        return 0;
    }
    //aloca memória para um novo elemento da fila
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL){//testa se alocação bem sucedida
        return 0;
    }
    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
    return 1;
}
```

```c
//Arquivo filaD.h
int insere_fila(Fila *fi, ALUNO al);
```

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

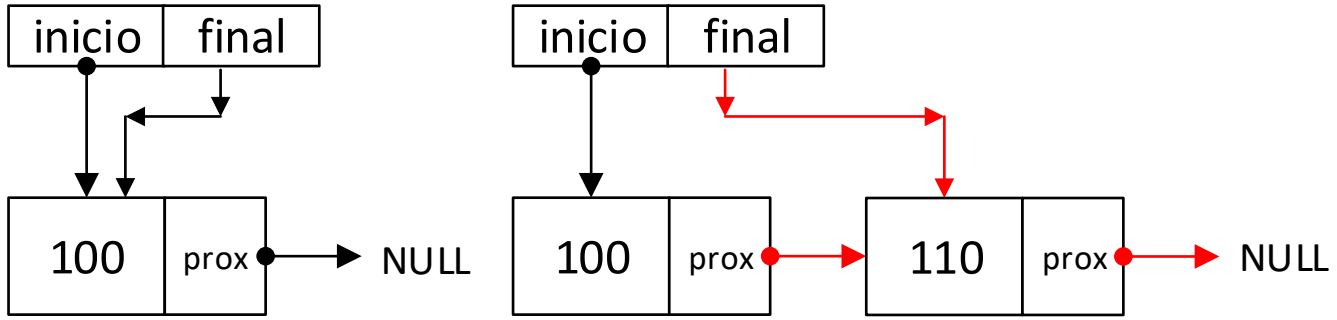
Estrutura de Dados 1

Fila Dinâmica - Inserção



Inserção em Fila vazia:

```
no->prox = NULL;
fi->inicio = no;
fi->final = no;
```



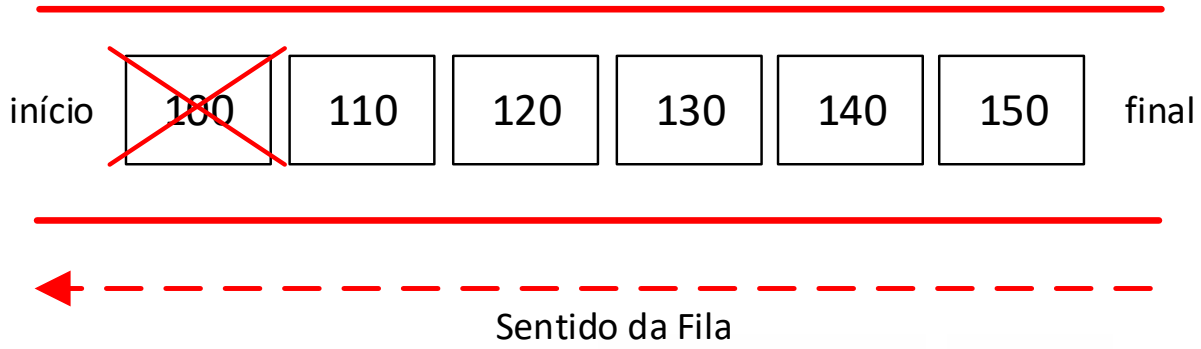
Inserção em Fila não vazia:

```
no->prox = NULL;
fi->inicio->prox = no;
fi->final = no;
```

Estrutura de Dados 1

Fila Dinâmica - Remoção

- Em uma Fila a remoção é sempre no seu início.
- Cuidado: não se pode remover de uma Fila vazia!



Estrutura de Dados 1

Fila Dinâmica - Remoção

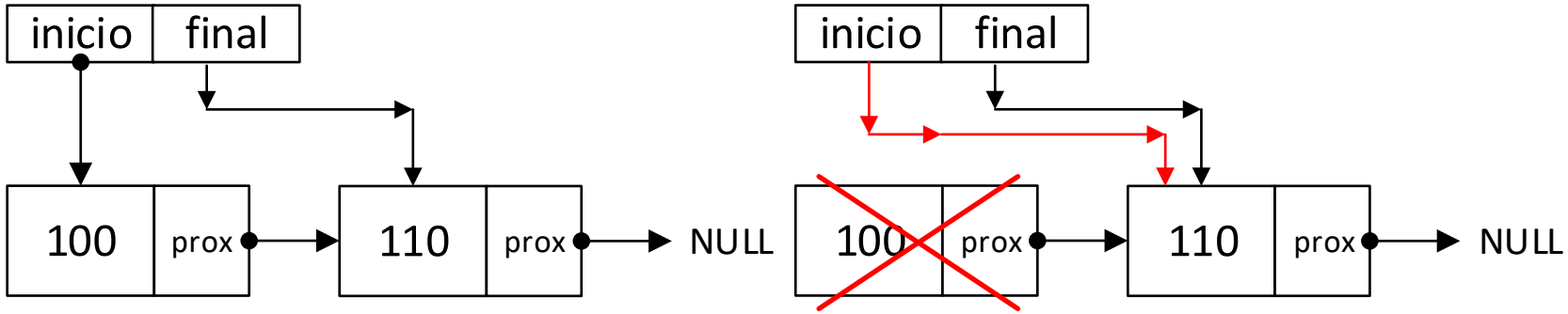
```c
//Arquivo filaD.h
int remove_fila(Fila *fi);
```

```c
//Arquivo filaD.c
int remove_fila(Fila *fi){
    if(fi == NULL){
        return 0;
    }
    if(fi->inicio == NULL){ //fila vazia
        return 0;
    }
    Elem *no = fi->inicio;
    fi->inicio = fi->inicio->prox;
    if(fi->inicio == NULL){  //fila ficou vazia
        fi->fim = NULL;
    }
    free(no);
    return 1;
}
```

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

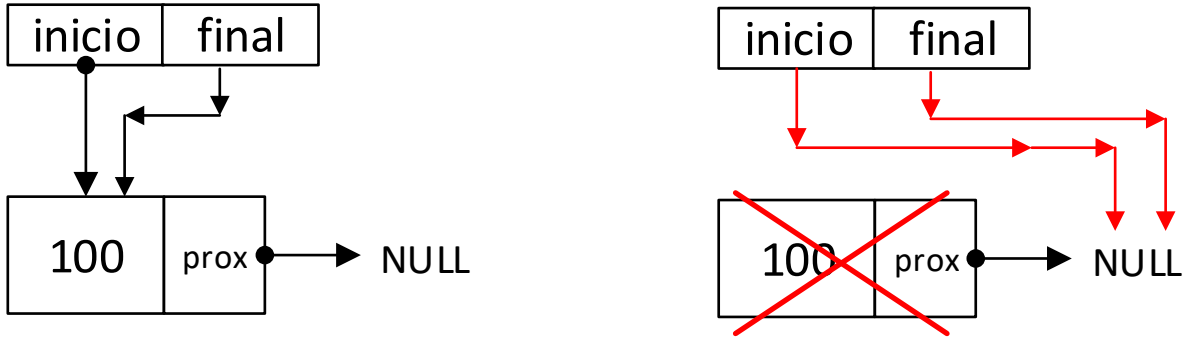
Estrutura de Dados 1

Fila Dinâmica - Remoção



Remoção em Fila não vazia:

```
*no = fi->inicio;
fi->inicio = fi->inicio->prox;
free(no);
```



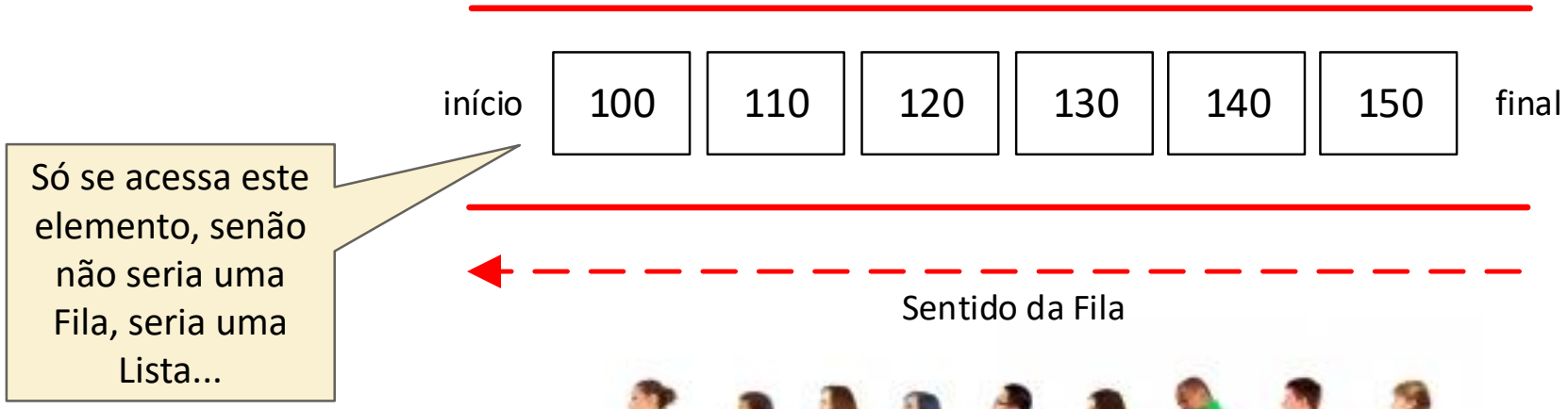
Fila fica vazia:

```
*no = fi->inicio;
fi->inicio = fi->inicio->prox;
fi->final = NULL;
free(no);
```

Estrutura de Dados 1

Fila Dinâmica - Consulta

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



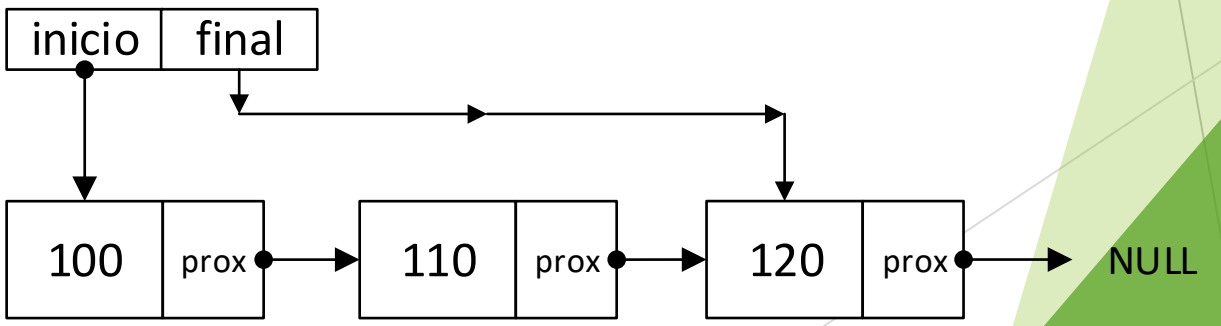
Estrutura de Dados 1

Fila Dinâmica - Consulta

```c
//Arquivo filaD.h
int consulta_fila(Fila *fi, ALUNO *al);
```

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

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



*al = fi->inicio->dados;

Estrutura de Dados 1

Atividade 1

- Entregue os arquivos no Moodle como “atividade 2 - Filha”

Estrutura de Dados 1

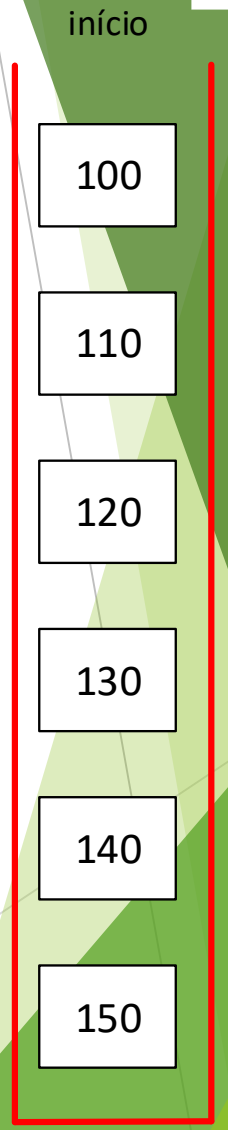
Pilha - Definição

- Sequência de elementos do mesmo tipo, como as “Listas” e “Filas”.

- São estruturas de dados do tipo LIFO (last-in-first-out), onde o último elemento a ser inserido, será o primeiro a ser retirado. Assim, uma pilha permite acesso a apenas um item de dados - o último inserido. Para processar o penúltimo item inserido, deve-se remover o último.

- Seus elementos possuem estrutura interna abstraída, ou seja, sua complexidade é arbitrária e não afeta o seu funcionamento.

Um elemento sobre o outro e o acesso sempre pelo topo da pilha



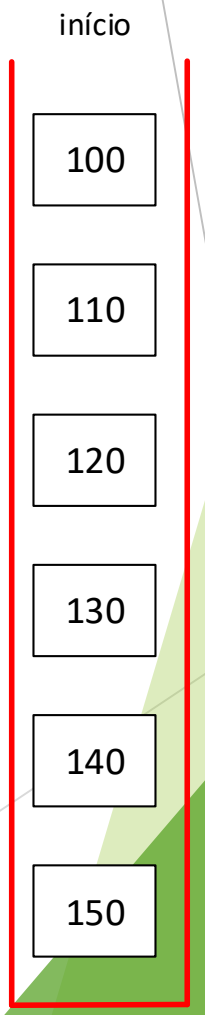
Estrutura de Dados 1

Pilha - Definição

- Aplicações:
  - Análise de uma expressão matemática;
  - Avaliação de uma expressão pós-fixa;
  - Converter expressão infixa para pós-fixa;
  - Converter números decimais para binário;
  - etc;.

Exemplo:
Infixa 2 + 3
Pós-fixa 23+
Mais informações, leia o documento disponível no Moodle

Não é possível remover um elemento no meio da pilha, sem desmachá-la



Estrutura de Dados 1

Pilha - Definição

- Em uma Pilha podemos realizar as seguintes operações:
  - Criação da Pilha;
  - Inserção de um elemento no início da Pilha;
  - Exclusão de um elemento no início da Pilha;
  - Acesso ao elemento do início da Pilha;
  - Destruição da Pilha;
  - Etc;
- Essas operações dependem do tipo de alocação de memória utilizada:
  - Estática;
  - Dinâmica.

Estrutura de Dados 1

Pilha - Alocação

- Alocação Estática:
  - O espaço de memória é alocado no momento da compilação (definição do Array – ou vetor);
  - Exige definição do número máximo de elementos que a Pilha suportará;
  - Acesso sequencial: elementos consecutivos na memória.
- Alocação Dinâmica:
  - O espaço de memória é alocado em tempo de execução;
  - A Pilha cresce a medida que novos elementos são armazenados, e diminui quando são removidos;
  - Acesso encadeado: cada elemento pode estar em uma área distinta da memória;
  - Para acessar um elemento, é preciso percorrer todos os seus antecessores na Pilha.

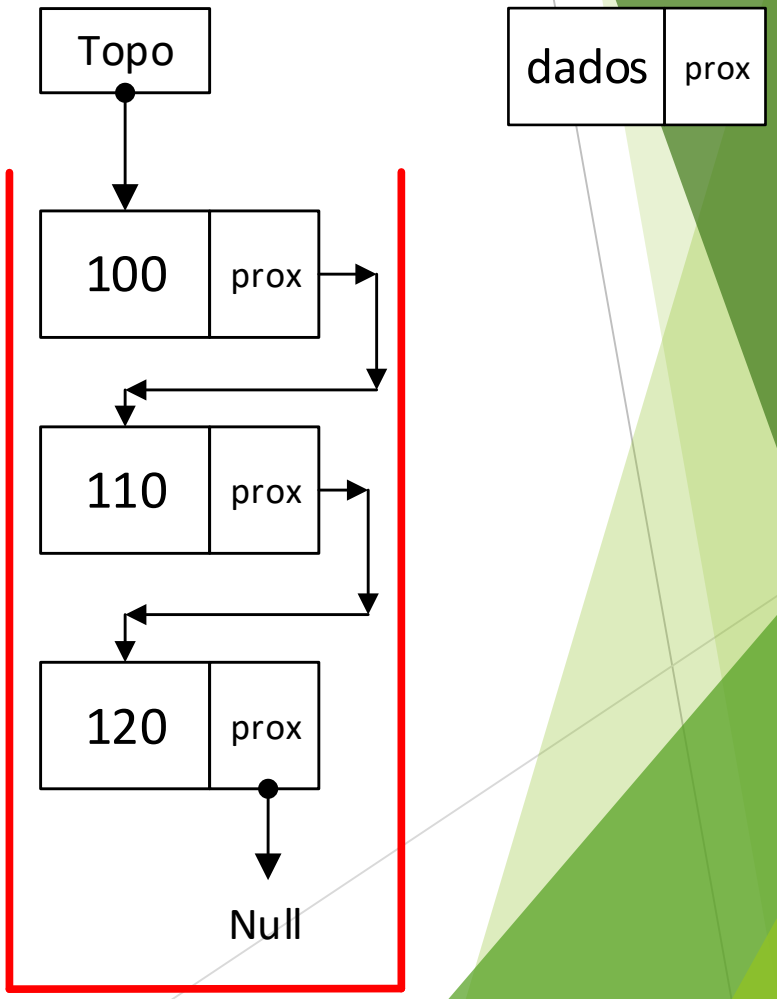
Como em uma Pilha acessamos sempre o seu topo, isso não é muito preocupante...

Estrutura de Dados 1

Pilha Dinâmica implementação

- Tipo de Pilha onde cada elemento aponta para o seu sucessor na Pilha.

- Utiliza um ponteiro especial (ponteiro para ponteiro), para o primeiro elemento da Pilha, e uma indicação de final de Pilha.



Estrutura de Dados 1

Pilha Dinâmica - Implementação

- pilhaD.h
  - Protótipos das funções;
  - Tipo de dado armazenado na Pilha;
  - O ponteiro Pilha.

- pilhaD.c
  - Tipo de dados Pilha;
  - Implementação de suas funções.

Estrutura de Dados 1

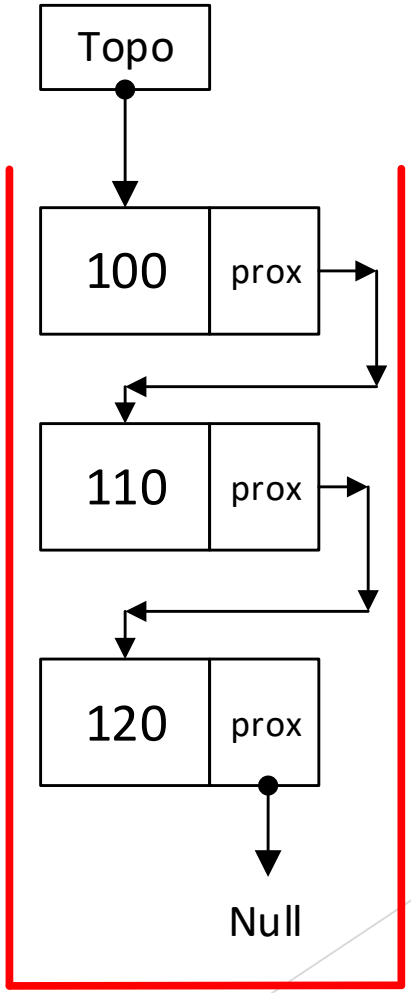
Pilha Dinâmica - Implementação

```c
//Arquivo pilhaD.h
typedef struct aluno{
    int matricula;
    float n1, n2, n3;
}ALUNO;

typedef struct elemento *Pilha;
```

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

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

typedef struct elemento Elem;
```

Apenas para não digitar muito a todo instante...

```c
//Arquivo principal main()
Pilha *pi;
```



Estrutura de Dados 1

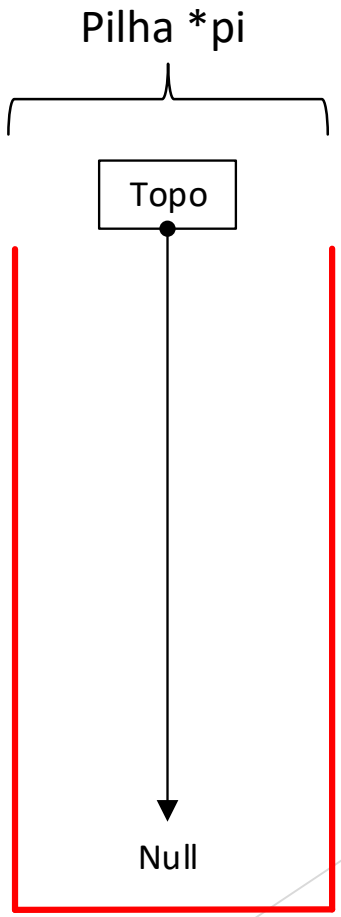
Pilha Dinâmica – Criar Pilha

```
//Arquivo pilhaD.h
Pilha cria_pilha();
```

```
//Arquivo pilhaD.c
Pilha cria_pilha(){
    Pilha *pi = (Pilha*) malloc(sizeof(Pilha));
    if(pi != NULL){
        *pi = NULL;
    }
    return pi;
}
```

Simplesmente cria o topo da Pilha

```
//Arquivo principal main()
pi = cria_pilha();
```



Estrutura de Dados 1

Pilha Dinâmica – Destruir Pilha

```c
//Arquivo pilhaD.h
void destroi_pilha(Pilha *pi);
```

```c
//Arquivo pilhaD.c
void destroi_pilha(Pilha *pi){
    if(pi != NULL){ //verifica se Pilha existe
        Elem *no;
        while ((*pi) != NULL){
            no = *pi;
            *pi = (*pi)->prox;
            free(no);
        }
        free(pi);
    }
}
```

Recebe o 1º elemento da Pilha.

O topo recebe o próximo elemento da Pilha.

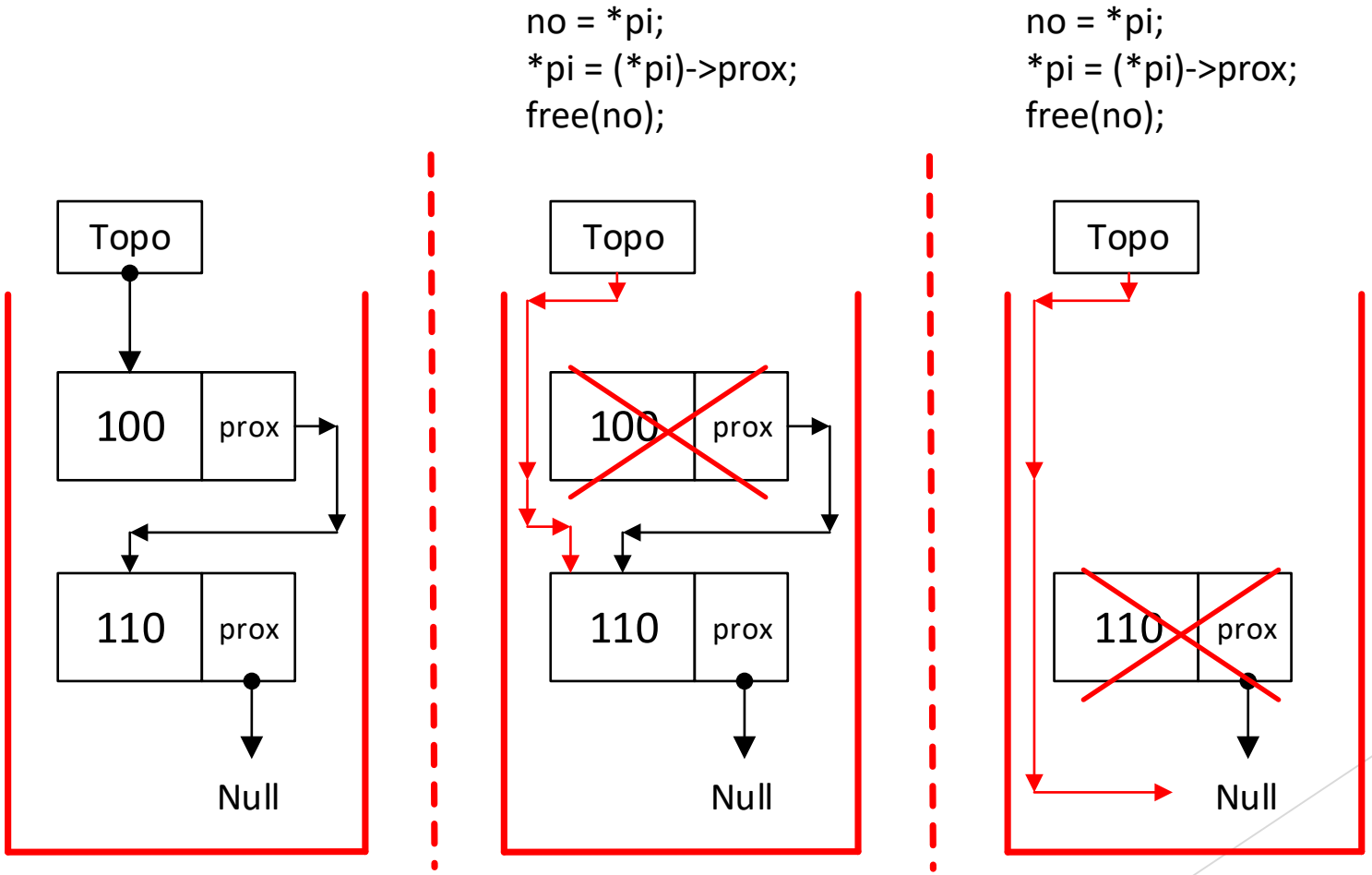
Libera o antigo topo da Pilha.

Remove o topo.

```c
//Arquivo principal main()
destroi_pilha(pi);
```

Estrutura de Dados 1

Pilha Dinâmica – Destruir Pilha



Estrutura de Dados 1

Pilha Dinâmica

- Informações básicas da Pilha dinâmica:
  - Tamanho;
  - Está cheia?
  - Está vazia?

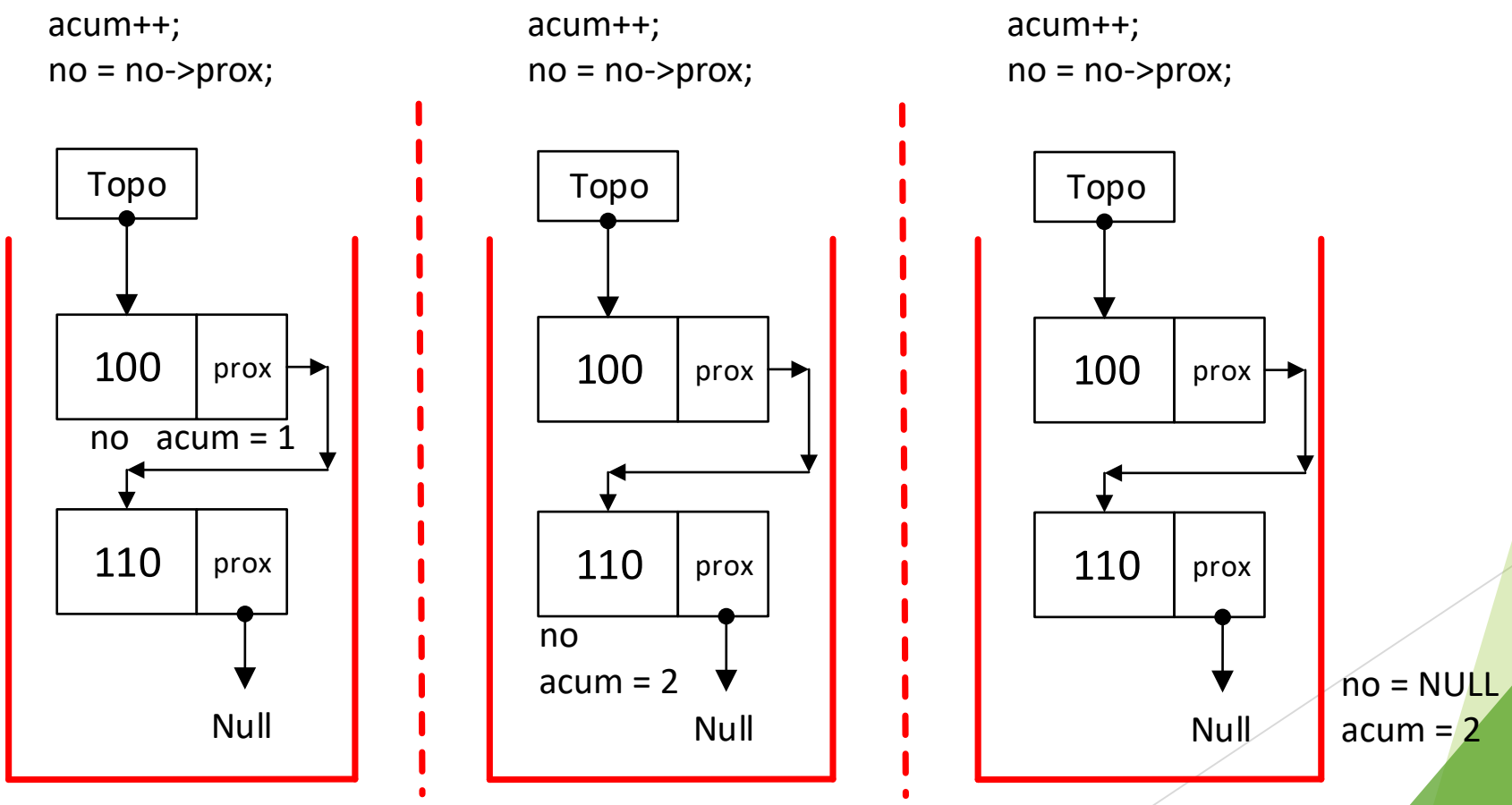
```c
//Arquivo pilhaD.h
int tamanho_pilha(Pilha *pi);
```

```c
//Arquivo principal main()
x = tamanho_pilha(pi);
printf("\nO tamanho da pilha e: %d", x);
```

```c
//Arquivo pilhaD.c
int tamanho_pilha(Pilha *pi){
    if(pi == NULL){
        return 0;
    }
    int acum = 0;
    Elem *no = *pi;
    while(no != NULL){
        acum++;
        no = no->prox;
    }
    return acum;
}
```

Estrutura de Dados 1

Pilha Dinâmica



Estrutura de Dados 1

Pilha Dinâmica – Pilha Cheia

- Em alocações dinâmicas, não faz sentido a verificação de pilha cheia. Tal fato nunca ocorrerá, mas para manter a padronização implementamos a função:

```c
//Arquivo pilhaD.h
int pilha_cheia(Pilha *pi);
```

```c
//Arquivo pilhaD.c
int pilha_cheia(Pilha *pi){
    return 0;
}
```

```c
//Arquivo principal main()
x = pilha_cheia(pi);
if(x){
    printf("\nA Pilha está cheia!");
}else{
    printf("\nA pilha nao esta cheia.");
}
```

Estrutura de Dados 1

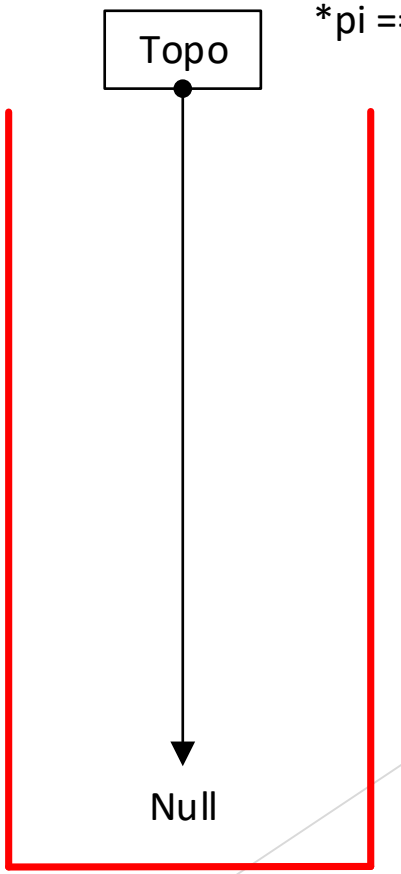
Pilha Dinâmica – Pilha Vazia

```c
//Arquivo pilhaD.h
int pilha_vazia(Pilha *pi);
```

```c
//Arquivo pilhaD.c
int pilha_vazia(Pilha *pi){
    if(pi == NULL){//pilha vazia
        return 1;
    }
    if(*pi == NULL){//topo aponta para NULL não
        return 1;   //temos elementos na pilha
    }
    return 0;
}
```

Temos um topo e ele não aponta para NULL, ou seja, a Pilha não está vazia.

```c
//Arquivo principal main()
x = pilha_vazia(pi);
if(x){
    printf("\nA Pilha está vazia!");
}else{
    printf("\nA pilha nao esta vazia.");
}
```



pi Representação da Pilha
*pi Representação do topo

Estrutura de Dados 1

Pilha Dinâmica – Inserção

- Em uma Pilha, a inserção se dá sempre em seu início (topo);
- Existe também o caso onde a inserção é feita em uma Pilha que está vazia.

```
//Arquivo pilhaD.h
int insere_pilha(Pilha *pi, ALUNO al);
```

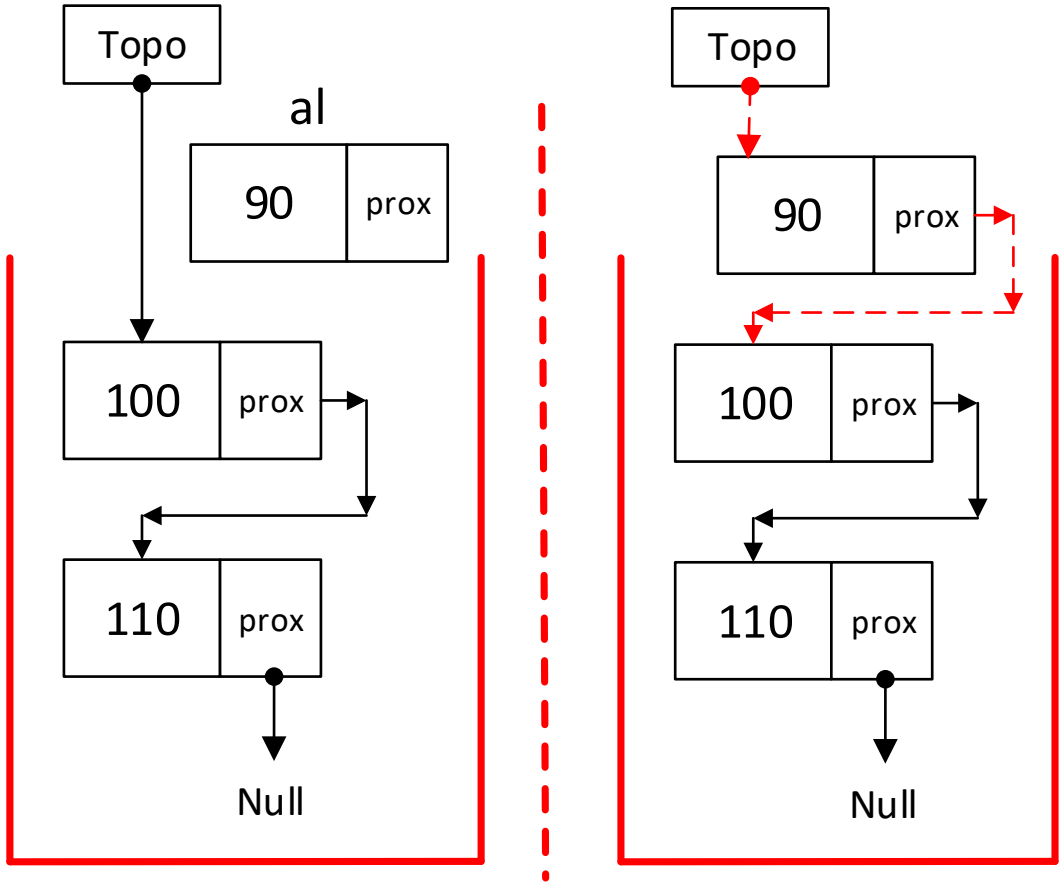
```
//Arquivo pilhaD.c
int insere_pilha(Pilha *pi, ALUNO al){
    if(pi == NULL){
        return 0;
    }
    Elem *no = (Elem*) malloc(sizeof(Elem));
    if (no == NULL){
        return 0;
    }
    no->dados = al;
    no->prox = (*pi);
    *pi = no;
    return 1;
}
```

```
//Arquivo principal main()
x = insere_pilha(pi, al);
if(x){
    printf("\nElemento inserido com sucesso!");
}else{
    printf("\nErro, Elemento nao inserido.");
}
```

Estrutura de Dados 1

Pilha Dinâmica – Inserção

```
no->dados = al;
no->prox = (*pi);
*pi = no;
```



Estrutura de Dados 1

Pilha Dinâmica – Remoção

- Em uma Pilha a Remoção se dá sempre em seu início;
- Cuidado: não se pode remover de uma Pilha que está vazia.

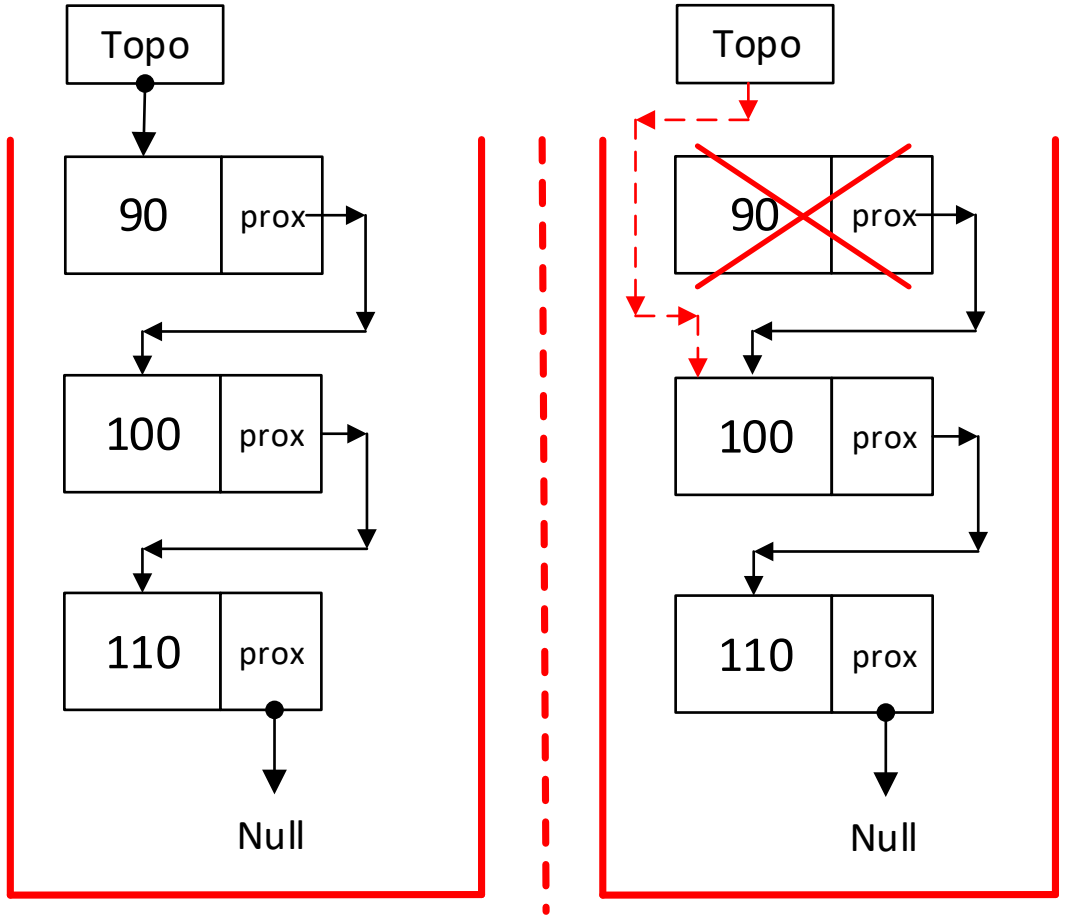
```c
//Arquivo pilhaD.h
int remove_pilha(Pilha *pi);
```

```c
//Arquivo pilhaD.c
int remove_pilha(Pilha *pi){
    if(pi == NULL){//Se Pilha existe
        return 0;
    }
    if((*pi) == NULL){//se pilha vazia
        return 0;
    }
    Elem *no = *pi;//no auxiliar recebe o topo da pilha
    *pi = no->prox;//topo da Pilha recebe próximo elemento
    free(no);
    return 1;
}
```

```c
//Arquivo principal main()
x = remove_pilha(pi);
if(x){
    printf("\nElemento removido com sucesso!");
}else{
    printf("\nErro, Elemento nao removido.");
}
```

Estrutura de Dados 1

Pilha Dinâmica – Remoção




```
Elem *no = *pi;
*pi = no->prox;
free(no);
```

Estrutura de Dados 1

Pilha Dinâmica – Consulta

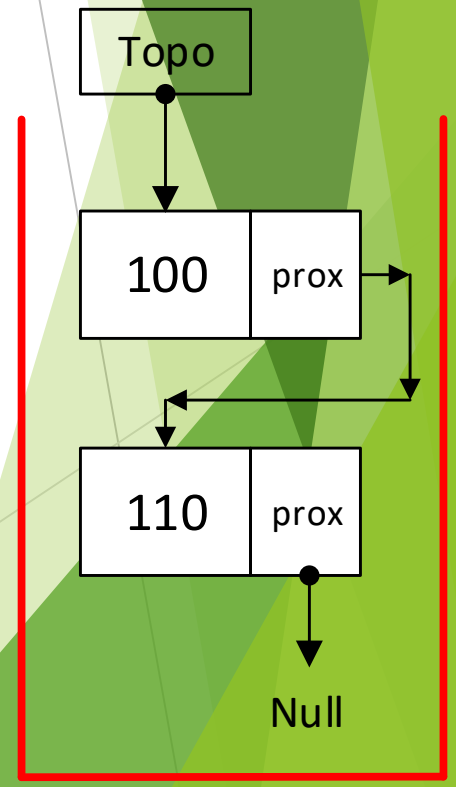
- Em uma Pilha a consulta se dá apenas ao elemento que está no seu início.

```c
//Arquivo pilhaD.h
int consulta_pilha(Pilha *pi, ALUNO *al);
```

```c
//Arquivo pilhaD.c
int consulta_pilha(Pilha *pi, ALUNO *al){
    if(pi == NULL){
        return 0;
    }
    if((*pi) == NULL){
        return 0;
    }
    *al = (*pi)->dados;
    return 1;
}
```

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

*al = (*pi)->dados



Estrutura de Dados 1

Atividade 2

- Entregue os arquivos no Moodle como “atividade 2 - Pilha”