

ED62A-COM2A

ESTRUTURAS DE DADOS

Aula 03b - Pilhas
(Implementação dinâmica)

Prof. Rafael G. Mantovani

02/04/2019

Roteiro

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

Roteiro

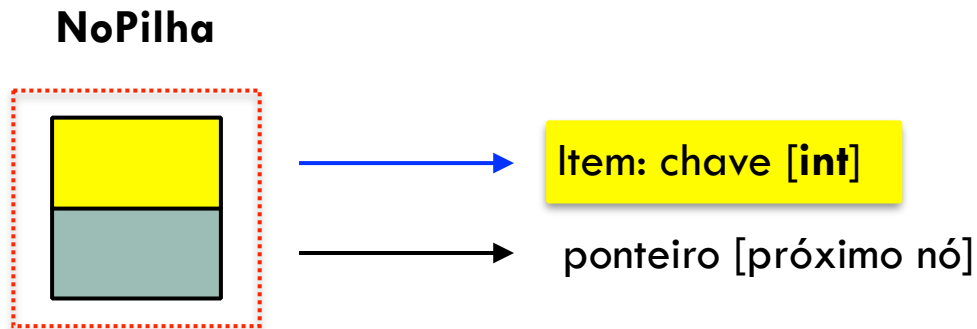
- 1** Introdução
- 2** Operações
- 3** Tipo abstrato (typedef)
- 4** Implementação com memória dinâmica
- 5** Síntese / Revisão
- 6** Referências

Introdução

- Pilhas Estáticas
 - array de N elementos
 - variável que controla o índice do topo
- Pilhas dinâmicas
 - elementos do tipo NoPilha
 - ponteiros
 - topo é um ponteiro para NoPilha

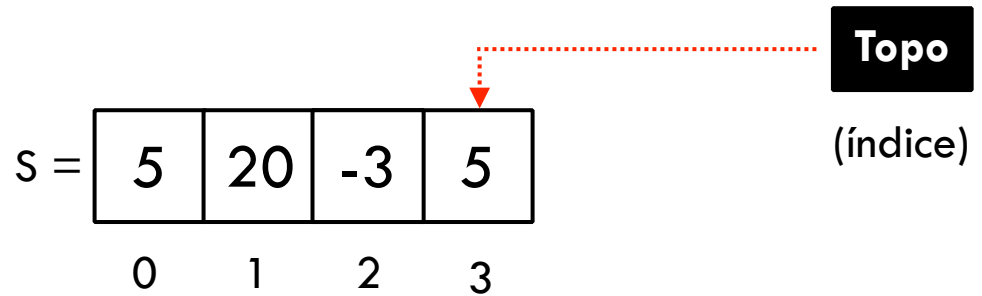
Introdução

- Elemento (objeto) → vários atributos



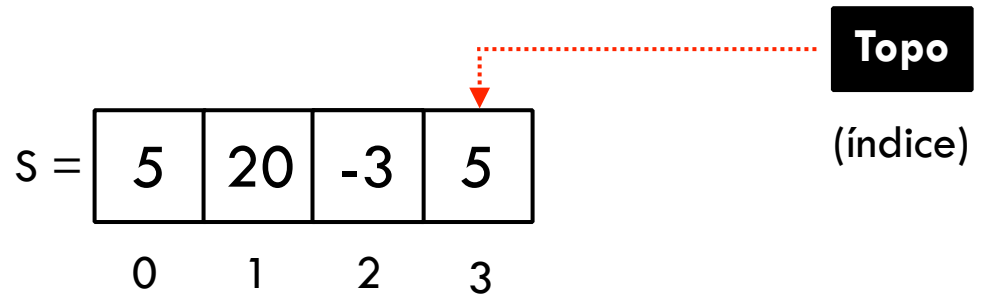
Introdução

* Pilha estática

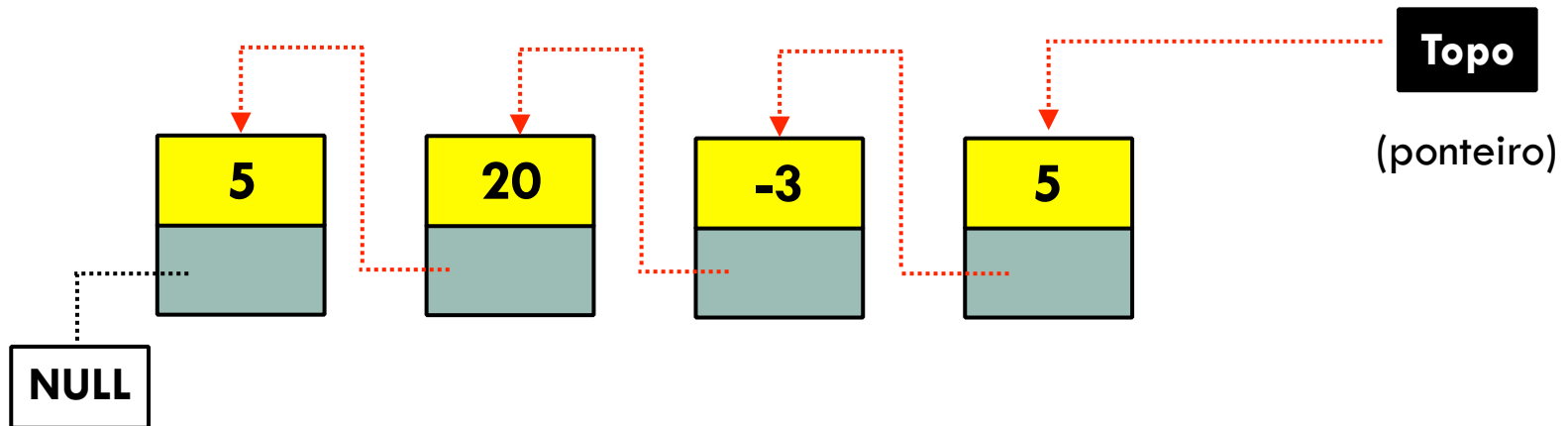


Introdução

* Pilha estática



* Pilha dinâmica



Roteiro

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

Operações

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

- `pesquisar(S, k)` → procurar **k** em **S** [TRUE/FALSE]
- `inserir(S, k)` → inserir **k** em **S**
- `remover(S, k)` → remover **k** em **S**
- `minimo(S)` → menor valor armazenado em **S**
- `maximo(X)` → maior valor armazenado em **S**
- `proximo(S, x)` → elemento sucessor a **x**
- `anterior(S, x)` → elemento antecessor a **x**
- `tamanho(S)` → tamanho de **S**
- `vazia(S)` → **S** está vazia? [TRUE/FALSE]
- `cheia(S)` → **S** está cheia? [TRUE/FALSE]

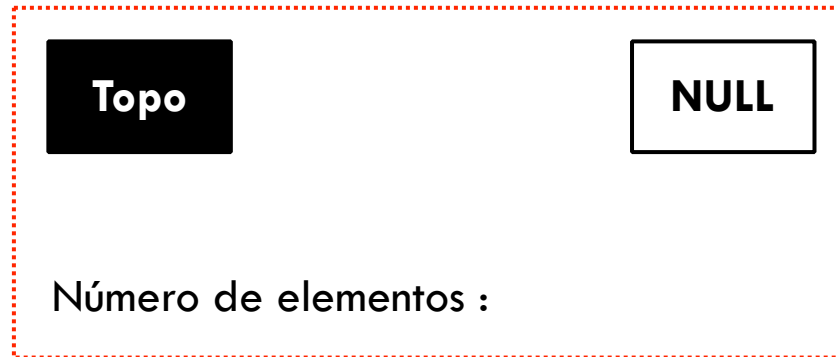
Operações

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

- ~~pesquisar(S, k) → procurar k em S [TRUE/FALSE]~~
- **inserir(S, k) → inserir k em S**
- **remover(S, k) → remover k em S**
- ~~minimo(S) → menor valor armazenado em S~~
- ~~maximo(X) → maior valor armazenado em S~~
- ~~proximo(S, x) → elemento sucessor a x~~
- ~~anterior(S, x) → elemento antecessor a x~~
- **tamanho(S) → tamanho de S**
- **vazia(S) → S está vazia? [TRUE/FALSE]**
- ~~cheia(S) → S está cheia? [TRUE/FALSE]~~
- **topo(S) → retorna o elemento do topo, sem remover**

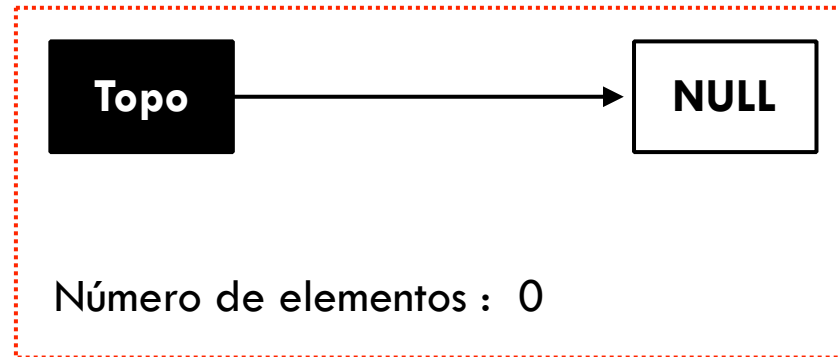
Inicialização da pilha

tipo Pilha Dinâmica



Inicialização da pilha

tipo Pilha Dinâmica

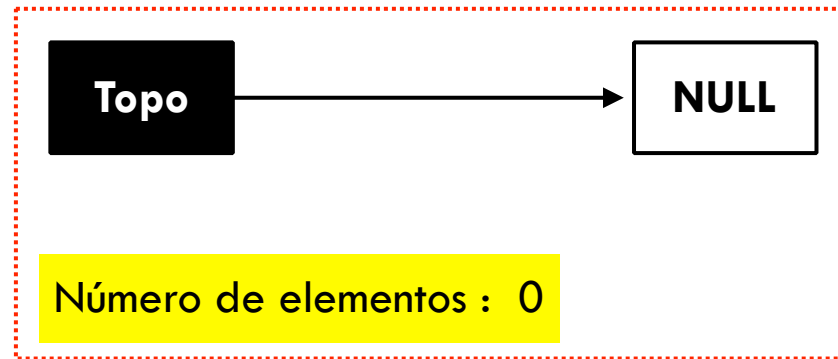


IniciaPilha (S)

1. S.topo = NULL;
2. S.tamanho = 0;

Tamanho da Pilha

tipo Pilha Dinâmica

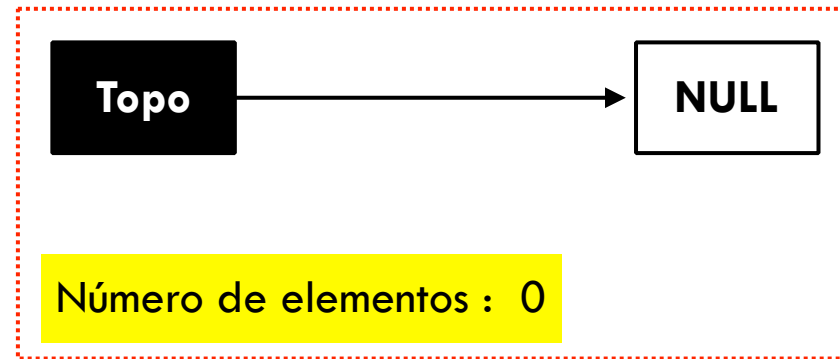


tamanhoPilha (S)

estaVazia (S)

Tamanho da Pilha

tipo Pilha Dinâmica



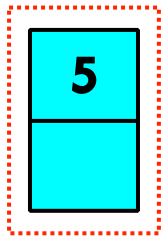
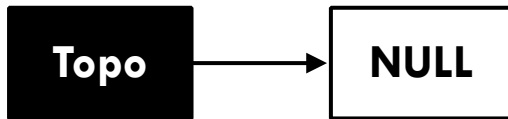
tamanhoPilha (S)
1. return (S.tamanho);

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

Inserção (Push)

- a) primeira inserção (elemento $x = 5$)

Número de elementos : 0

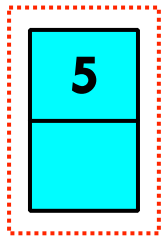
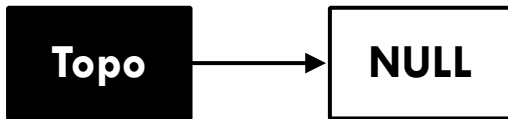


NoPilha
(Aux)

Inserção (Push)

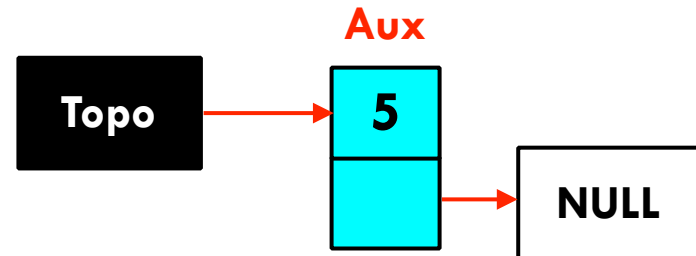
- a) primeira inserção (elemento $x = 5$)

Número de elementos : 0



NoPilha
(Aux)

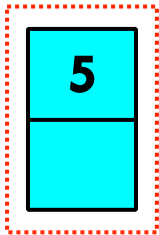
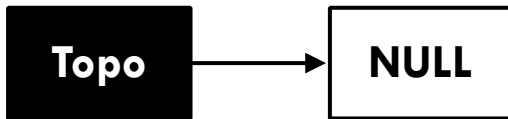
Número de elementos : 1



Inserção (Push)

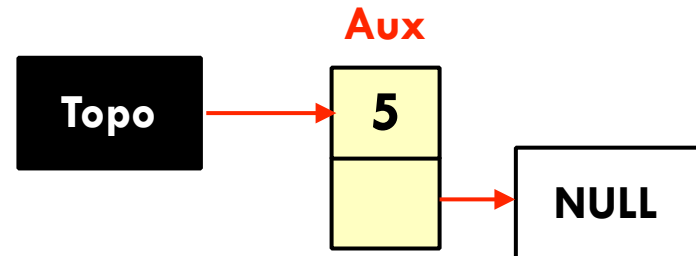
- a) primeira inserção (elemento $x = 5$)

Número de elementos : 0



NoPilha (Aux)
alocado dinamicamente

Número de elementos : 1



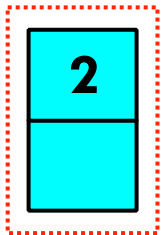
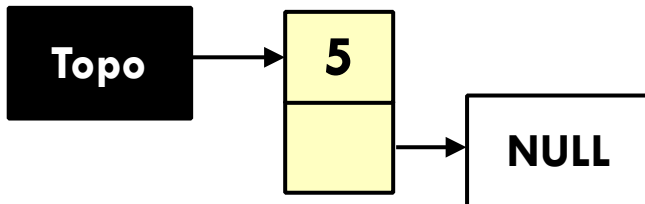
O que aconteceu?

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 $x = 2$)

Número de elementos : 1

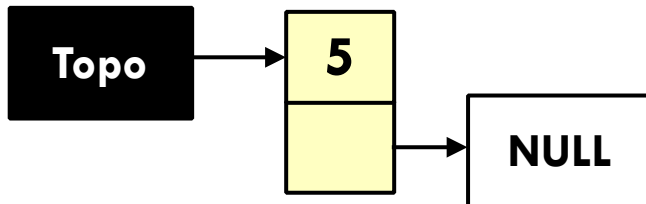


NoPilha (Aux)
alocado dinamicamente

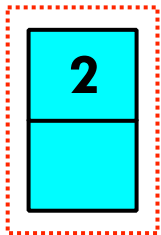
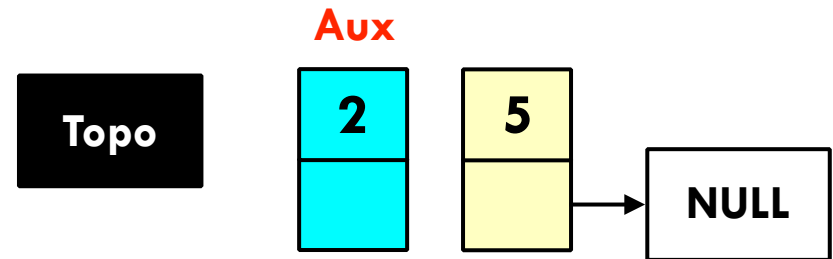
Inserção (Push)

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

Número de elementos : 1



Número de elementos : 1

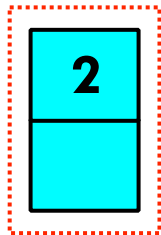
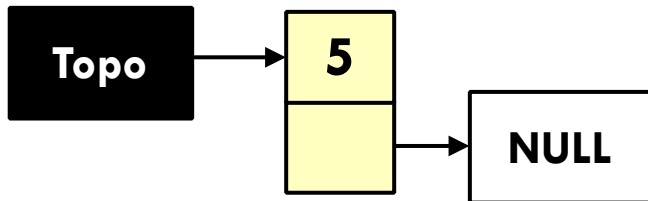


NoPilha (Aux)
alocado dinamicamente

Inserção (Push)

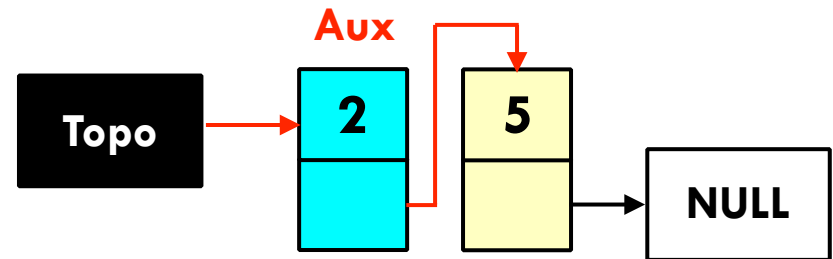
- b) não é primeira inserção (elemento $x = 2$)

Número de elementos : 1



NoPilha (Aux)
alocado dinamicamente

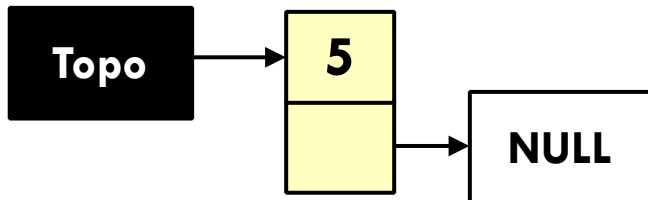
Número de elementos : 2



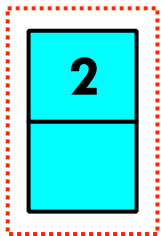
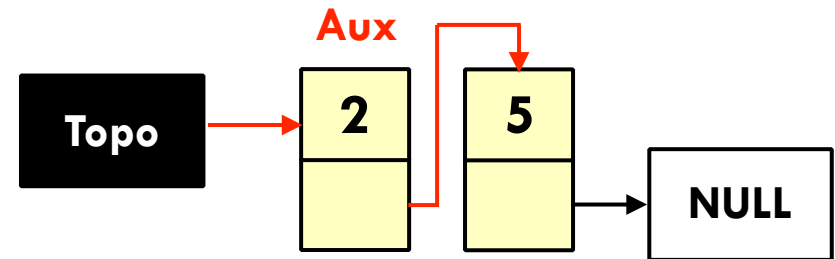
Inserção (Push)

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

Número de elementos : 1



Número de elementos : 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

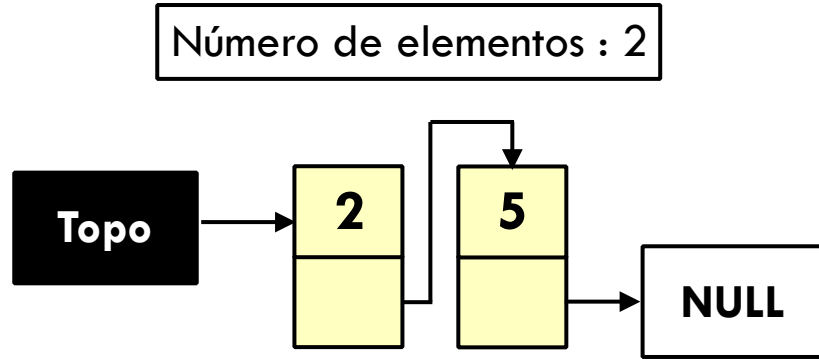
Inserção (Push)

- Pseudocódigo

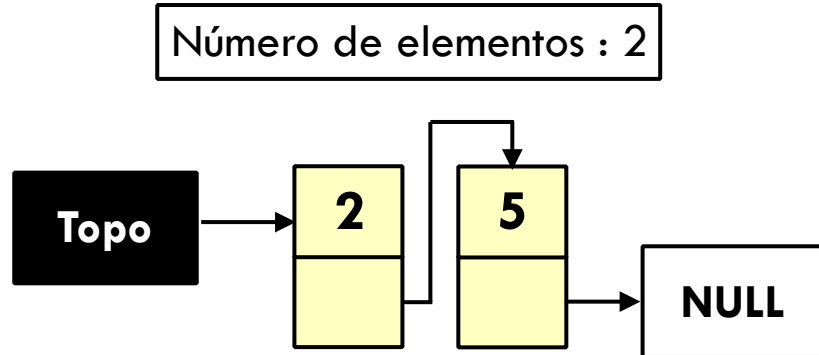
`inserir/push (S, x)`

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

Remoção (Pop)



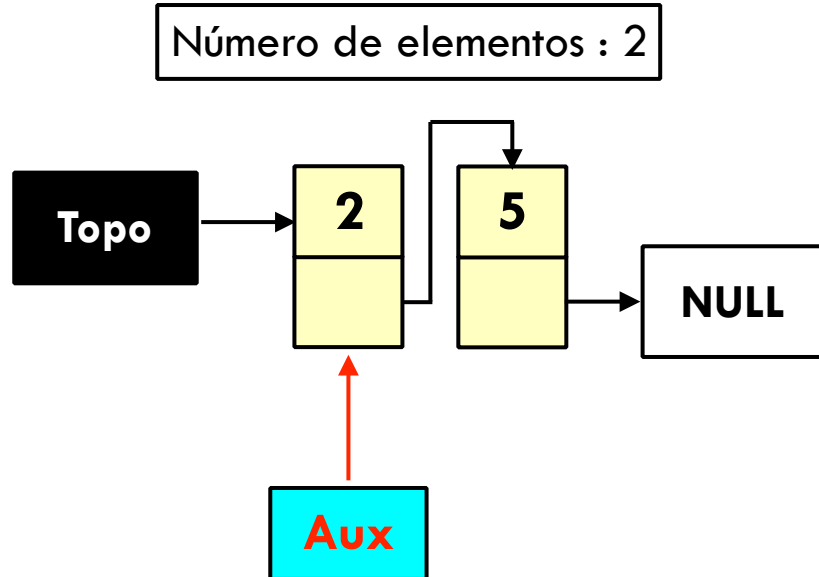
Remoção (Pop)



Aux

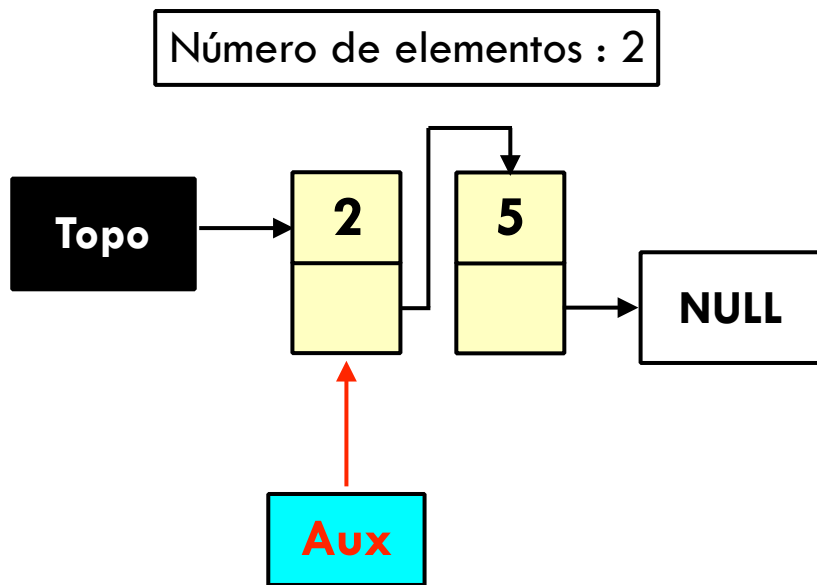
ponteiro para NoPilha

Remoção (Pop)

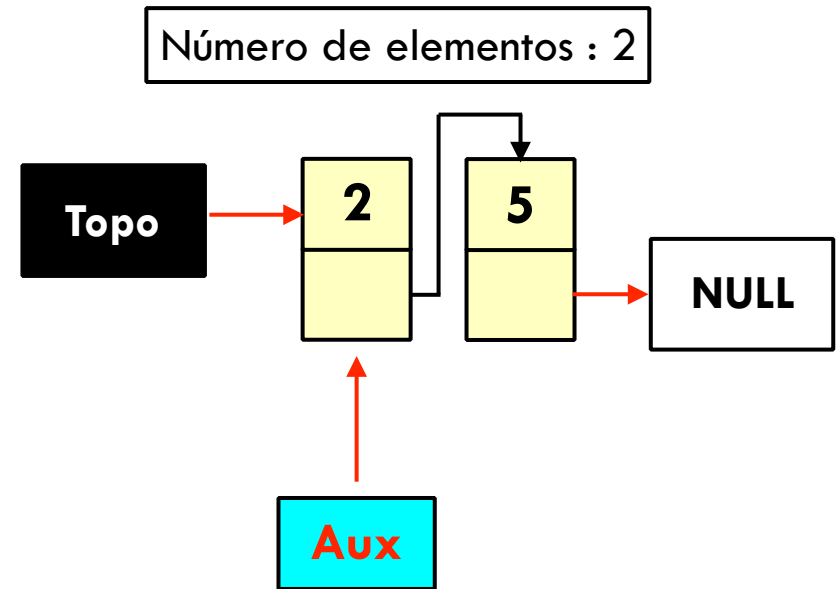


ponteiro para NoPilha

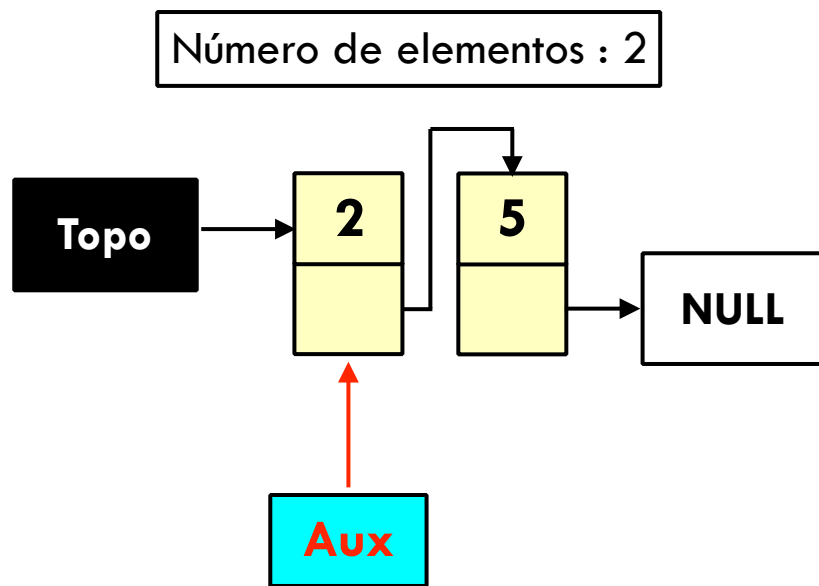
Remoção (Pop)



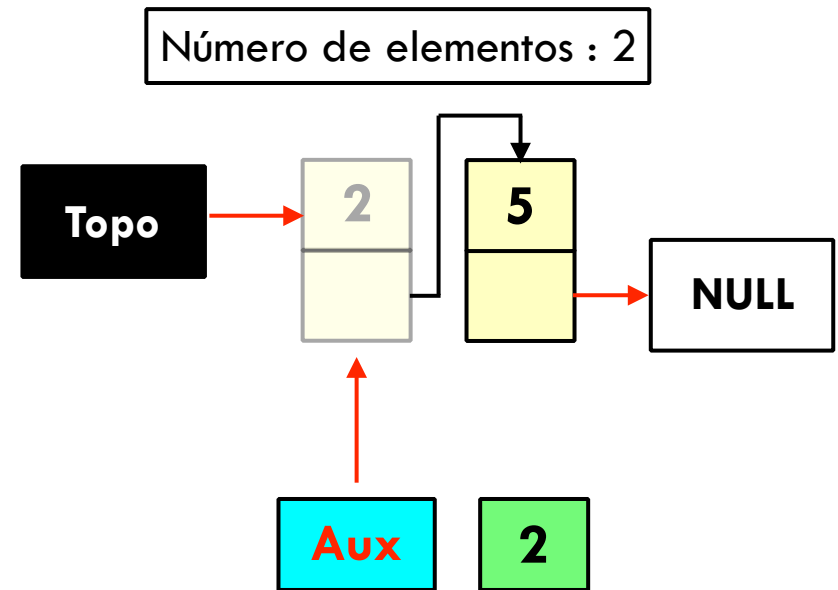
ponteiro para NoPilha



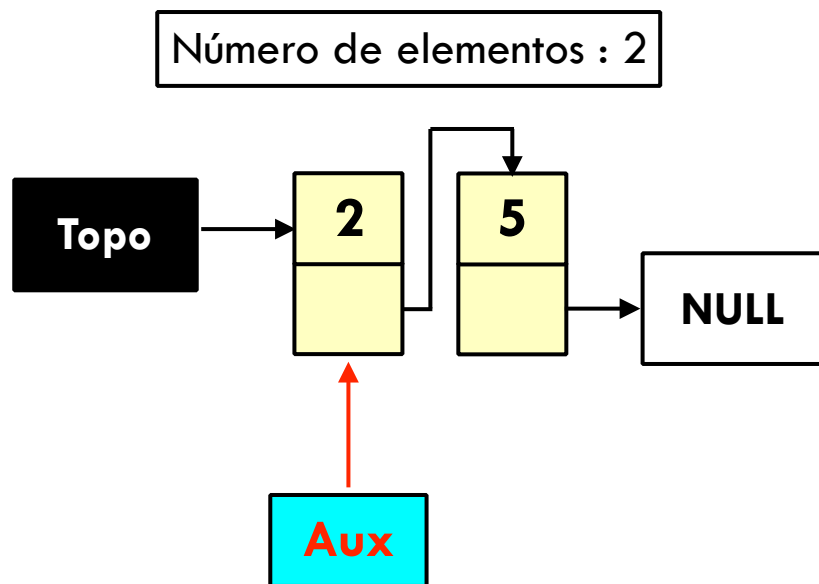
Remoção (Pop)



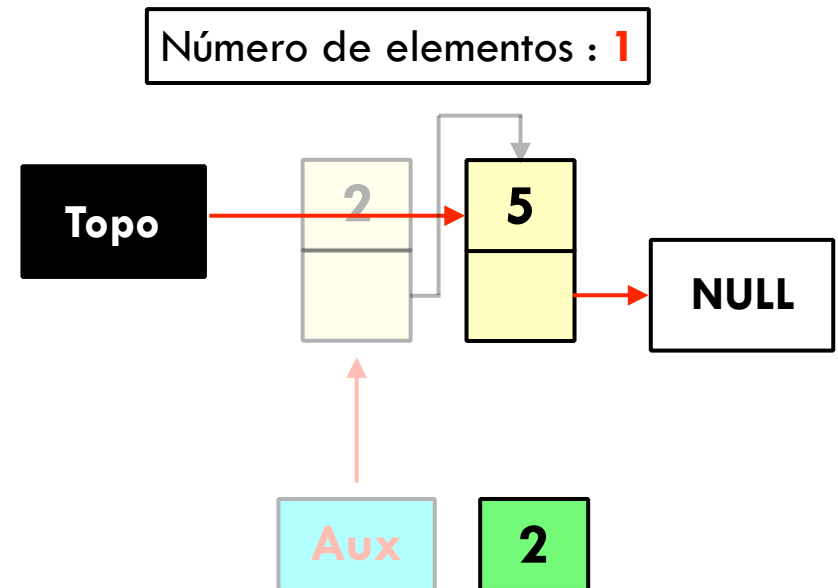
ponteiro para NoPilha



Remoção (Pop)

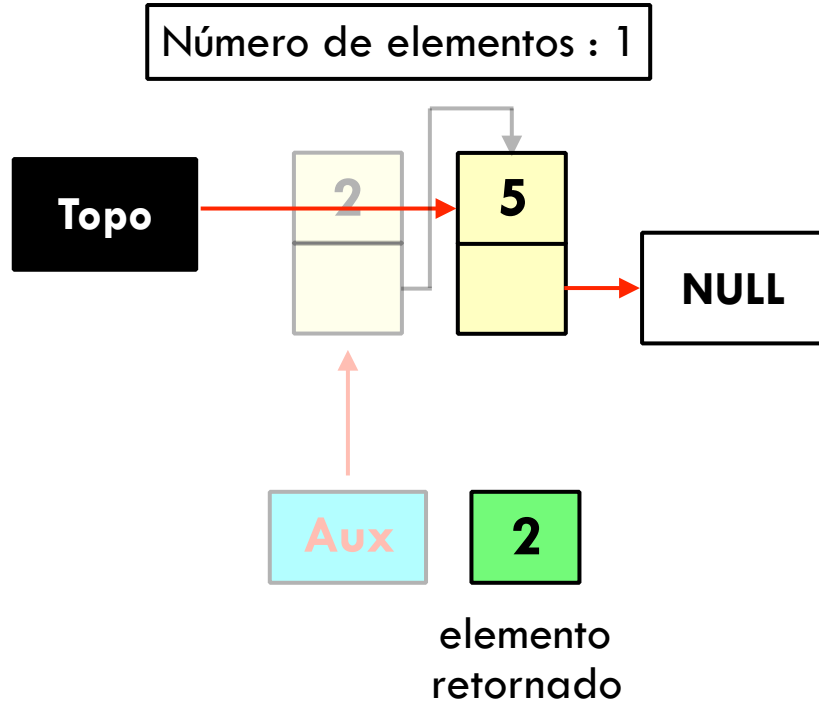


ponteiro para NoPilha



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)

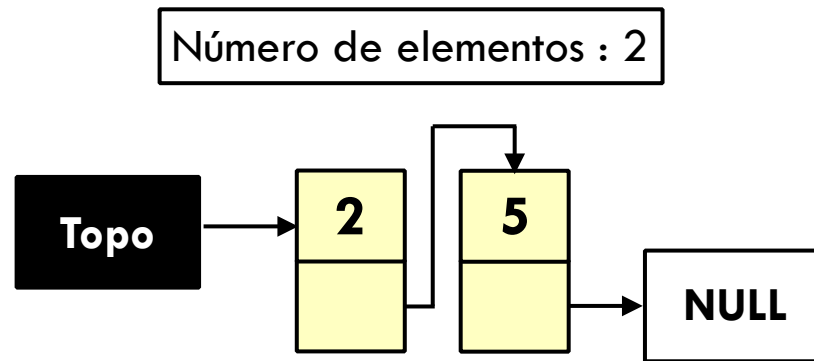
- Pseudocódigo

remover/pop (S, x)

Se a pilha não estiver vazia

1. criar um novo nó 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)
 - Pop(S)
 - Push(S, 85)
 - Pop(S)
 - Push(S, 16)
- 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;  
} Item;  
  
typedef struct NoPilha *Ponteiro;  
  
typedef struct NoPilha {  
    Item topo;  
    Ponteiro proximo;  
} NoPilha;  
  
typedef struct {  
    Ponteiro topo;  
    int tamanho;  
} pilhaDinamica;
```

Implementação (Dinâmica)

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

implementa o nosso
objeto

```
typedef struct NoPilha *Ponteiro;
```

implementa o tipo que permite
concatenar os nós dinâmicos

```
typedef struct NoPilha {  
    Item topo;  
    Ponteiro proximo;  
} NoPilha;
```

implementa os nós da pilha

```
typedef struct {  
    Ponteiro topo;  
    int tamanho;  
} pilhaDinamica;
```

implementa o TDA
para Pilha

Implementação (Dinâmica)

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

implementa o nosso
objeto

```
typedef struct NoPilha *Ponteiro;
```

implementa o tipo que permite
concatenar os nós dinâmicos

```
typedef struct NoPilha {  
    Item topo;  
    Ponteiro proximo;  
} NoPilha;
```

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

```
typedef struct {  
    Ponteiro topo;  
    int tamanho;  
} pilhaDinamica;
```

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, Item *item);  
void imprimePilha(pilhaDinamica *pilha);  
int estaVazia(pilhaDinamica *pilha);  
int tamanhoPilha(pilhaDinamica *pilha);  
Item 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 *pilha) {  
    pilha->topo = NULL;  
    pilha->tamanho = 0;  
}
```

```
void empilha(pilhaDinamica *pilha) {  
    Ponteiro aux;  
    aux = (Ponteiro) malloc(sizeof(NoPilha));  
  
    /* atualiza os ponteiros */  
  
    pilha->tamanho++;  
}
```


Roteiro

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

Próximas Aulas

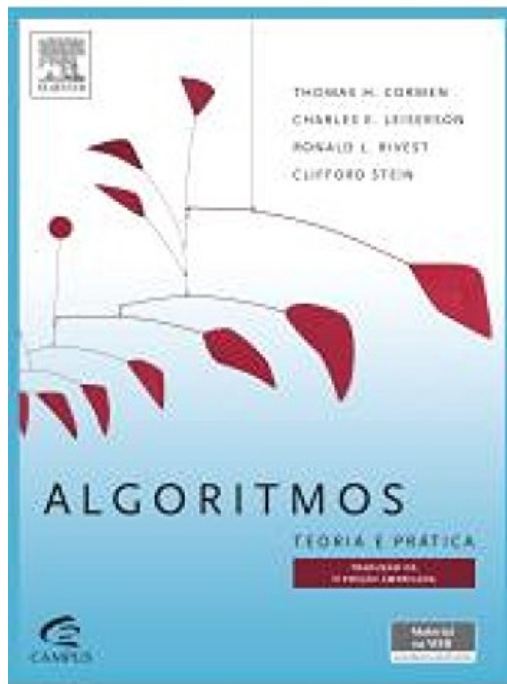


- Filas
 - estáticas
 - dinâmicas
- Implementação de Listas Lineares
 - single-linked
 - double-linked

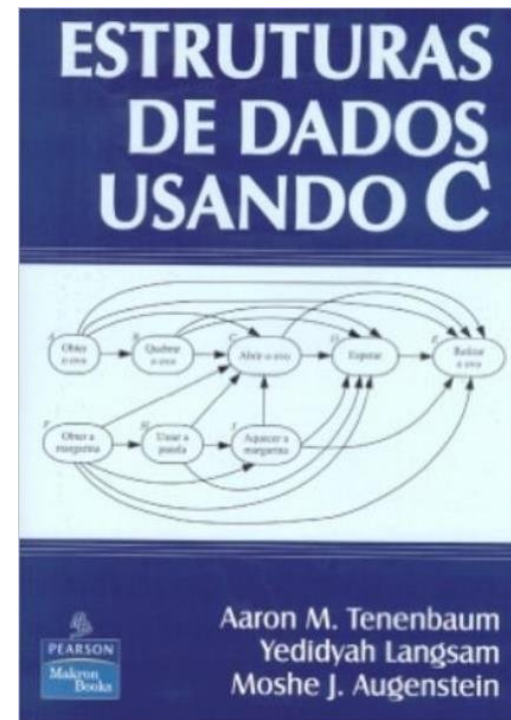
Roteiro

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

Referências sugeridas

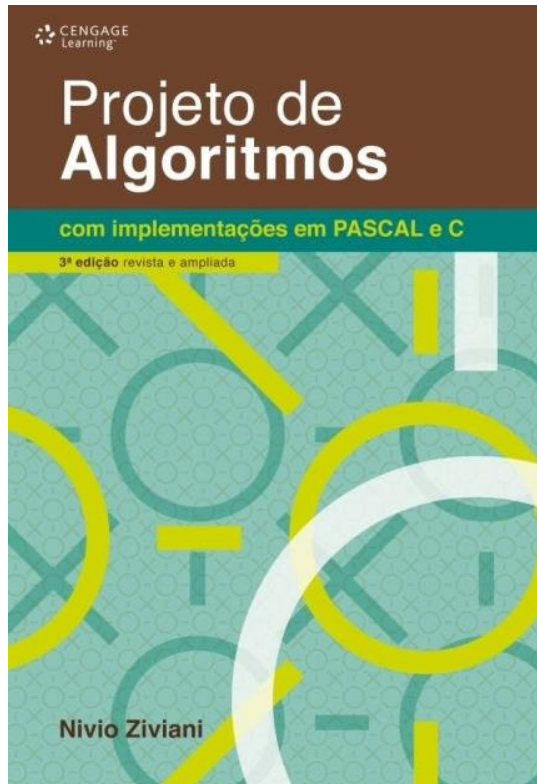


[Cormen et al, 2018]



[Tenenbaum et al, 1995]

Referências sugeridas



[Ziviani, 2010]



[Drozdek, 2017]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br