ED62A-COM2A
ESTRUTURAS DE DADOS

Aula 03b - Pilhas
(Implementação dinâmica)

Prof. Rafael G. Mantovani

27/08/2019

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil
Agosto, 2019

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

Introdução

- Pilhas Estáticas
  - ?

Introdução

- Pilhas Estáticas
  - vetor de N elementos
  - variável que controla o índice do topo

- Pilhas dinâmicas

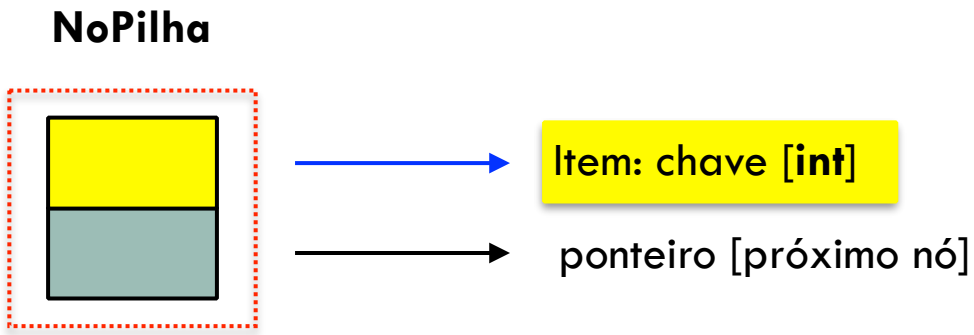
Introdução

- Pilhas Estáticas
  - vetor de N elementos
  - variável que controla o índice do topo

- Pilhas dinâmicas
  - elementos do tipo NoPilha
  - ponteiros
  - topo é um ponteiro para NoPilha

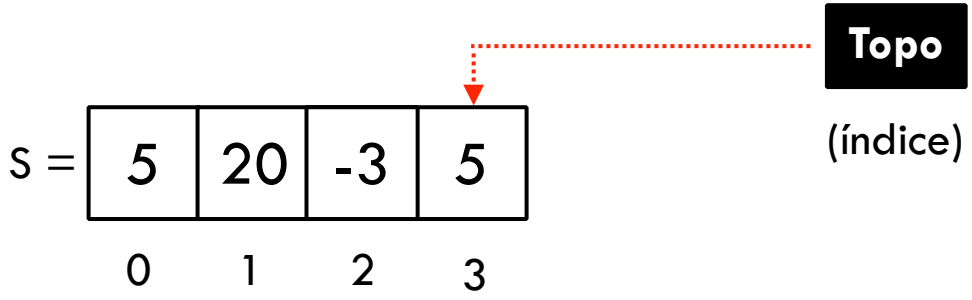
Introdução

- Elemento (objeto) → vários atributos



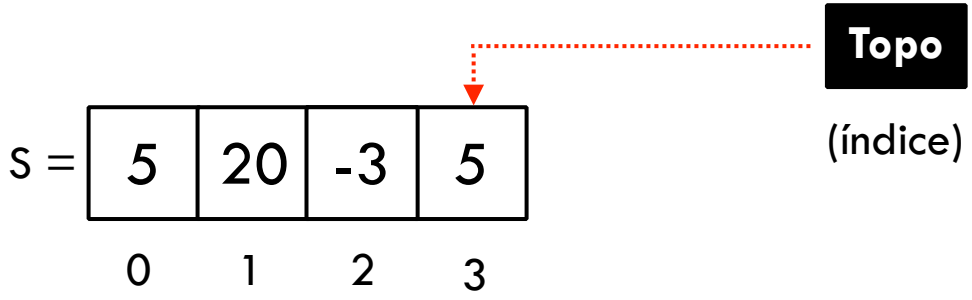
Introdução

* Pilha estática

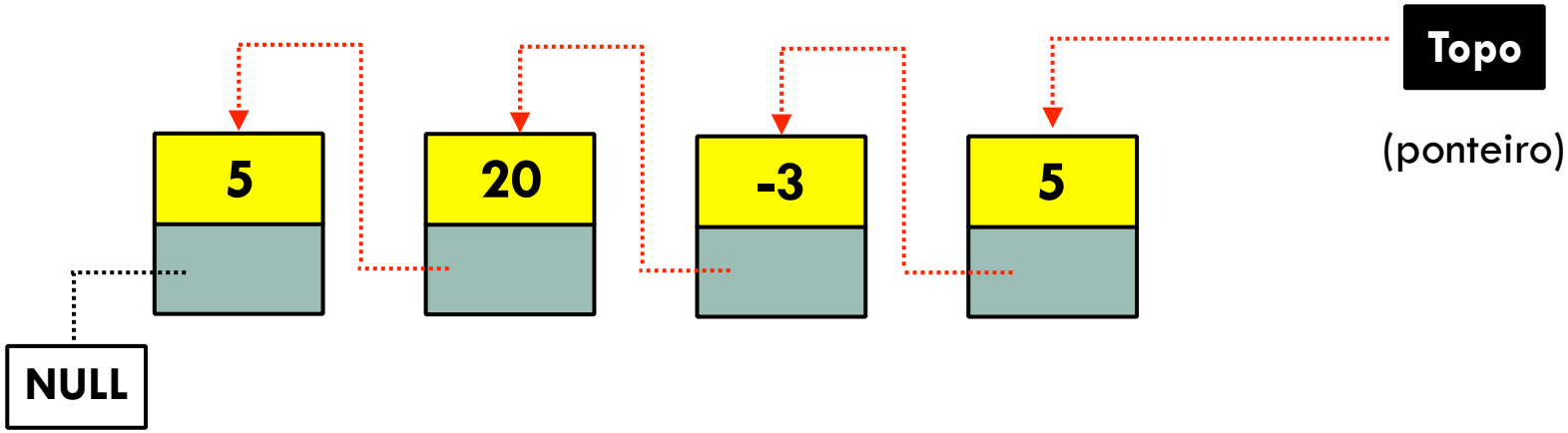


Introdução

* Pilha estática



* Pilha dinâmica



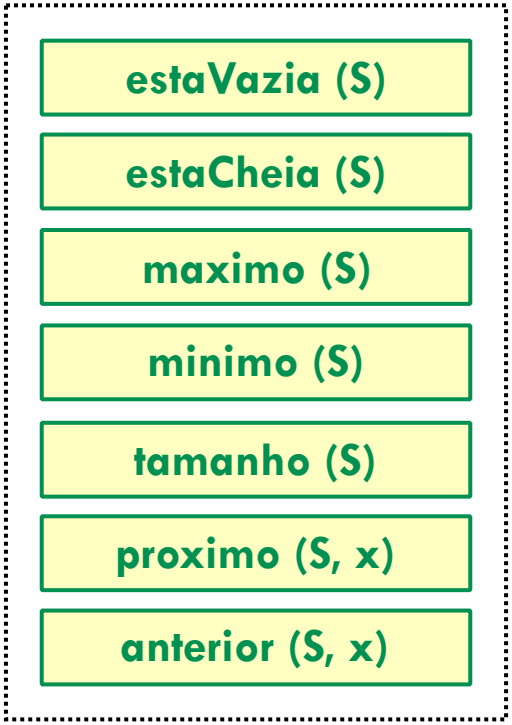
Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

Operações em Pilhas Dinâmicas

Dada uma estrutura **S**, chave **k**, elemento **x**:

Operações de modificação



Operações adicionais de consulta

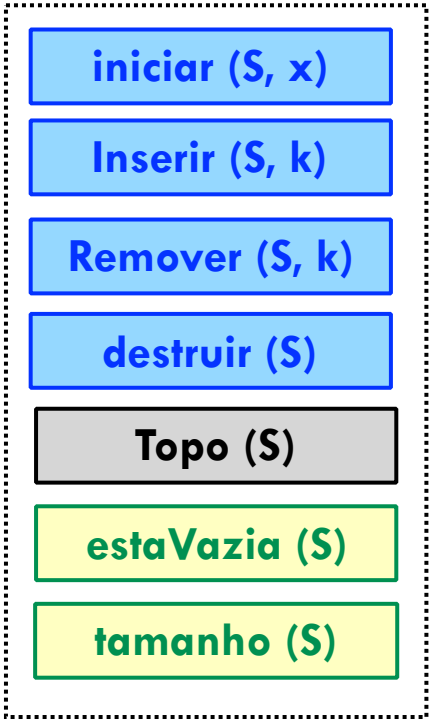
Operações em Pilhas Dinâmicas

Dada uma estrutura **S**, chave **k**, elemento **x**:

Operações de modificação

Operações adicionais de consulta

Operações em Pilhas Dinâmicas



Inicializa a pilha e suas variáveis

Inserir objeto na pilha (empilhar)

Remover objeto da pilha (desempilhar)

Destruir estrutura

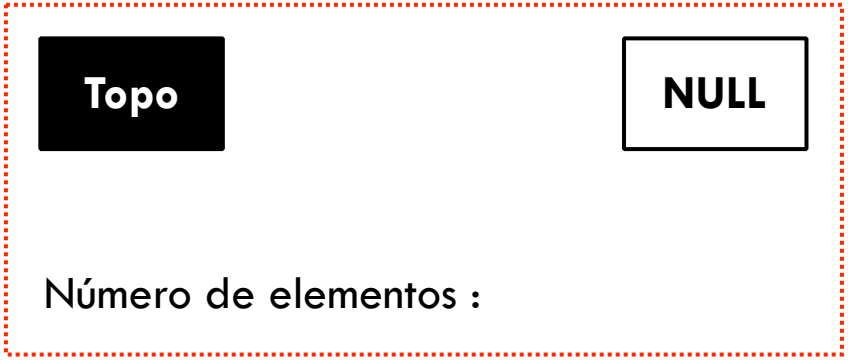
Retorna o objeto do topo, sem remover

Retorna booleano indicando se a pilha está vazia

Retorna a quantidade de elementos na pilha

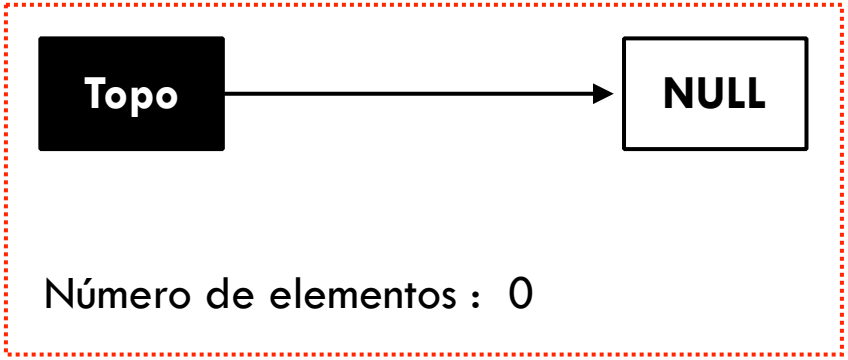
Inicialização da pilha

tipo Pilha Dinâmica



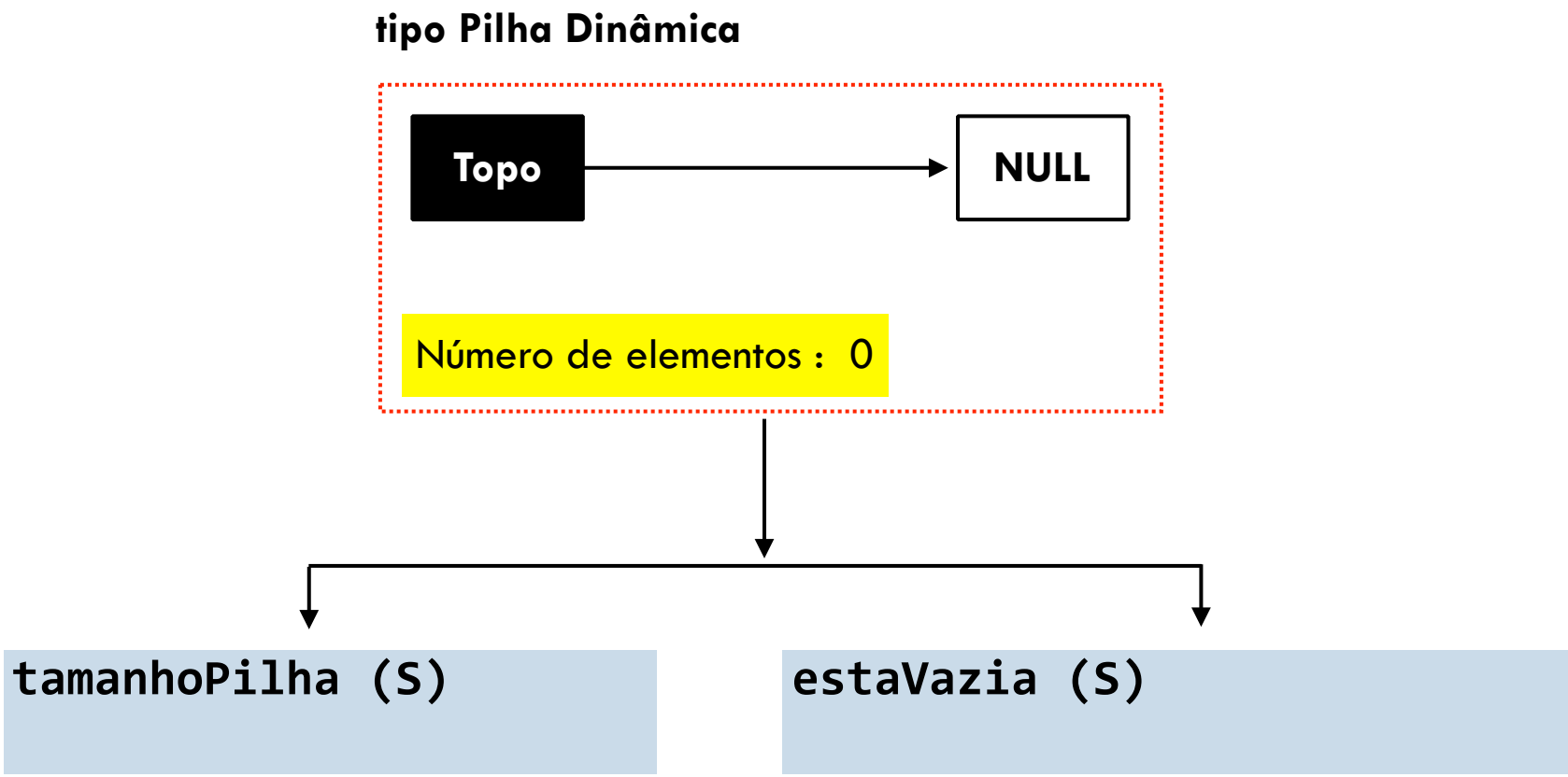
Inicialização da pilha

tipo Pilha Dinâmica

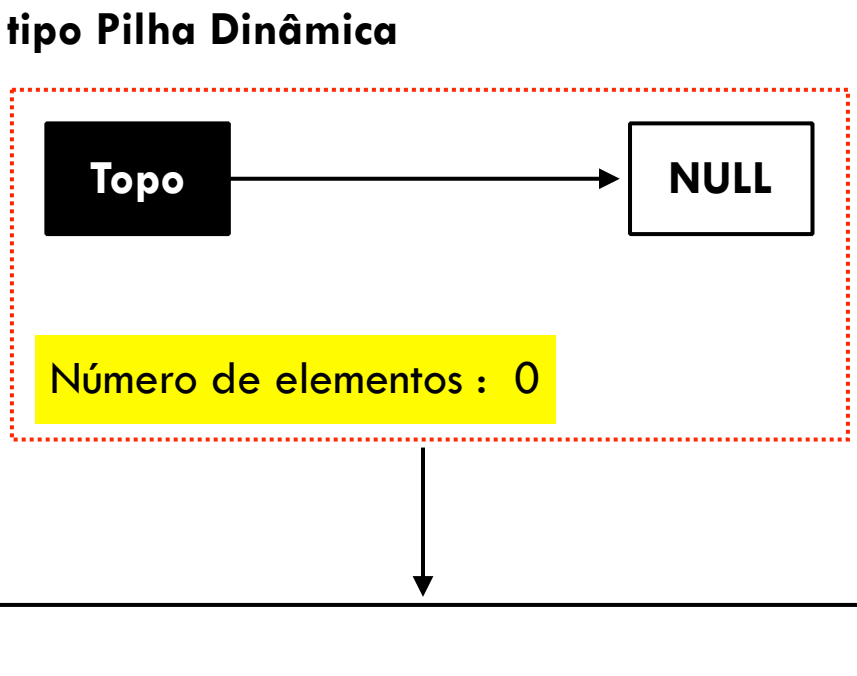



```
IniciaPilha (S)
1. S.topo = NULL;
2. S.tamanho = 0;
```

Tamanho da Pilha



Tamanho da Pilha

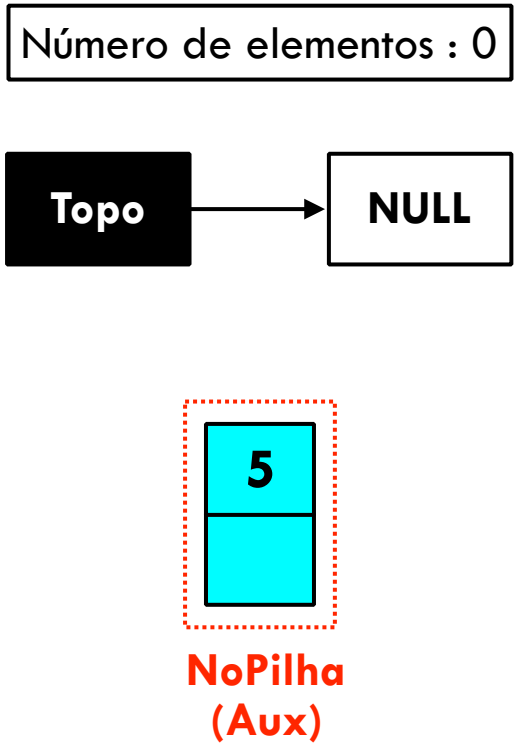



```
tamanhoPilha (S)
1. return (S.tamanho);
```

```
estaVazia (S)
1. return (S.tamanho == 0);
```

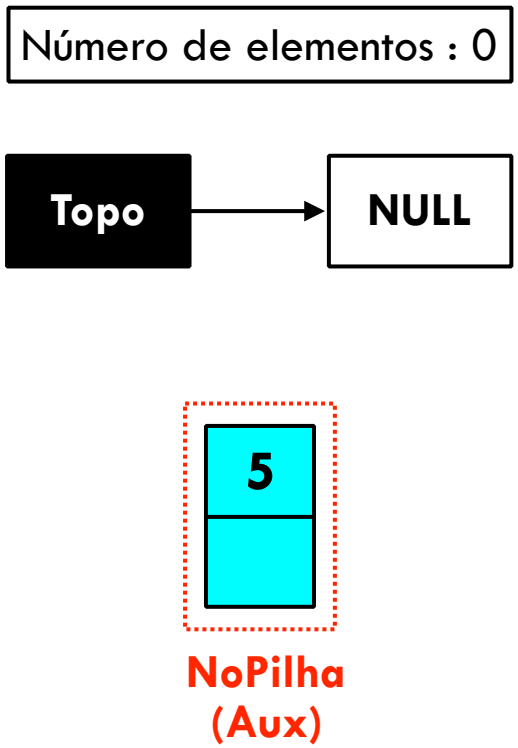
Inserção (Push)

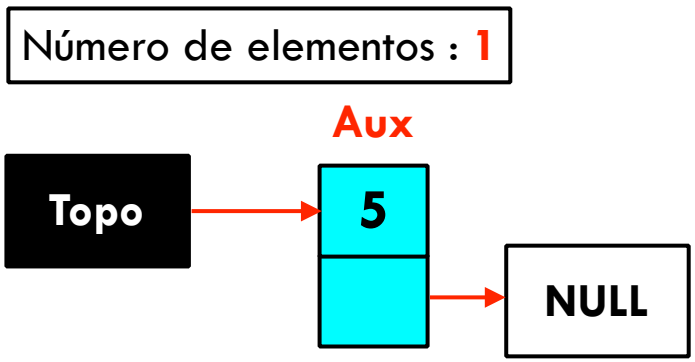
- a) primeira inserção (elemento x = 5)



Inserção (Push)

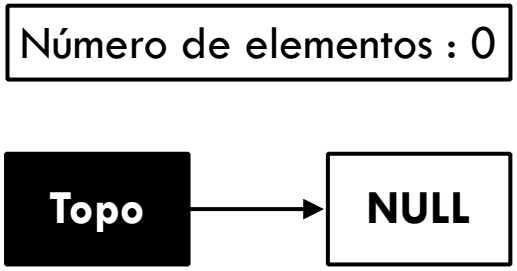
- a) primeira inserção (elemento x = 5)

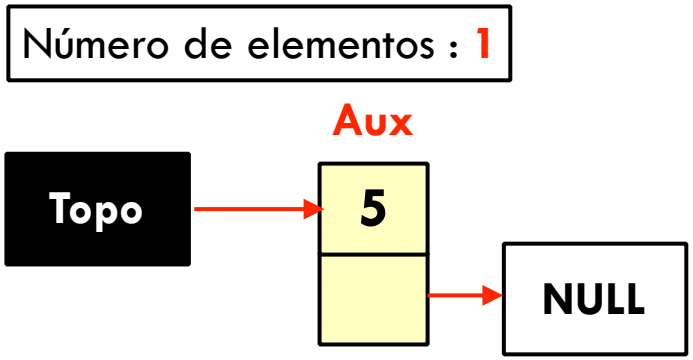


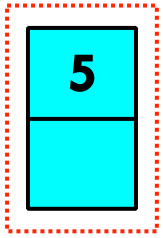


Inserção (Push)

- a) primeira inserção (elemento x = 5)







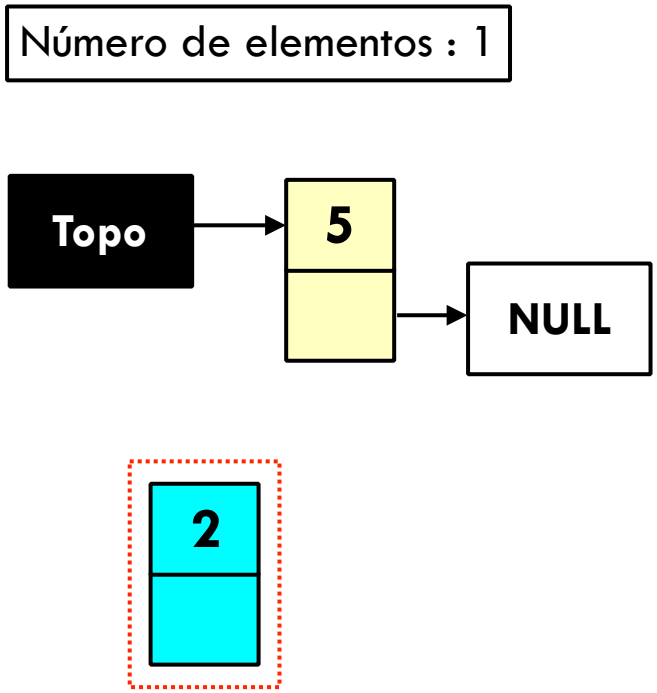
NoPilha (Aux)
alocado dinamicamente

O que aconteceu?

1. **Topo** passa a apontar para **Aux** (novo nó)
2. **Aux** aponta para quem o **Topo** apontava antes

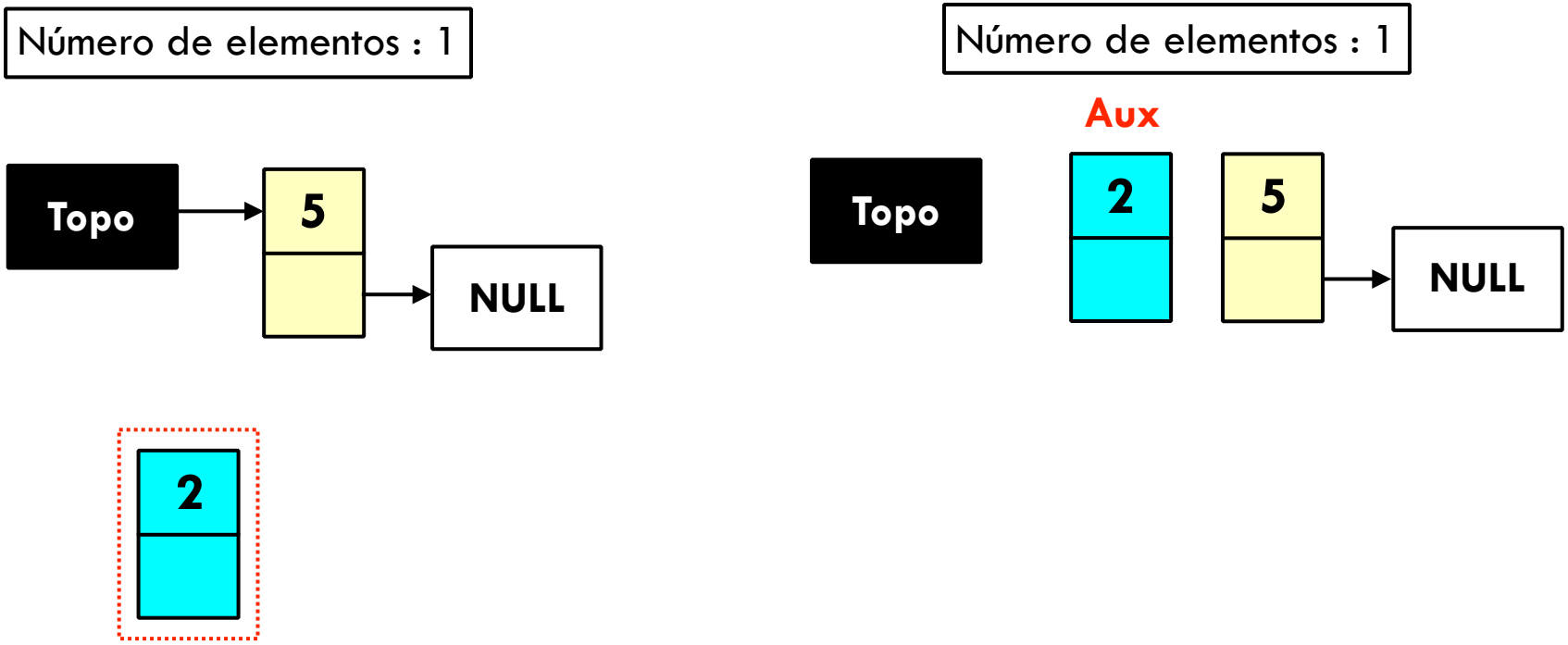
Inserção (Push)

- b) não é primeira inserção (elemento x = 2)



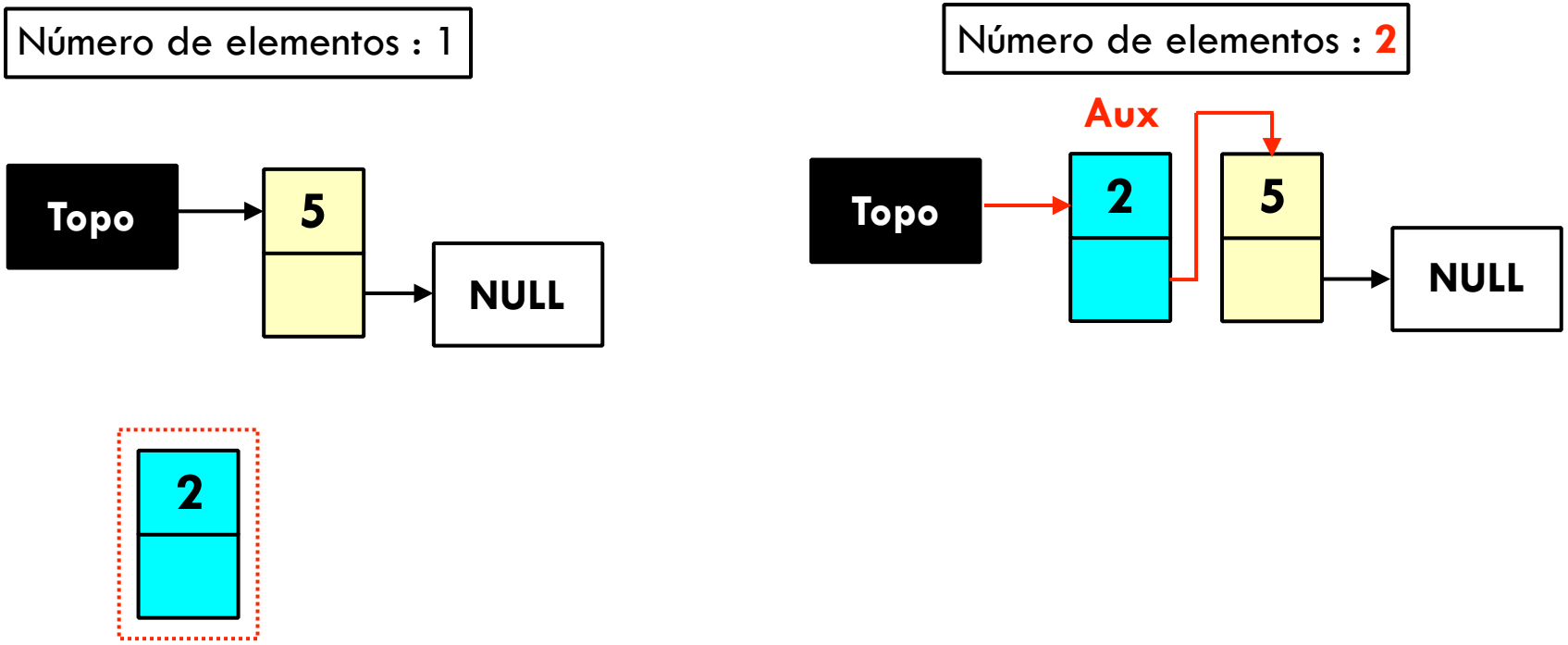
Inserção (Push)

- b) não é primeira inserção (elemento x = 2)



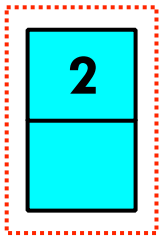
Inserção (Push)

- b) não é primeira inserção (elemento x = 2)



Inserção (Push)

- b) não é primeira inserção (elemento x = 2)



NoPilha (Aux)
alocado dinamicamente

O que aconteceu?

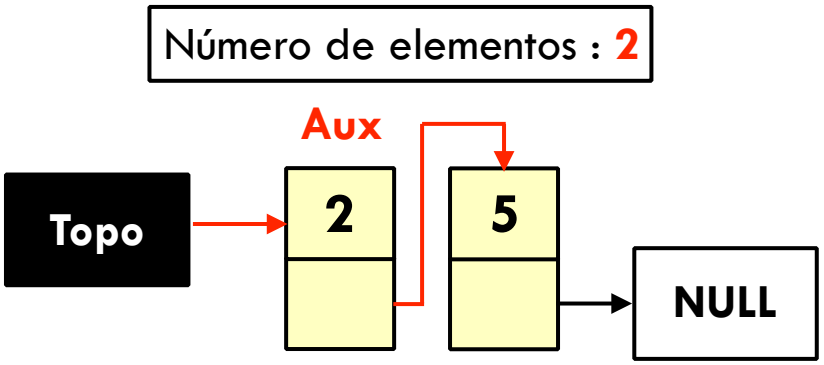
1. **Topo** passa a apontar para **Aux** (novo nó)
2. **Aux** aponta para quem o **Topo** apontava antes

Inserção (Push)

- b) não é primeira inserção (elemento x = 2)

O que aconteceu?

1. **Topo** passa a apontar para **Aux** (novo nó)
2. **Aux** aponta para quem o **Topo** apontava antes



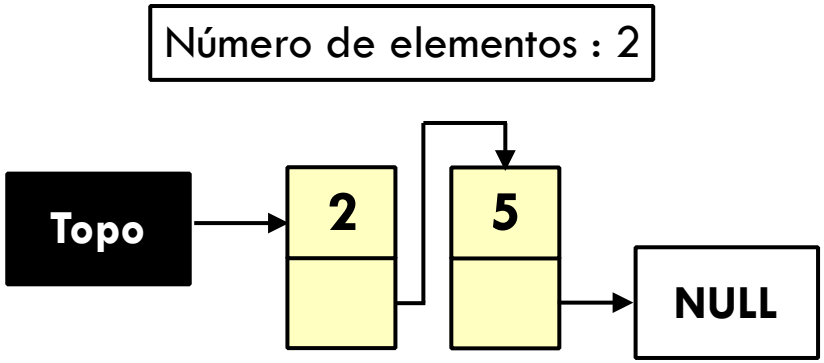
Inserção (Push)

- Pseudocódigo

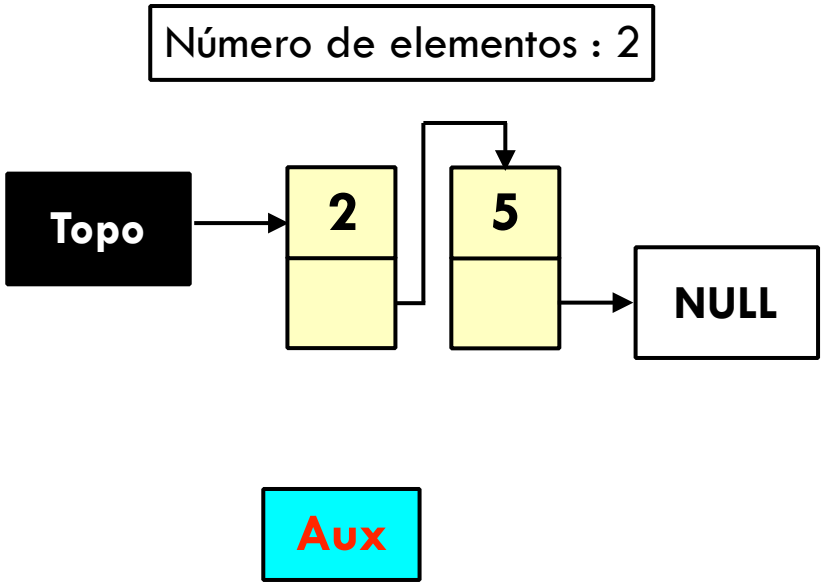
```
inserir/push (S, x)

1. criar um novo nó de pilha Aux // ponteiro NoPilha
2. alocar a memória do novo nó
3. Aux recebe o item a ser inserido
4. ponteiro de Aux aponta para quem o topo aponta
5. Topo aponta para o novo nó Aux
6. Incrementa a quantidade de elementos na pilha
```

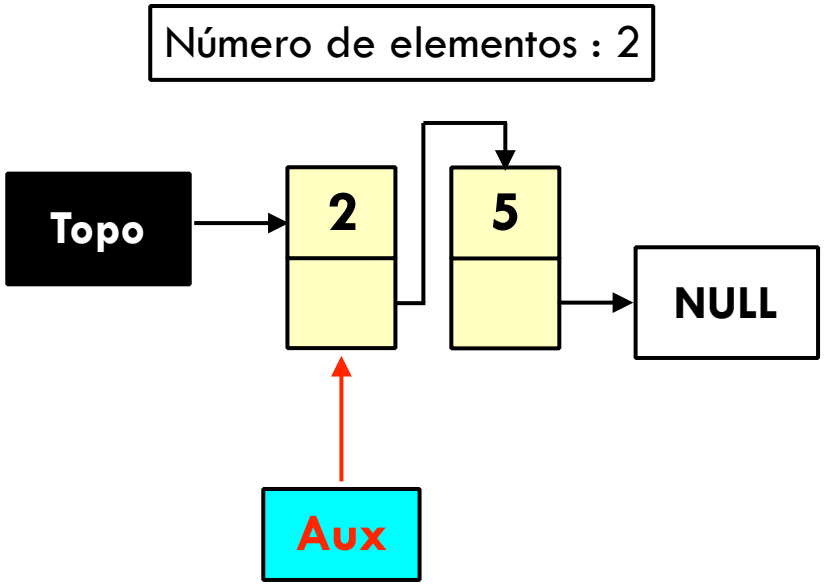
Remoção (Pop)



Remoção (Pop)

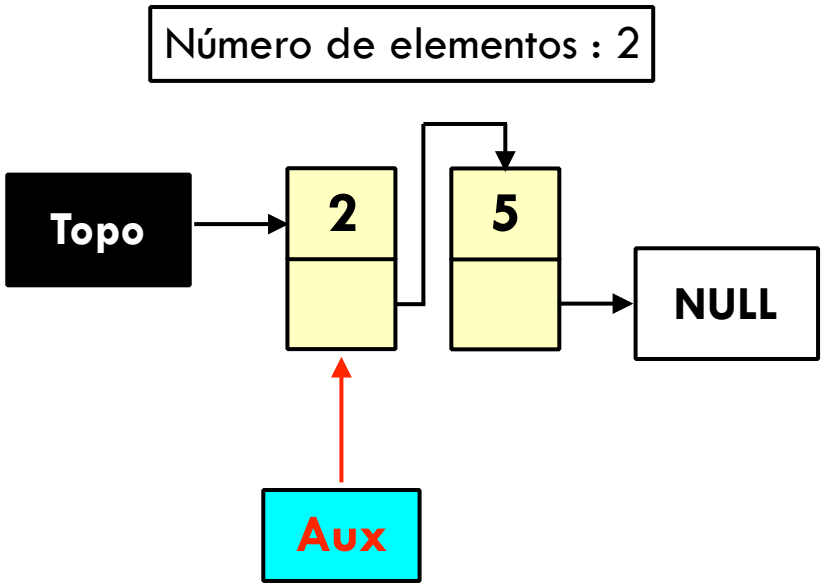

Número de elementos : 2
Topo
2
5
NULL
Aux
ponteiro para NoPilha

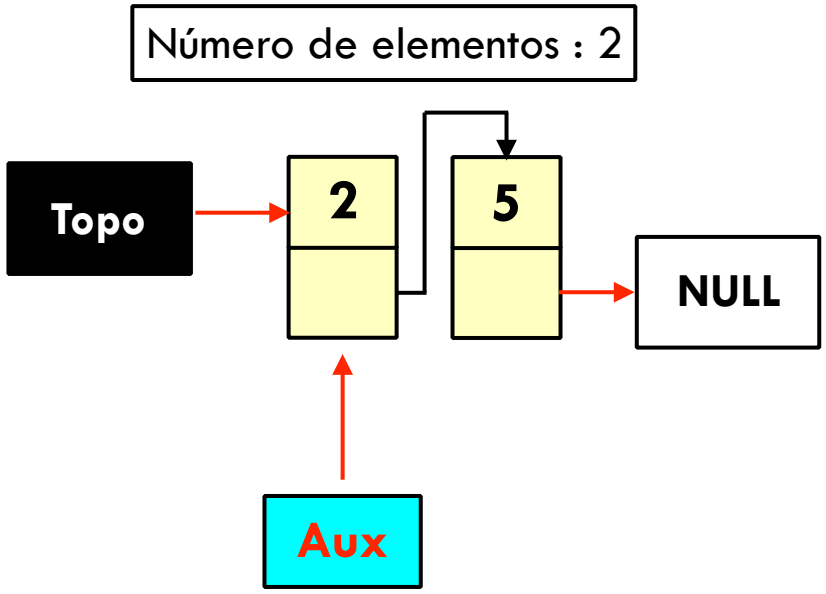
Remoção (Pop)



ponteiro para NoPilha

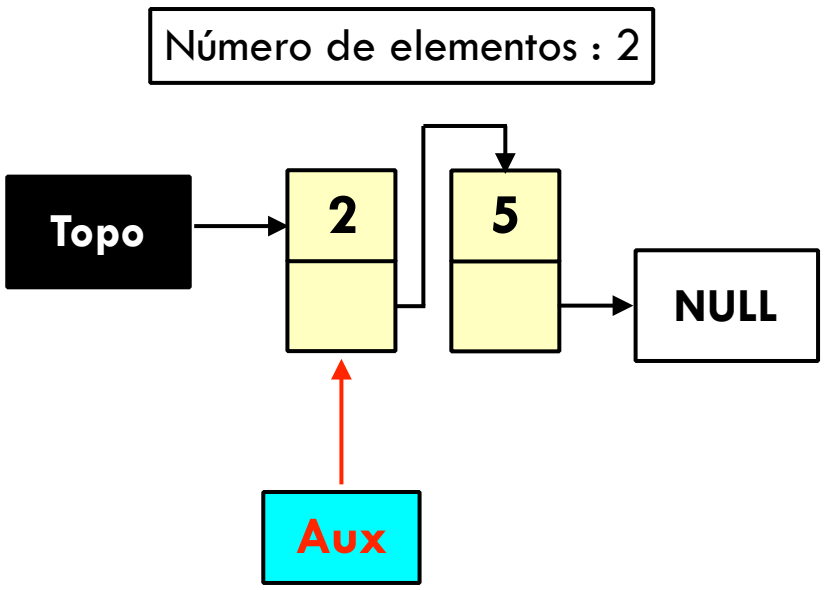
Remoção (Pop)

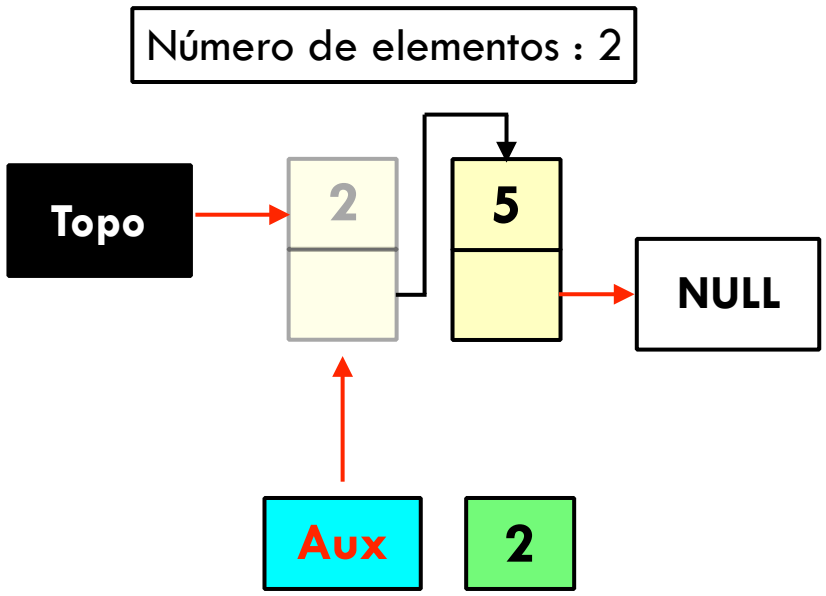




ponteiro para NoPilha

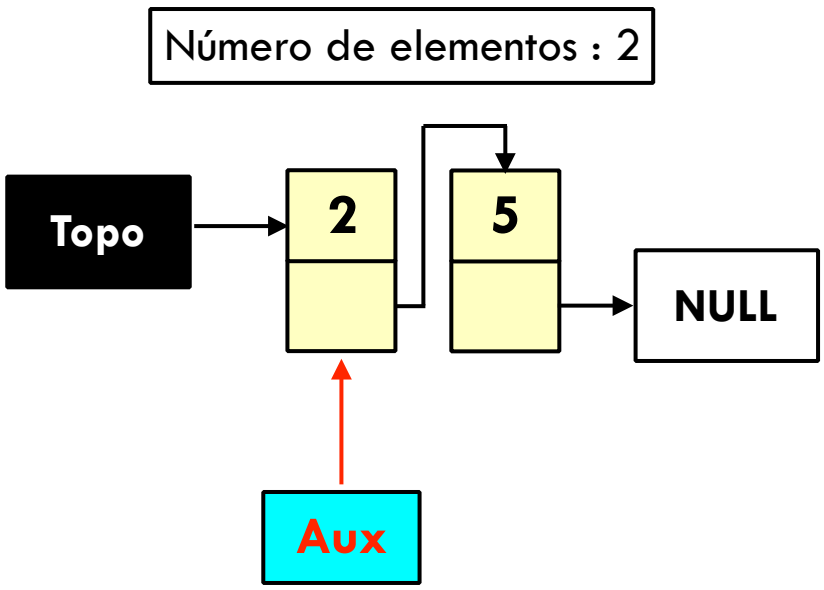
Remoção (Pop)

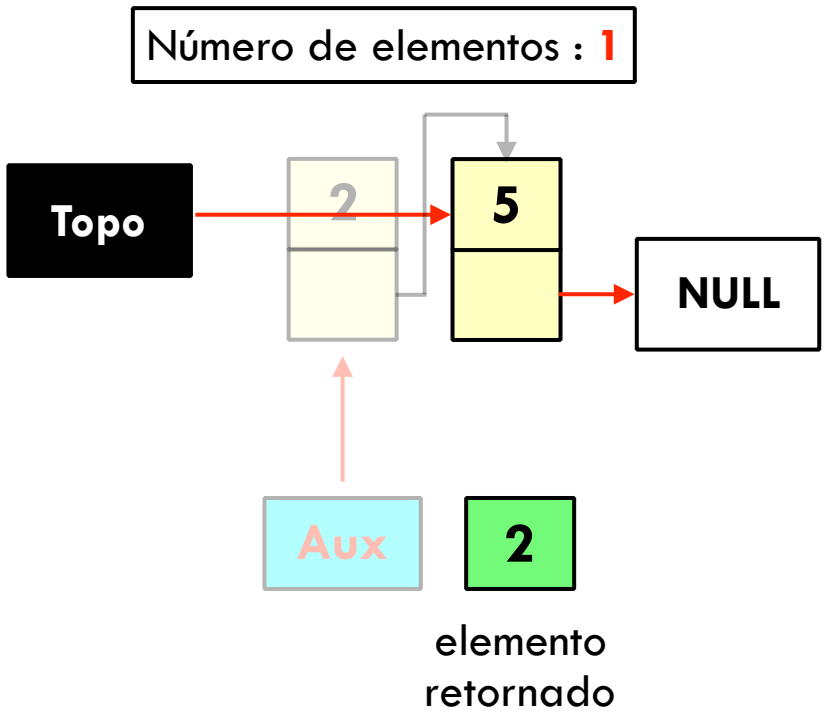




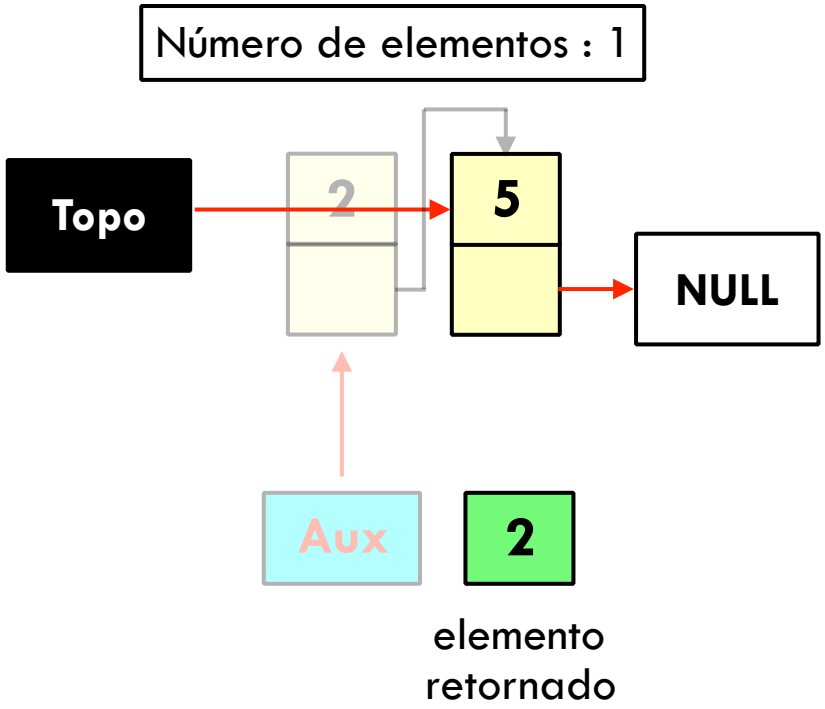
ponteiro para NoPilha

Remoção (Pop)


Número de elementos : 2
Topo
2
5
NULL
Aux
ponteiro para NoPilha


Número de elementos : 1
Topo
2
5
NULL
Aux
2
elemento
retornado

Remoção (Pop)



O que aconteceu?

1. Criamos um nó auxiliar para retornar deslocar a memória do elemento a ser removido
2. **Aux** aponta para quem o **Topo** aponta
3. **Topo** recebe o próximo nó do nó apontado por ele
4. Desalocamos a memória de **Aux**
5. Decrementamos a quantidade de itens na pilha
6. Retornamos o elemento removido

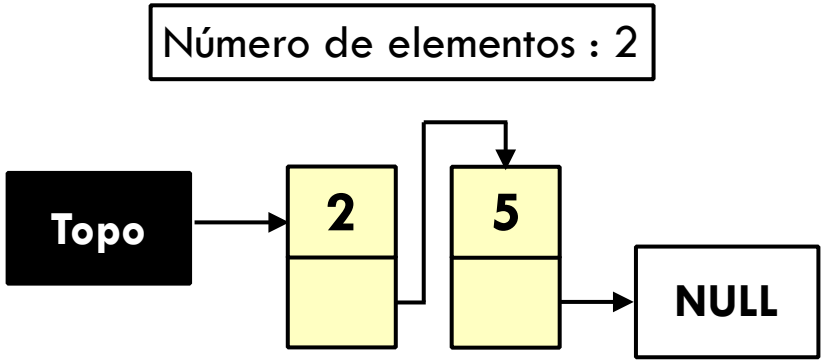
Remoção (Pop)

- Pseudocódigo

```
remover/pop (S, x)

 Se a pilha não estiver vazia
    1. criar um novo no de pilha Aux // ponteiro NoPilha
    2. recebe o conteúdo do elemento do Topo
    3. Aux aponta para o nó do Topo atual
    4. Topo recebe o próximo nó do nó atual do Topo
    5. Desalocamos e liberamos a memória de Aux
    6. Decrementamos a quantidade de elementos na pilha
    7. Retornamos o elemento x
```

Acessar topo (sem remoção)



- Pseudocódigo

```
Top (S, x)
1. x = S.topo.item;
2. return(x);
```

Exercício 01

- Ilustre cada estado de uma pilha dinâmica após realizar as seguintes operações (em ordem)
  - Push(S, 42)
  - Push(S, 13)
  - Push(S, 3)
  - Top(S)
  - Push(S, 85)
  - Pop(S)
  - Push(S, 16)
  - Pop(S)

- Considere que a pilha está inicialmente vazia

Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

Implementação (Dinâmica)

```
typedef struct {
    int key;
} Objeto;

typedef struct NoPilha *PtrNoPilha;

typedef struct NoPilha {
  Objeto obj;
  PtrNoPilha proximo;
} NoPilha;

typedef struct {
    PtrNoPilha topo;
    int tamanho;
} PilhaDinamica;
```

Implementação (Dinâmica)

```
typedef struct {
    int key;
} Objeto;

typedef struct NoPilha *PtrNoPilha;

typedef struct NoPilha {
  Objeto obj;
  PtrNoPilha proximo;
} NoPilha;

typedef struct {
    PtrNoPilha topo;
    int tamanho;
} PilhaDinamica;
```

implementa o nosso objeto

implementa o tipo que permite concatenar os nós dinâmicos

implementa os nós da pilha

implementa o TDA para Pilha

Implementação (Dinâmica)

```
typedef struct {
    int key;
} Objeto;

typedef struct NoPilha *PtrNoPilha;

typedef struct NoPilha {
  Objeto obj;
  PtrNoPilha proximo;
} NoPilha;

typedef struct {
    PtrNoPilha topo;
    int tamanho;
} PilhaDinamica;
```

implementa o nosso objeto

implementa o tipo que permite concatenar os nós dinâmicos

implementa os nós da pilha (estrutura recursiva) !!!

implementa o TDA para Pilha

Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. **Implementação com memória dinâmica**
5. Síntese / Revisão
6. Referências

Implementação (Dinâmica)

```
void iniciaPilha(PilhaDinamica *pilha);
void empilha(Item item, PilhaDinamica *pilha);
void desempilha(PilhaDinamica *pilha, Objeto *item);
void imprimePilha(PilhaDinamica *pilha);
bool estaVazia(PilhaDinamica *pilha);
bool tamanhoPilha(PilhaDinamica *pilha);
Objeto topo(PilhaDinamica *pilha);
```

Exercício 02

- Mãos a obra: implemente um TDA para Pilha com alocação dinâmica, e suas funções de manipulação.

Implementação (Dinâmica)

```
void iniciaPilha(PilhaDinamica *p) {
  p->topo = NULL;
  p->tamanho = 0;
}
```

```
void empilha(PilhaDinamica *p) {
  PtrNoPilha aux;
  aux = (PtrNoPilha) malloc(sizeof(NoPilha));

  /* atualiza os ponteiros */

  p->tamanho++;
}
```

Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

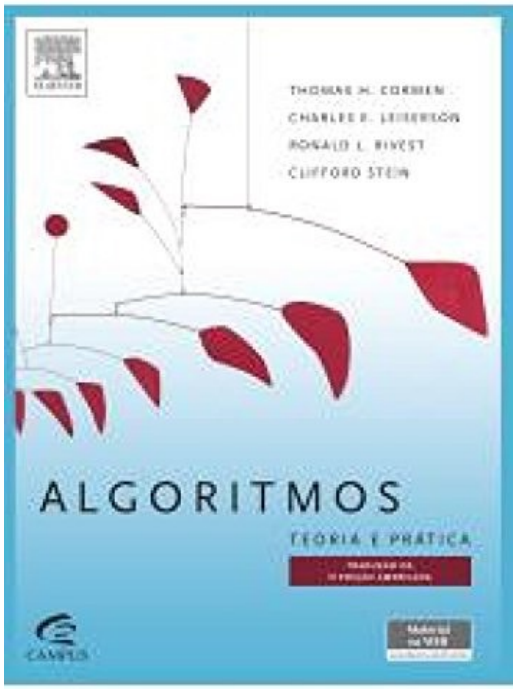
Próximas Aulas

- Filas
  - estáticas
  - dinâmicas
- Implementação de Listas Lineares
  - single-linked
  - double-linked

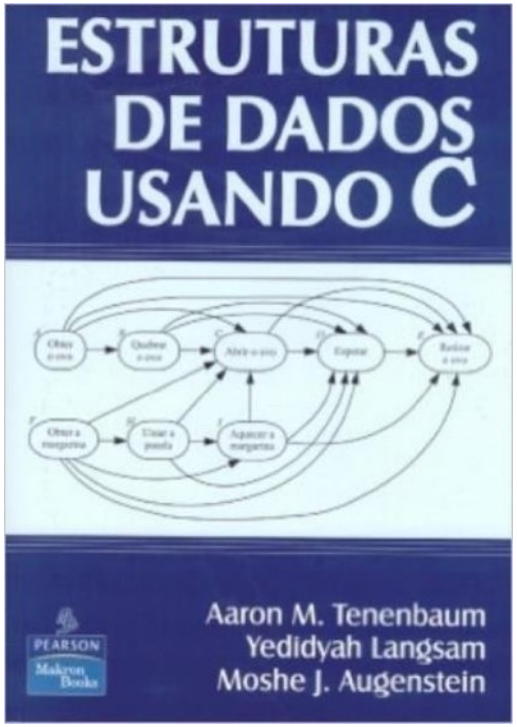
Roteiro

1. Introdução
2. Operações
3. Tipo abstrato (typedef)
4. Implementação com memória dinâmica
5. Síntese / Revisão
6. Referências

Referências sugeridas

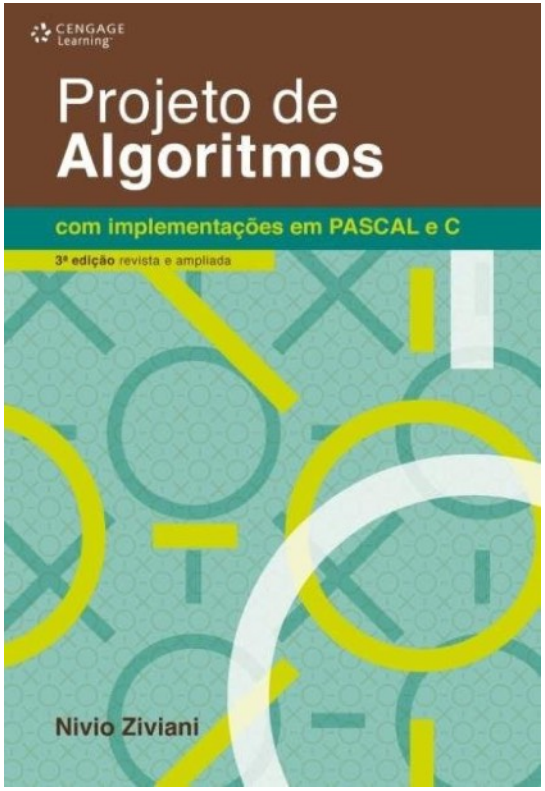


[Cormen et al, 2018]



[Tenenbaum et al, 1995]

Referências sugeridas



[Ziviani, 2010]

[Drozdek, 2017]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br