Estruturas de Dados 1

12 – Pilhas Dinâmicas

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

Estrutura de Dados 1

Pilha - Definição

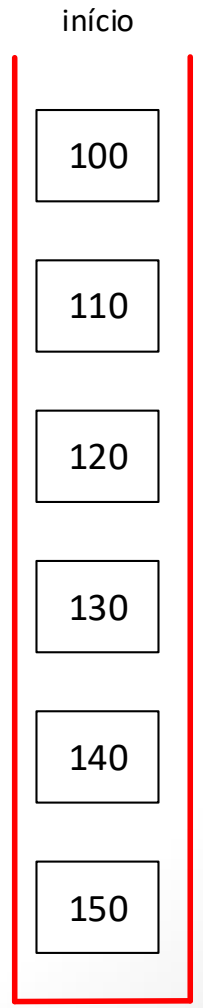
- Sequência de elementos do mesmo tipo, como as “Listas” e “Filas”.
- São estruturas de dados do tipo *LIFO* (*last-in-first-out*), onde o último elemento a ser inserido, será o primeiro a ser retirado. Assim, uma pilha permite acesso a apenas um item de dados - o último inserido. Para processar o penúltimo 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.

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 desmanchá-la



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.

Pilha - Alocação

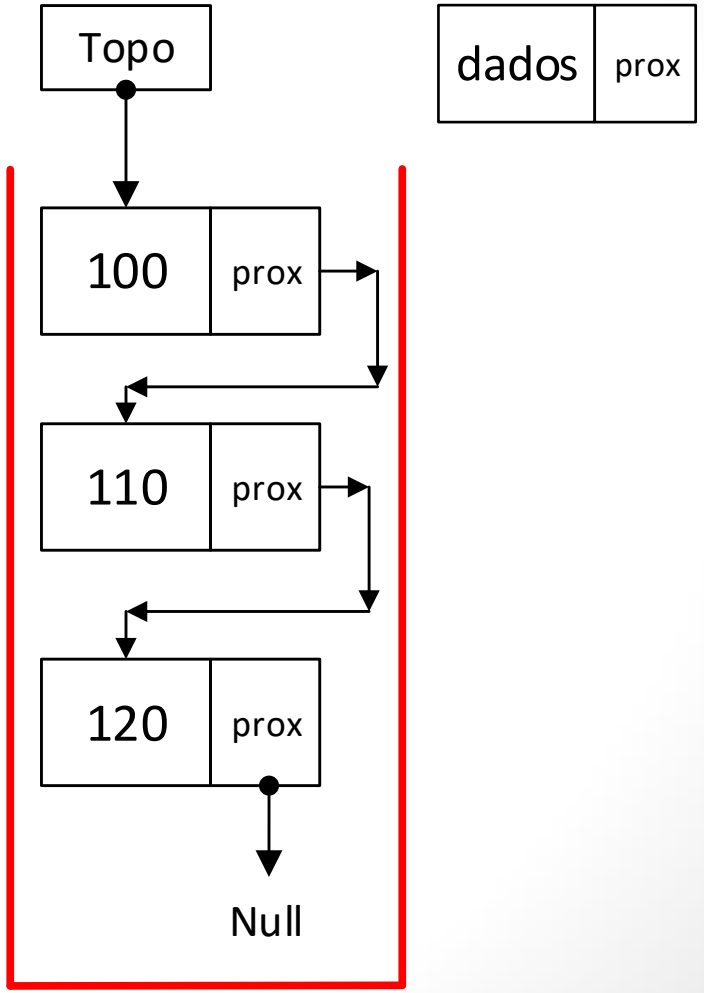
- Alocação Estática:
  - O espaço de memória é alocado no momento da compilação (definição do *Array* – ou vetor);
  - Exige definição do número máximo de elementos que a Pilha suportará;
  - Acesso sequencial: elementos consecutivos na memória.
- Alocação Dinâmica:
  - O espaço de memória é alocado 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 somente o elemento em seu topo, isso não é muito preocupante...

Pilha Dinâmica implementação

- Tipo de Pilha é uma estrutura de dados 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.



Pilha Dinâmica - Implementação

- pilhaD.h
  - Protótipos das funções que manipulam o tipo de dado Pilha;
  - Tipo de dado que será armazenado na Pilha;
  - O ponteiro Pilha.
- pilhaD.c
  - Tipo de dados Pilha;
  - Implementação de suas funções que manipulam o tipo de dado Pilha.

Fila Dinâmica - Implementação

- Variáveis para utilização no programa
  - A variável x será utilizada para o retorno de informações de execução das funções de manipulação do tipo de dado Pilha.
  - As variáveis `al1`, `al2` e `al3` serão inseridas na Fila. A variável `al` será utilizada para o retorno de dados da função consulta.

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

int main()
{
    int x; //para os codigos de erro
    ALUNO al, 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;
```

Pilha Dinâmica - Implementação

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

typedef struct elemento *Pilha;
```

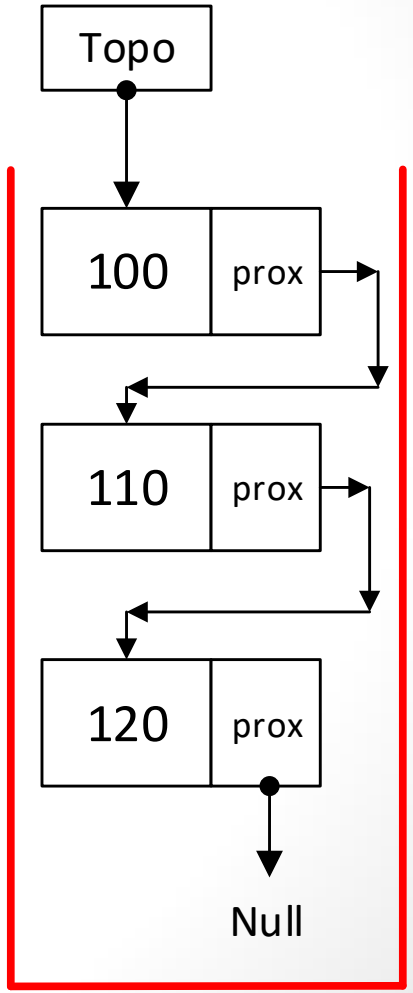
```c
//Arquivo main.c
pi = cria_pilha();
```

```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...



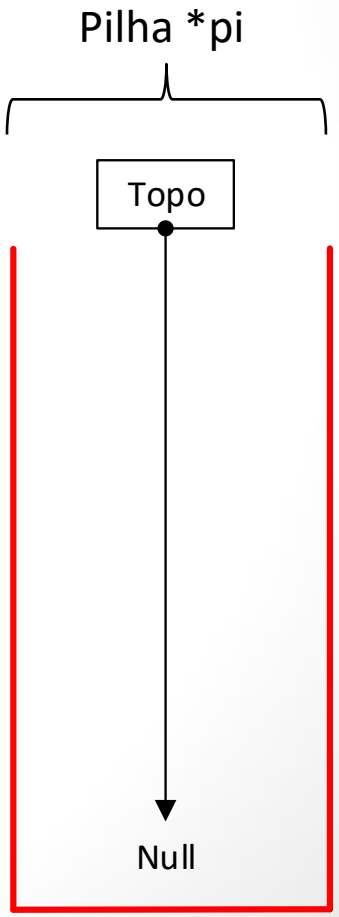
Pilha Dinâmica – Criar Pilha

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

```c
//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

```c
//Arquivo main.c
pi = cria_pilha();
```



Pilha Dinâmica – Destruir Pilha

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

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

Atenção!!
Esta deve ser a última função a ser chamada no `main()`

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

O topo recebe o próximo elemento da Pilha.

Recebe o 1º elemento da Pilha.

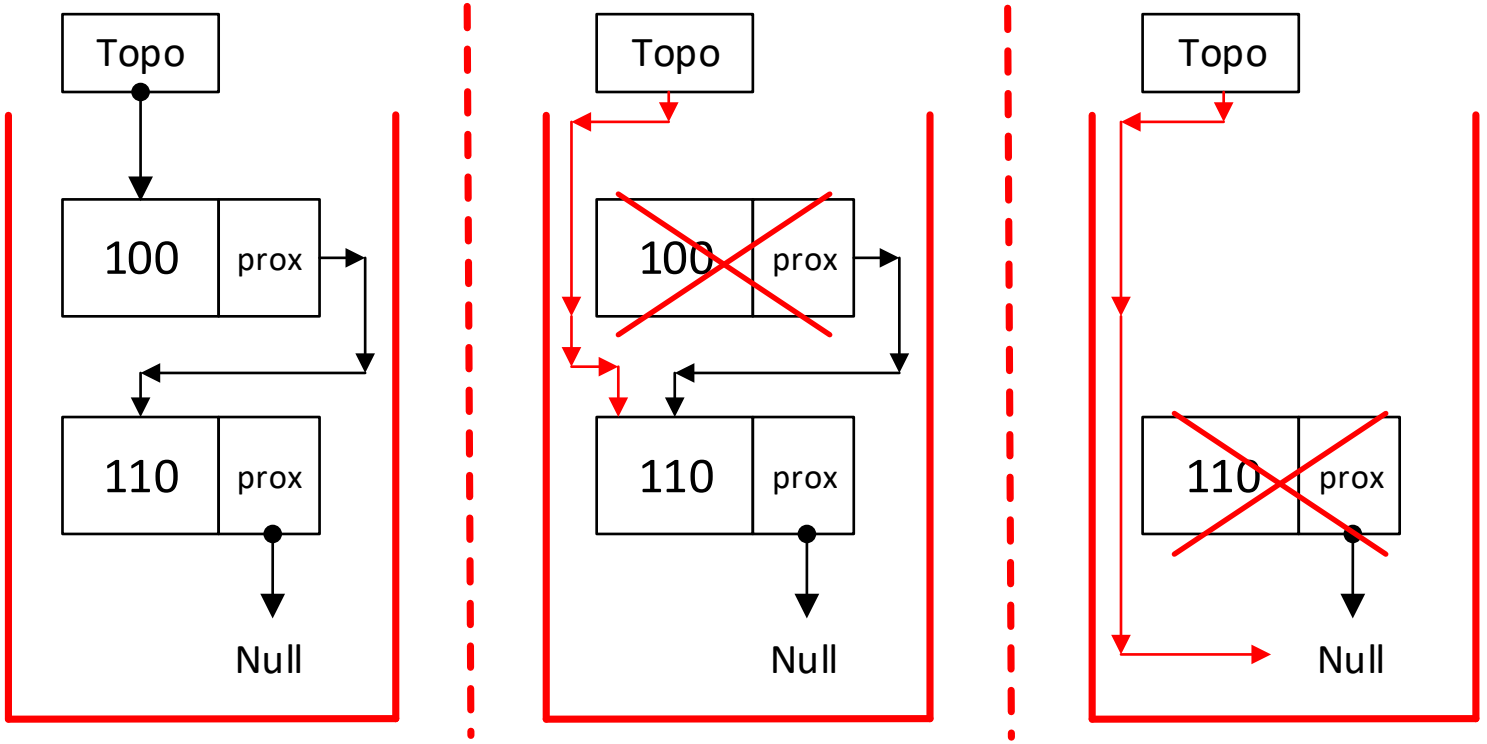
Remove o topo.

Libera o antigo topo da Pilha.

Pilha Dinâmica – Destruir Pilha

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

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



Pilha Dinâmica

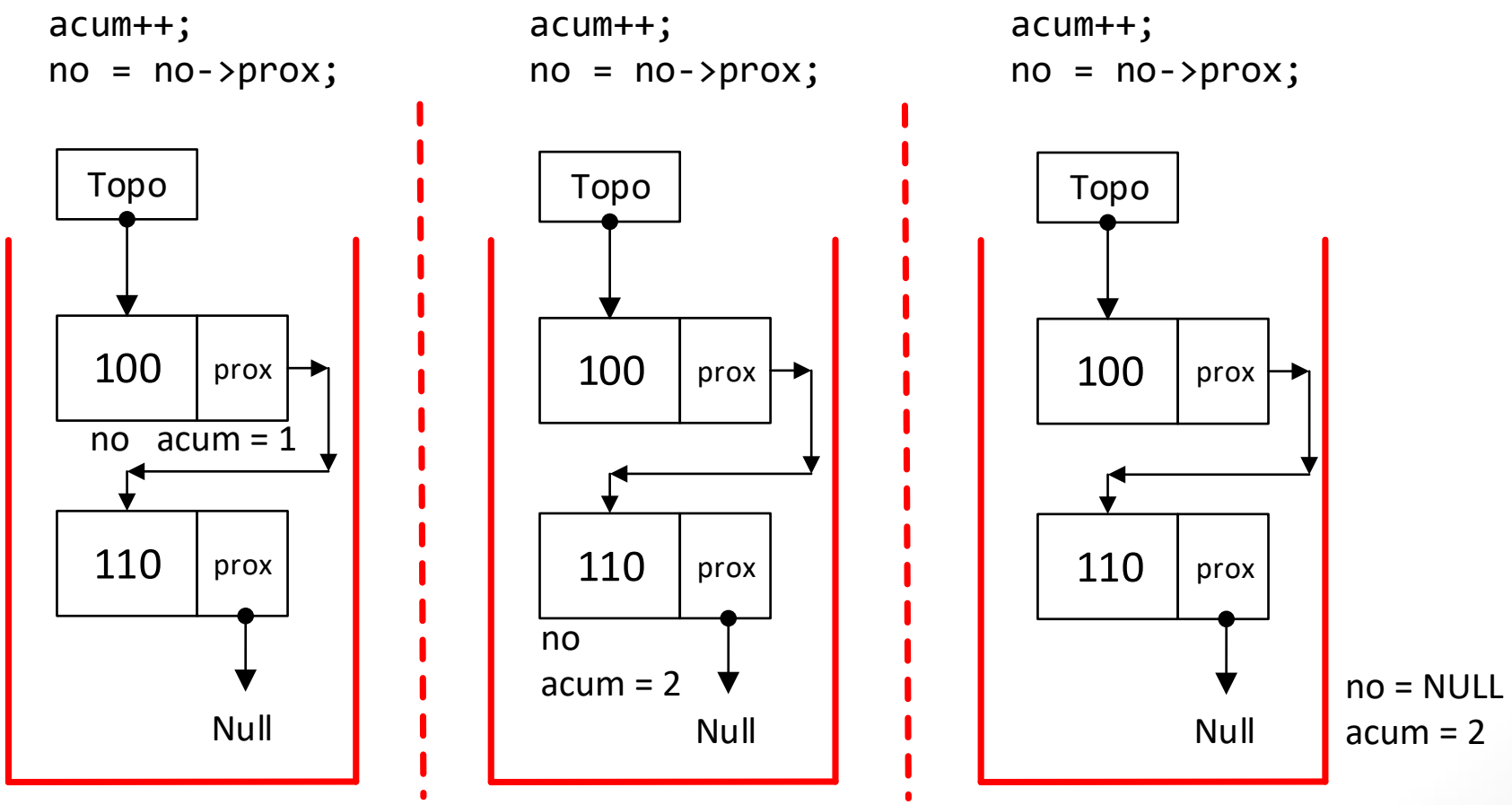
- Informações básicas da Pilha dinâmica:
  - Tamanho;
  - Está cheia?
  - Está vazia?

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

```c
//Arquivo main.c
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;
}
```

Pilha Dinâmica



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

Pilha Dinâmica – Pilha Vazia

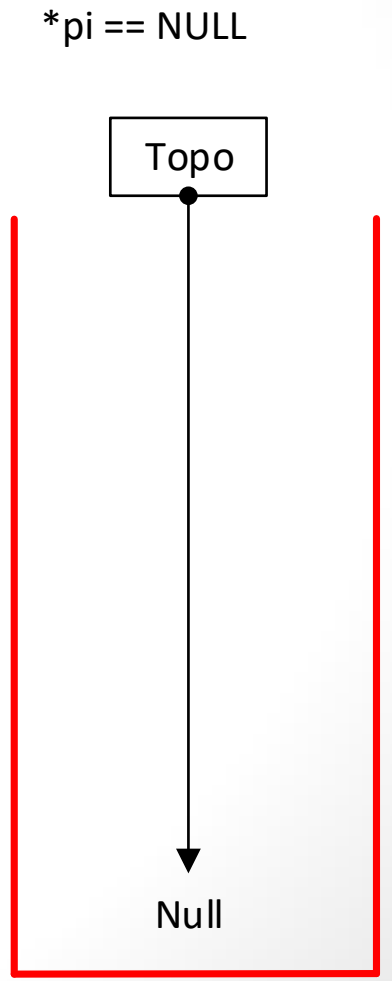
```c
//Arquivo pilhaD.h
int pilha_vazia(Pilha *pi);
```

```c
//Arquivo pilhaD.c
int pilha_vazia(Pilha *pi){
    if(pi == NULL){//não alocada
        return 1;
    }
    if(*pi == NULL){//pilha existe, mas
        return 1;    //sem elementos
    }
    return 0;
}
```

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

```c
//Arquivo main.c
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



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 main.c
x = insere_pilha(pi, al1);
if(x){
    printf("\nElemento inserido com sucesso!");
}else{
    printf("\nErro, Elemento nao inserido.");
}
x = insere_pilha(pi, al2);
if(x){
    printf("\nElemento inserido com sucesso!");
}else{
    printf("\nErro, Elemento nao inserido.");
}
x = insere_pilha(pi, al3);
if(x){
    printf("\nElemento inserido com sucesso!");
}else{
    printf("\nErro, Elemento nao inserido.");
}
```

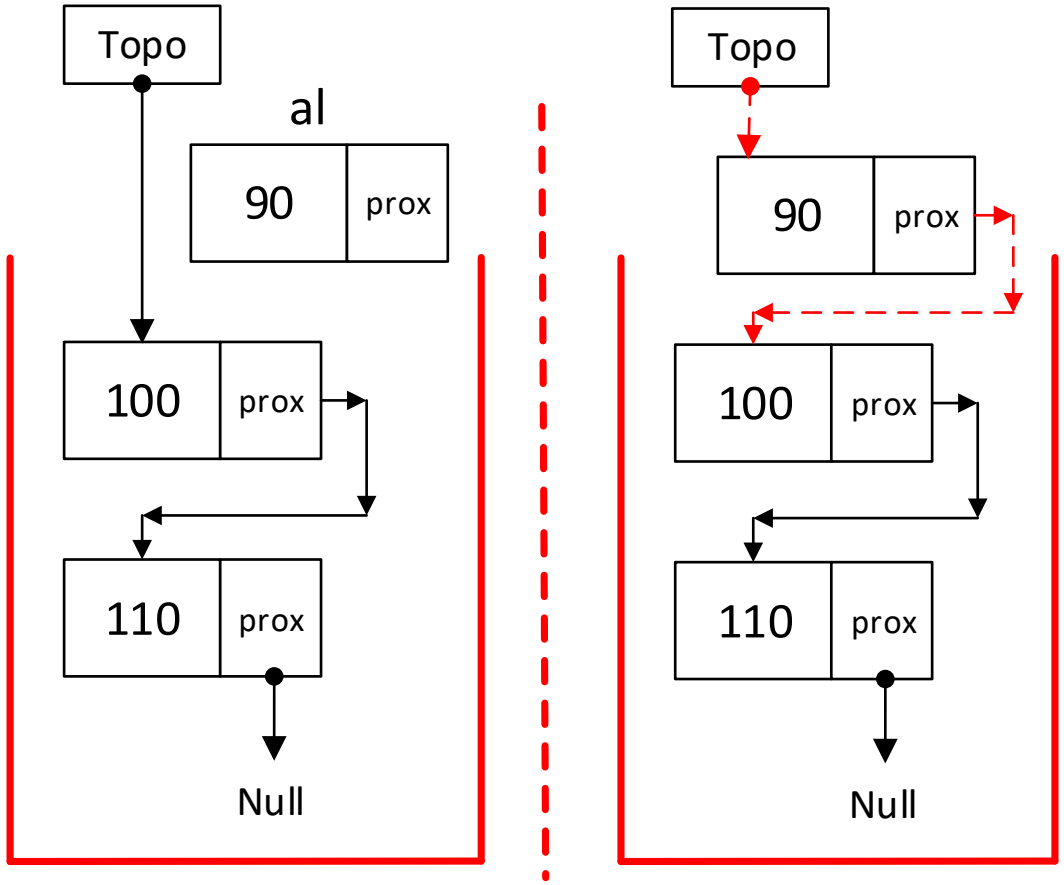
Pilha Dinâmica – Inserção

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

```c
//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;
}
```

Pilha Dinâmica – Inserção

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



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);
```

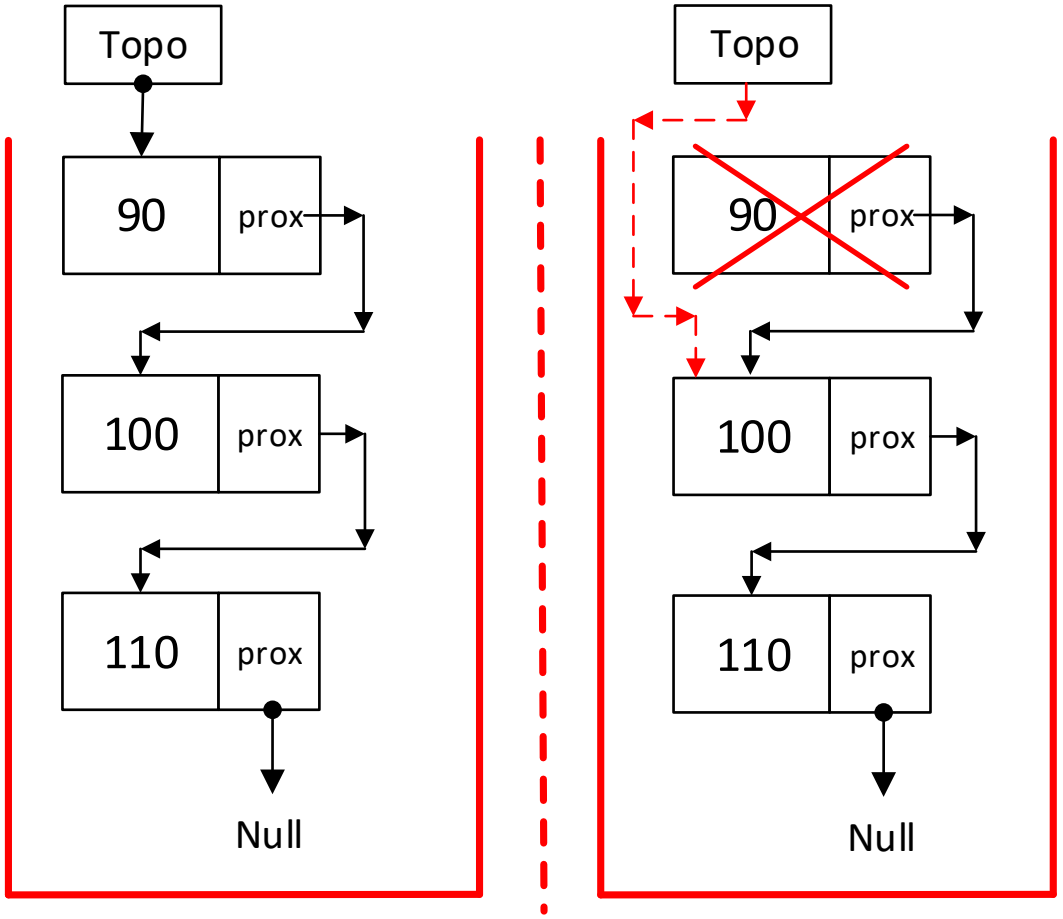
```c
//Arquivo pilhaD.c
int remove_pilha(Pilha *pi){
    if(pi == NULL){//se pilha
        return 0;  //não alocada
    }
    if((*pi) == NULL){//se pilha vazia
        return 0;
    }
    Elem *no = *pi;
    *pi = no->prox;
    free(no);
    return 1;
}
```

Nó auxiliar recebe o topo da pilha

Topo da pilha recebe o próximo elemento

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

Pilha Dinâmica – Remoção




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

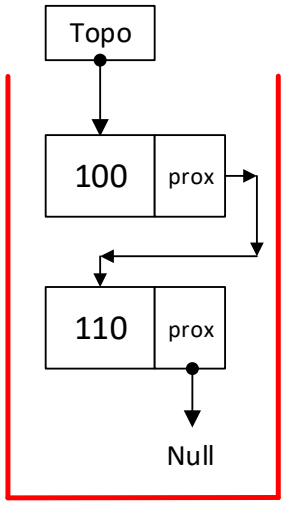
Pilha Dinâmica – Consulta

- 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;
}
```

*al = (*pi)->dados




```c
//Arquivo main.c
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.");
}
```

Estrutura de Dados 1

Atividade 2

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