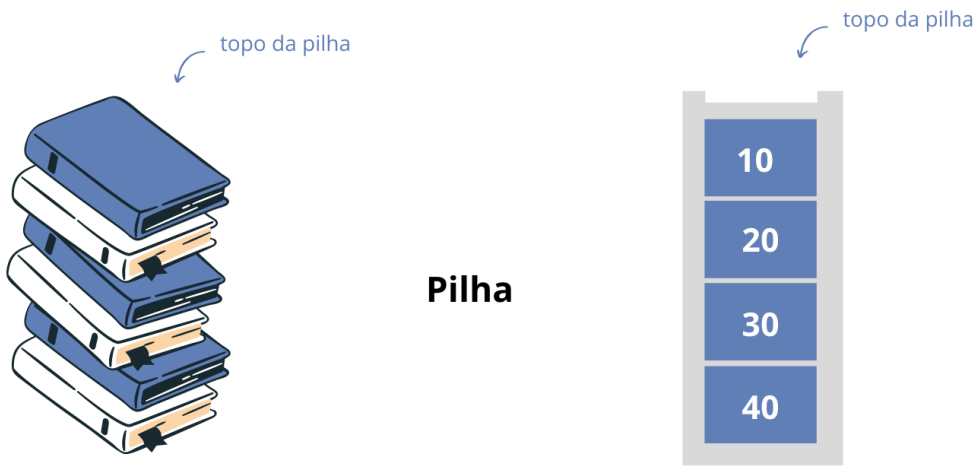
Algoritmos e Programação I

Módulo 8 - Pilha: Estática e Encadeada

Profª. Elisa de Cássia Silva Rodrigues

Pilha

- Definição:
  - Estrutura de dados do tipo lista, com restrições para inserção e remoção de elementos, utilizada para armazenar e organizar uma sequência de elementos do mesmo tipo.
  - As inserções e remoções ocorrem na mesma extremidade da pilha, chamada de topo (início ou final).
  - Estrutura do tipo LIFO (*Last In First Out*), onde o último elemento a entrar é o primeiro a sair.



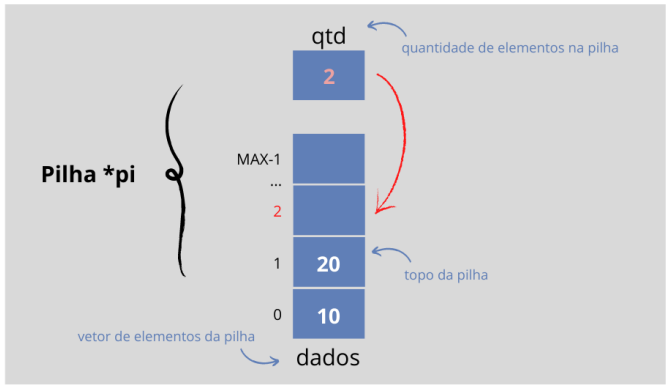
Pilha

- Operações básicas:
  - Criação da pilha.
  - Inserção de um elemento no topo da pilha (início).
  - Remoção de um elemento do topo da pilha (início).
  - Consulta ao elemento do topo da pilha.
  - Destruição da pilha.
  - Informação sobre tamanho da pilha.
  - Informação sobre a pilha estar vazia ou cheia.

A implementação das operações de uma pilha depende do tipo de alocação de memória usada (estática ou dinâmica).

Pilha Estática

- Definição:
  - Estrutura de dados do tipo pilha que é definida utilizando alocação estática e acesso sequencial dos elementos.
- Características:
  - Definida por um vetor com elementos sequenciais na memória.
  - Exige a definição prévia do número máximo de elementos (MAX).
  - Campo adicional para armazenar a quantidade de elementos na pilha.
  - Deve-se definir a extremidade da pilha que representará o topo:
    - ★ *Na Pilha Estática definimos o final da pilha como topo.*



Pilha Estática

- Definição do TAD Pilha Estática:
  - Definir os arquivos `pilhaEstatica.h` e `pilhaEstatica.c`.
  - Declarar o tipo de dado que irá representar a pilha no arquivo `.h`:

```
typedef struct pilha Pilha;
```

  - Definir o tipo de dado que será armazenado dentro da pilha (`int`).
  - Declarar a estrutura para representar a pilha estática no arquivo `.c`:

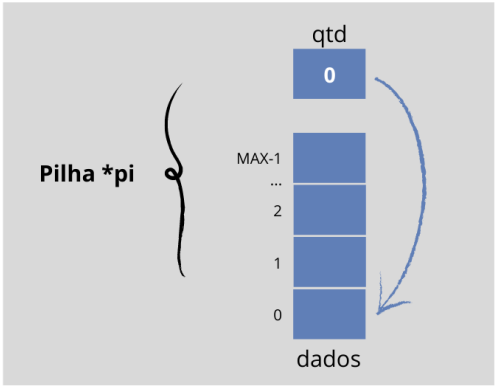
```
struct pilha
  int qtd;
  int dados[MAX]; // MAX representa o tamanho da pilha
};
```

- Declarar um ponteiro do tipo `Pilha` para acessar o TAD (`main.c`):

```
Pilha *pi;
```

Pilha Estática

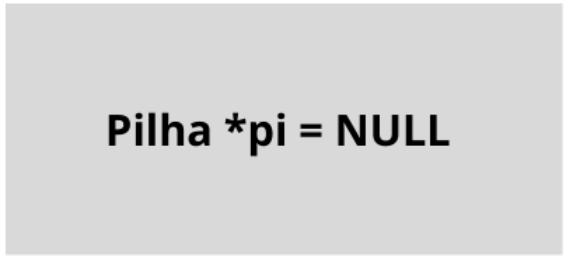
- Criação da pilha:
  - Antes de usar uma pilha é preciso criar uma pilha vazia.
  - Isto é, alocar um espaço na memória para a estrutura:
    - ★ Alocação dinâmica da estrutura `Pilha` usando `malloc()`.
  - A pilha está vazia, quando `qtd = 0`.



Note que o vetor `dados[]` que armazena os elementos da pilha é alocado estaticamente durante a alocação da estrutura `Pilha`.

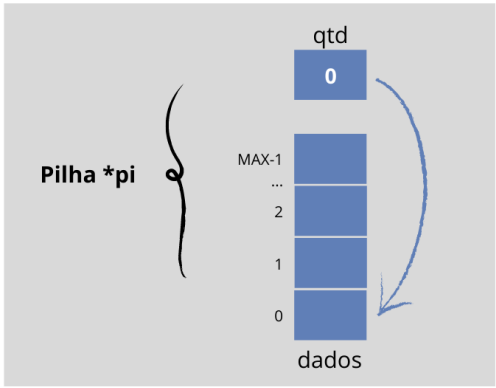
Pilha Estática

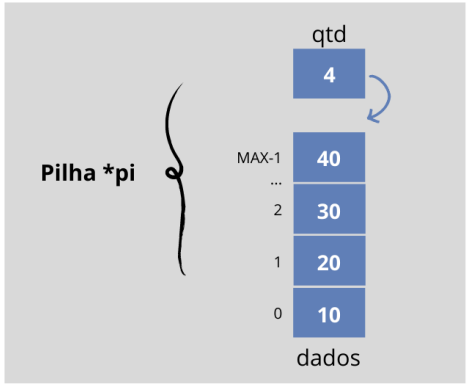
- Destruição da pilha:
  - Deve-se liberar a memória alocada para a estrutura:
    - ★ Liberação da estrutura `Pilha` usando `free()`.



Pilha Estática

- Informações básicas sobre a pilha:
  - Tamanho da pilha (valor do campo `qtd`).
  - Pilha vazia (`qtd = 0`).
  - Pilha cheia (`qtd = MAX`).



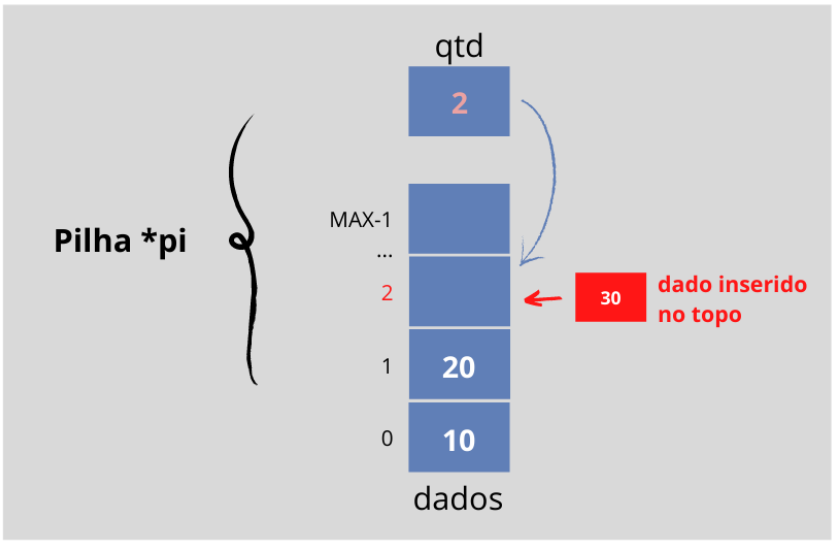


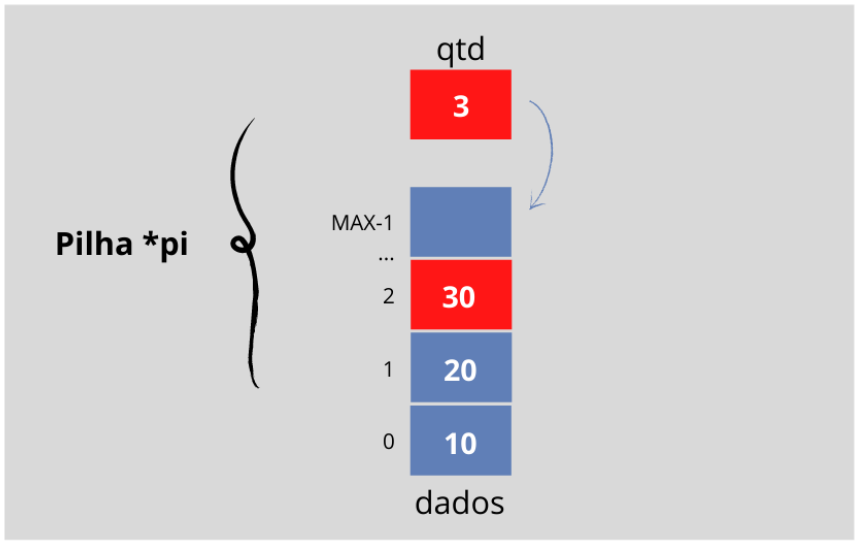
Pilha Estática

- Inserção (empilhar ou *push*):
  - Ato de guardar elementos dentro da pilha.
  - Apenas inserção no final da pilha (topo).
  - Operação de inserção envolve o teste de estouro da pilha:
    - ★ Necessário verificar se é possível inserir um novo elemento na pilha.
    - ★ Ou seja, se a pilha não está cheia.

Pilha Estática

- Inserção no final da pilha (topo):
  - Não envolve o deslocamento de elementos do vetor.
  - O novo elemento é inserido logo após a última posição ocupada do vetor, ou seja, na primeira posição livre que é indicada pelo índice armazenado na variável `qtd` da estrutura `Pilha`.
  - Ao fim da operação deve-se incrementar o campo `qtd`.



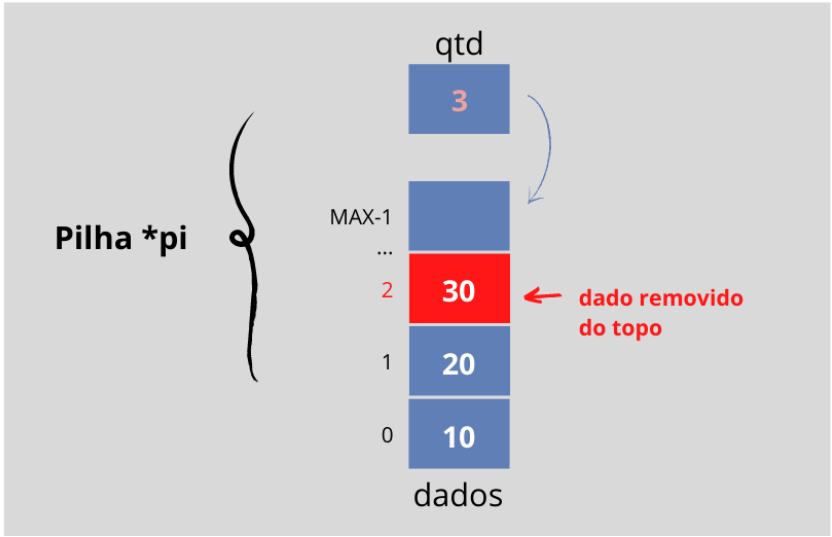


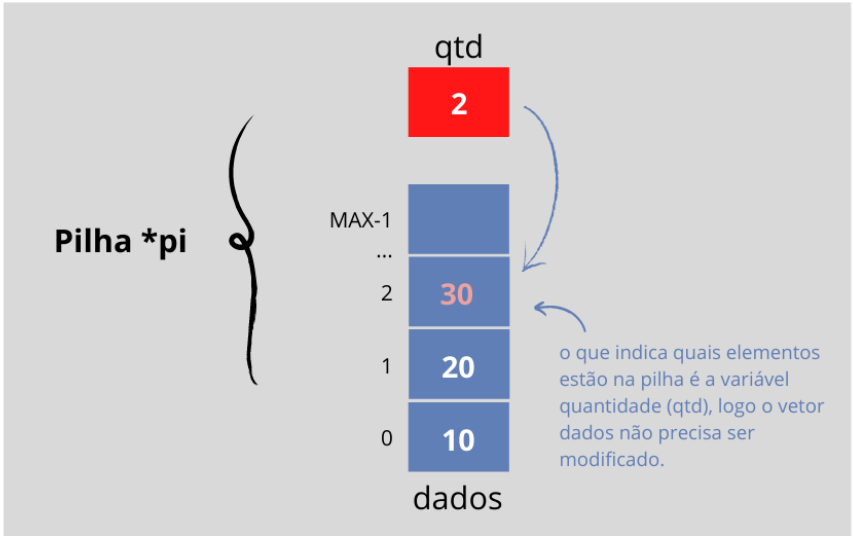
Pilha Estática

- Remoção (desempilhar ou *pop*):
  - Existindo uma pilha, e ela possuindo elementos, é possível excluí-los.
  - Apenas remoção do final da pilha (topo).
  - Operação de remoção envolve o teste de pilha vazia.
    - ★ Necessário verificar se existem elementos dentro da pilha.
    - ★ Ou seja, se a pilha não está vazia.

Pilha Estática

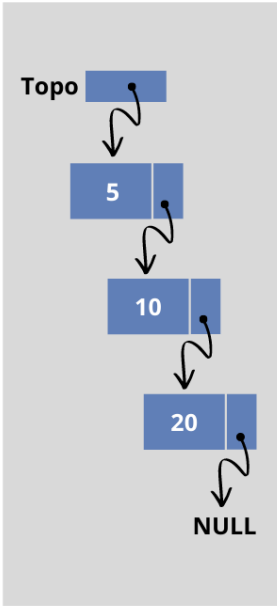
- Remoção do final (topo) da pilha:
  - Não envolve o deslocamento de elementos do vetor.
  - O elemento a ser removido ocupa a última posição da pilha, ou seja, a posição indicada pelo índice armazenado na variável `qtd - 1` da estrutura `Pilha`.
  - Ao fim da operação deve-se decrementar o campo `qtd`.





Pilha Encadeada

- Definição:
  - Estrutura de dados do tipo pilha que é definida utilizando alocação dinâmica e acesso encadeado dos elementos.
- Características:
  - Definir a extremidade da pilha que representará o topo:
    - ★ *Na Pilha Encadeada, o topo é o início da pilha.*
  - Armazenar um ponteiro que indica o topo da pilha.
  - Cada elemento possui um dado e um ponteiro para o próximo da pilha.
  - Cada elemento é alocado dinamicamente quando é inserido na pilha.
  - Se um elemento é removido, a memória alocada para ele é liberada.



Pilha Encadeada

- Definição do TAD Pilha Encadeada:
  - Definir os arquivos `pilhaEncadeada.h` e `pilhaEncadeada.c`.
  - Declarar o tipo de dado que irá representar a pilha no arquivo `.h`:

```
typedef struct elemento* Pilha;
```

  - Definir o tipo de dado que será armazenado dentro da pilha (`int`).
  - Declarar a estrutura para representar a pilha encadeada no arquivo `.c`:

```
struct elemento{
  int dado;
  struct elemento *prox;
};
typedef struct elemento Elemento;
```

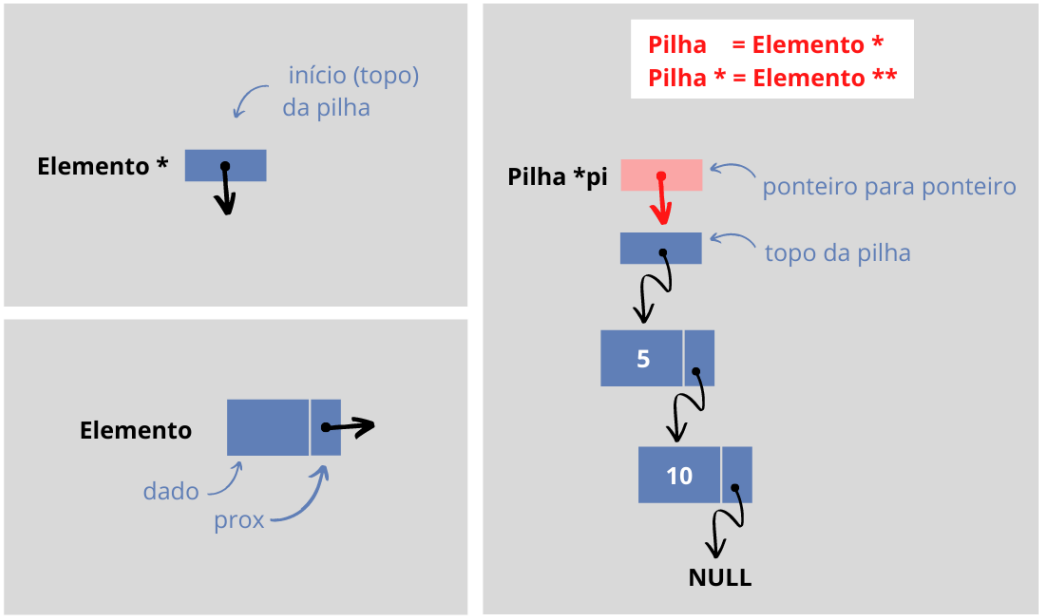
- Declarar um ponteiro do tipo `Pilha` para acessar o TAD (`main.c`):

```
Pilha *pi;
```

Pilha Encadeada

- Ilustração dos tipos de dados Pilha e Elemento:

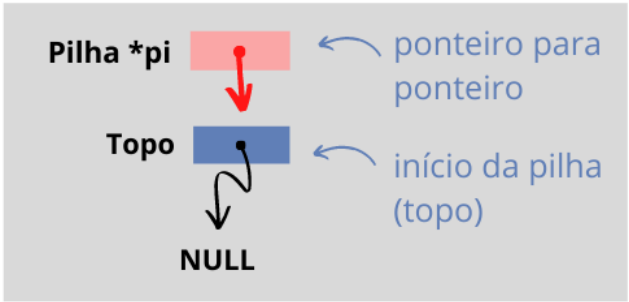
```
typedef struct elemento Elemento;
typedef struct elemento* Pilha;
```



Note que, nesta implementação, a estrutura Pilha é abstrata (ponteiro para Elemento), ou seja, não é definida uma `struct pilha` (nó descritor).

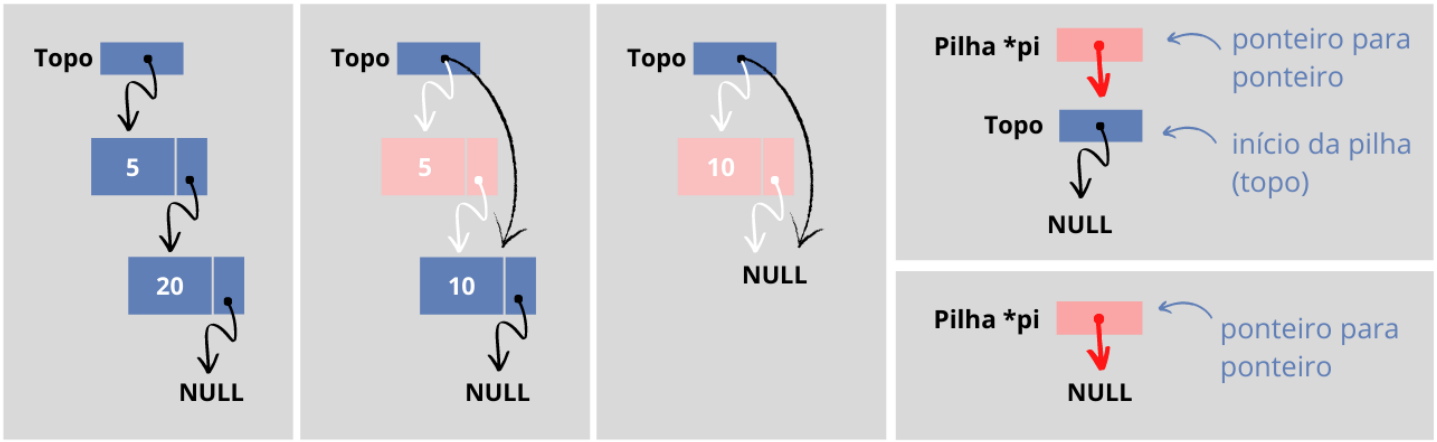
Pilha Encadeada

- Criação da pilha:
  - Antes de usar uma pilha é preciso criar uma pilha vazia.
  - Isto é, alocar um espaço na memória para o descritor da pilha:
    - ★ Alocação dinâmica de um ponteiro do tipo `Pilha` usando `malloc()`.
  - A pilha está vazia, quando `pi != NULL` e `*pi == NULL`.



Pilha Encadeada

- Destruição da pilha:
  - Inicialmente, deve-se liberar a memória alocada para todos os elementos da pilha:
    - ★ Liberação da estrutura `Elemento` usando `free()`.
  - Deve-se liberar a memória alocada para o ponteiro do topo da pilha:
    - ★ Liberação do ponteiro do tipo `Pilha` usando `free()`.

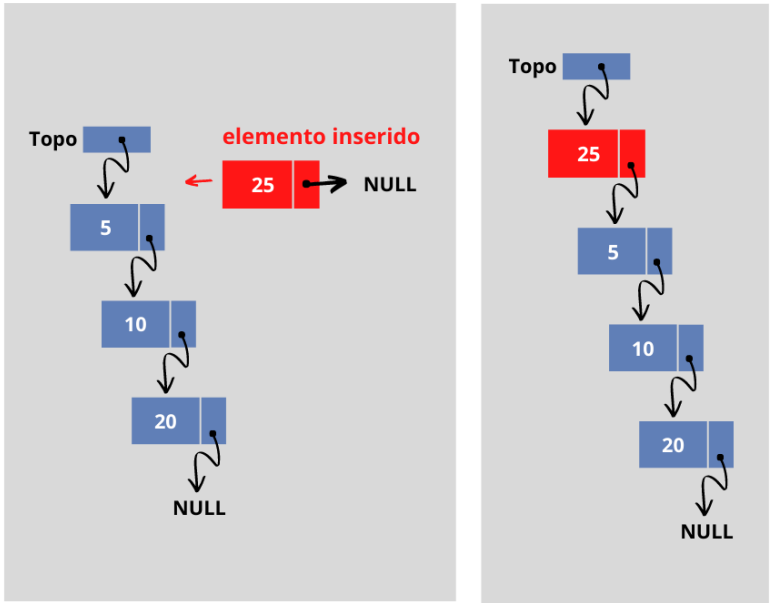


Pilha Encadeada

- Inserção (empilhar ou *push*):
  - Ato de guardar elementos dentro da pilha.
  - Apenas inserção no início da pilha (topo).
  - Operação de inserção envolve alocação dinâmica de memória:
    - ★ Necessário verificar se a pilha existe (`pi != NULL`).
    - ★ Se existir, deve-se verificar se o novo elemento foi alocado corretamente.

Pilha Encadeada

- Inserção no início da pilha (topo):
  - Envolve a criação de um novo elemento (alocação de memória).
  - Atribui-se o valor do novo elemento ao campo `dado`.
  - O ponteiro `prox` do novo elemento aponta para o elemento do topo.
  - O ponteiro que indica o topo da pilha (`*pi`) aponta para o novo elemento.

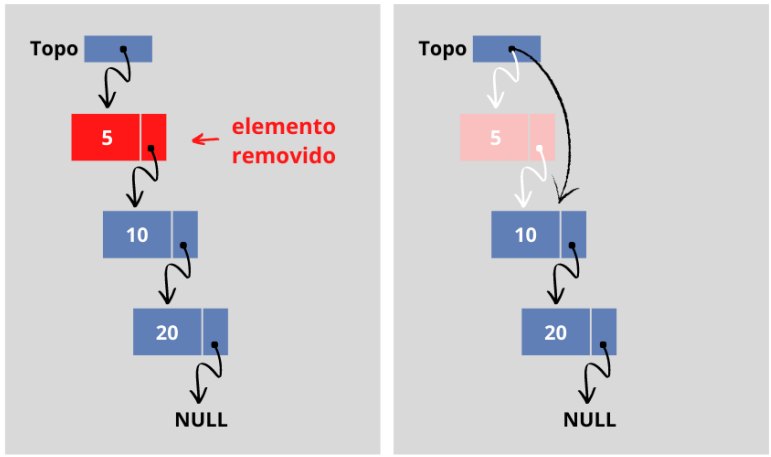


Pilha Encadeada

- Remoção (desempilhar ou *pop*):
  - Existindo uma pilha, e ela possuindo elementos, é possível excluí-los.
  - Apenas remoção no início da pilha (topo).
  - Operação de remoção envolve o teste de pilha vazia.
    - ★ Necessário verificar se a pilha existe (`pi != NULL`).
    - ★ Se existir, deve-se verificar se existem elementos dentro da pilha.
    - ★ Ou seja, se a pilha não está vazia (`*pi != NULL`).

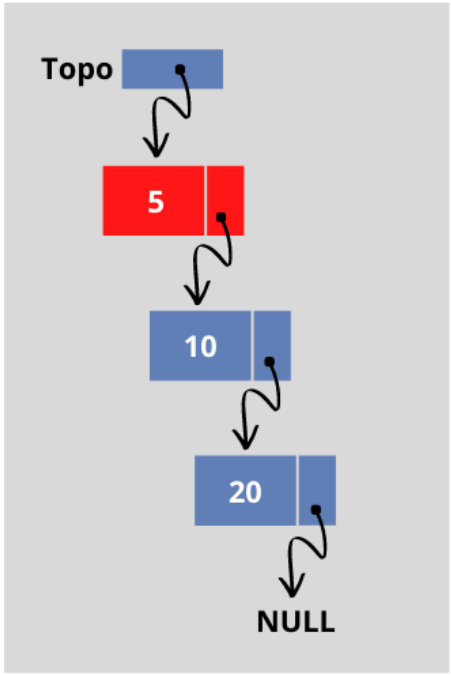
Pilha Encadeada

- Remoção do início da pilha (topo):
  - Envolve a liberação da memória alocada para o elemento do topo.
  - O ponteiro que indica o elemento do topo (`*pi`) aponta para o 2º elemento da pilha.
  - Por fim, libera-se a memória do 1º elemento usando a função `free()`.



Pilha Encadeada

- Consulta:
  - A pilha permite acesso apenas ao elemento do topo (`*pi`).



Pilha Estática x Pilha Encadeada

Características	Pilha Estática	Pilha Encadeada
Vantagem	Facilidade de criar e destruir a pilha	Melhor uso da memória
	Complexidade dos algoritmos não depende do tamanho da pilha: *O (1)*	Complexidade dos algoritmos de inserção, remoção e consulta não depende do tamanho da pilha: *O (1)*
	Não é necessário percorrer a pilha toda para destruí-la.	Não precisa definir tamanho máximo da pilha
Desvantagem	Necessidade de definir o tamanho máximo	Necessidade de percorrer a pilha toda para destruí-la.
Utilização	Quando o tamanho máximo é bem definido	Quando o tamanho máximo não é bem definido
	Pilhas pequenas	

- Implementação TAD Pilha Encadeada:

  https://repl.it/@elisa_rodrigues/Modulo8-PilhaEncadeada

- Exercício para fixação:

  Implemente a TAD Pilha Estática.

Referências Bibliográficas

1. BACKES, A. ***Estrutura de dados descomplicada em linguagem C***. 2016.

   -> **Capítulo 8: Pilhas**

   -> **Material Complementar - Vídeo aulas (38ª a 44ª):**

   https://programacaodescomplicada.wordpress.com/indice/estrutura-de-dados/