ED62A-COM2A
ESTRUTURAS DE DADOS

Aula 03 - Pilhas
(Implementação estática)

Prof. Rafael G. Mantovani

26/03/2019

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil
Março, 2019

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

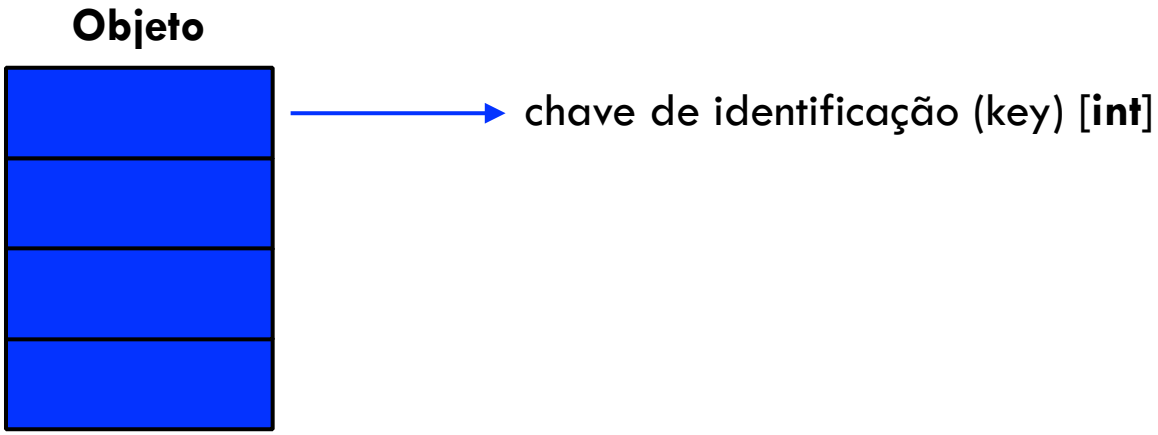
Introdução

- Conjuntos são fundamentais para Computação / Matemática
  - Matemática = conjuntos são invariáveis
  - Computação = são dinâmicos

- Operações
  - diferentes operações em um conjunto
  - inserir, eliminar e testar a existência de um elemento
  - a melhor forma de implementar depende das operações

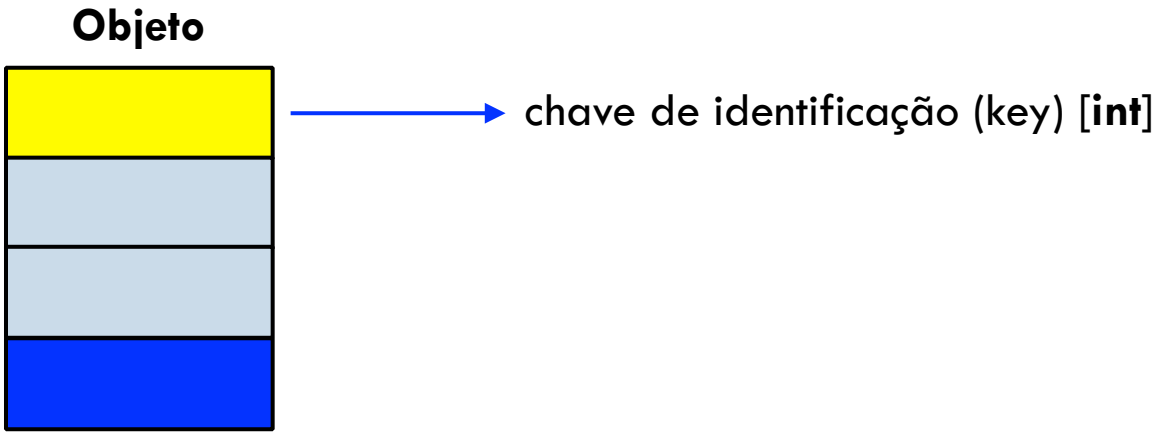
Introdução

- Elemento (objeto) → vários atributos



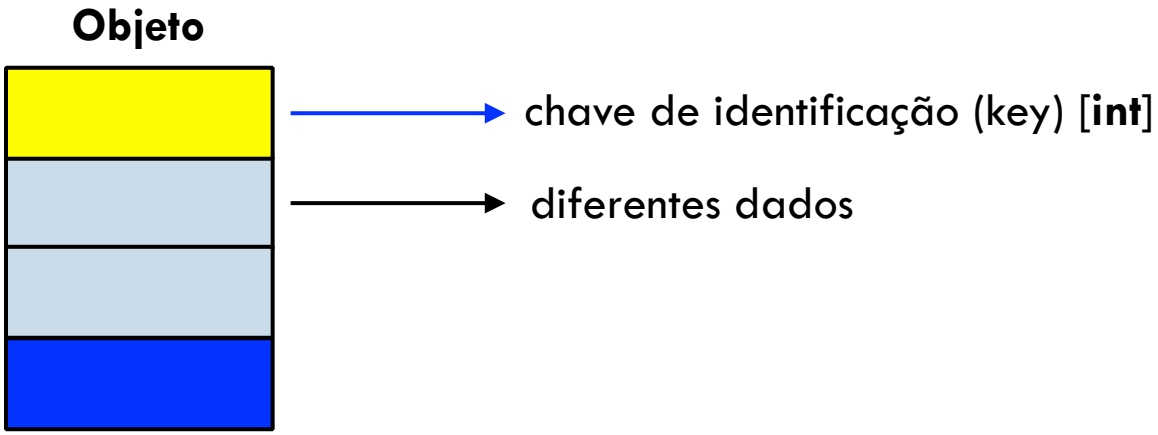
Introdução

- Elemento (objeto) → vários atributos



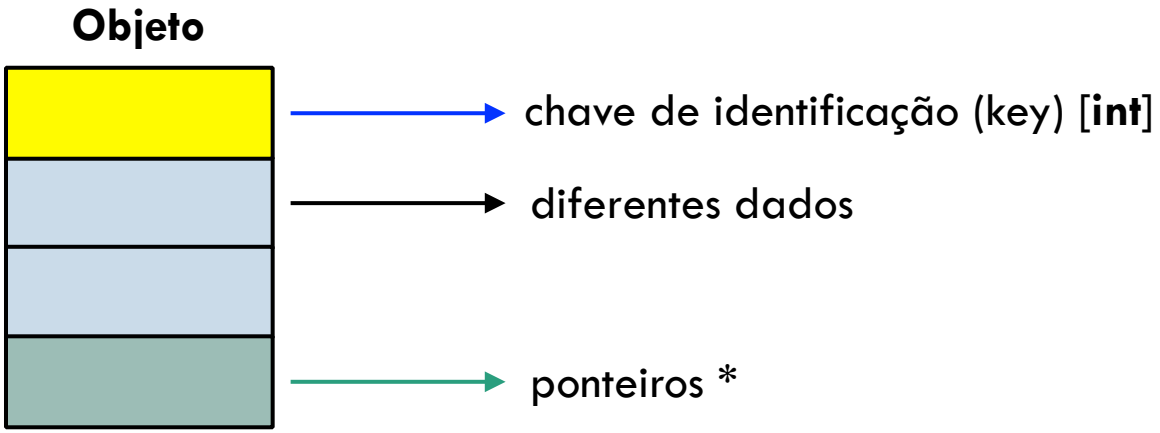
Introdução

- Elemento (objeto) → vários atributos



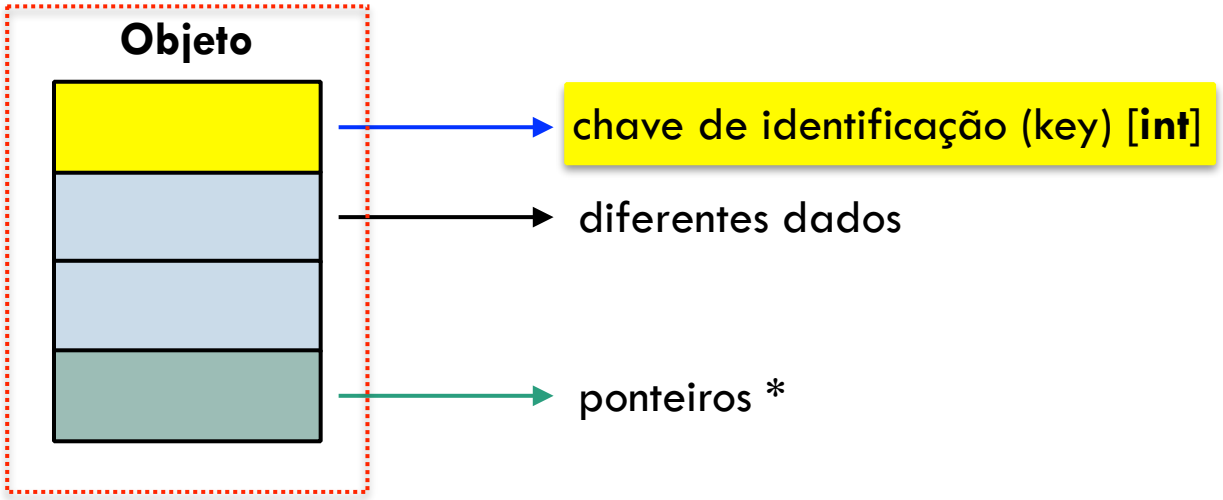
Introdução

- Elemento (objeto) → vários atributos



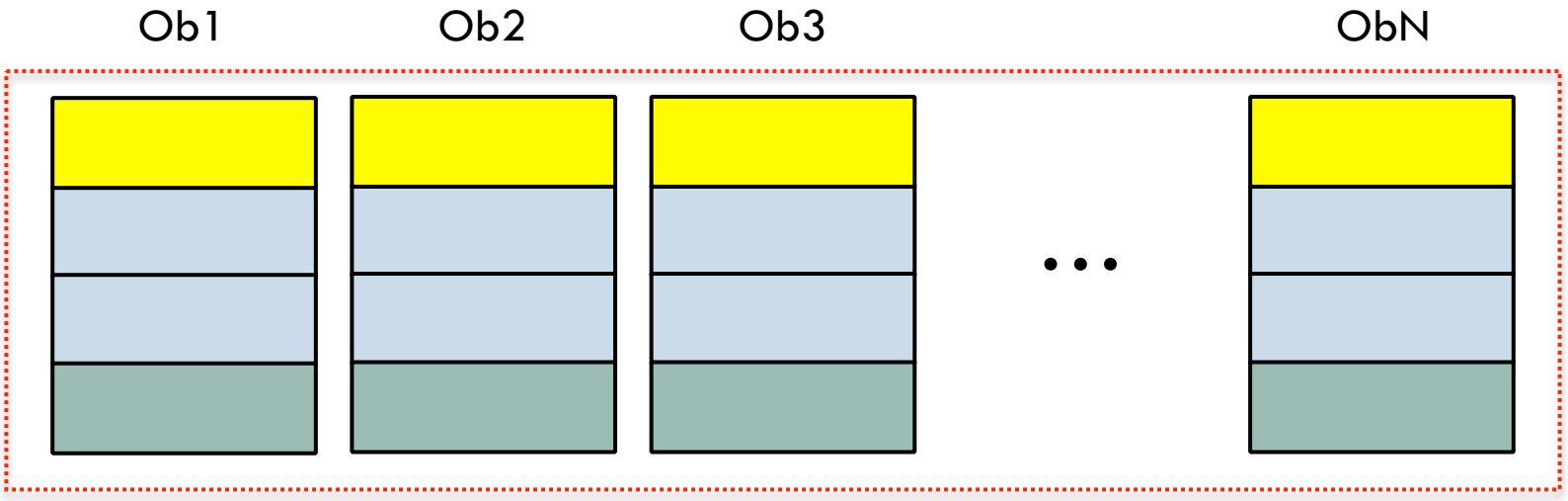
Introdução

- Elemento (objeto) → vários atributos



Introdução

- Estrutura = Arranjo de N Objetos



Estrutura de Dados

Operações

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

- search(S, k) → procurar k em S [TRUE/FALSE]
- insert(S, k) → inserir k em S
- delete(S, k) → remover k em S
- minimum(S) → menor valor armazenado em S
- maximum(X) → maior valor armazenado em S
- next(S, x) → elemento sucessor a x
- previous(S, x) → elemento antecessor a x
- size(S) → tamanho de S
- empty(S) → S está vazia? [TRUE/FALSE]
- full(S) → S está cheia? [TRUE/FALSE]

Roteiro

1. Introdução
2. **Pilhas**
3. Operações
4. Implementação com memória estática
5. Implementação com memória dinâmica
6. Síntese / Revisão
7. Referências

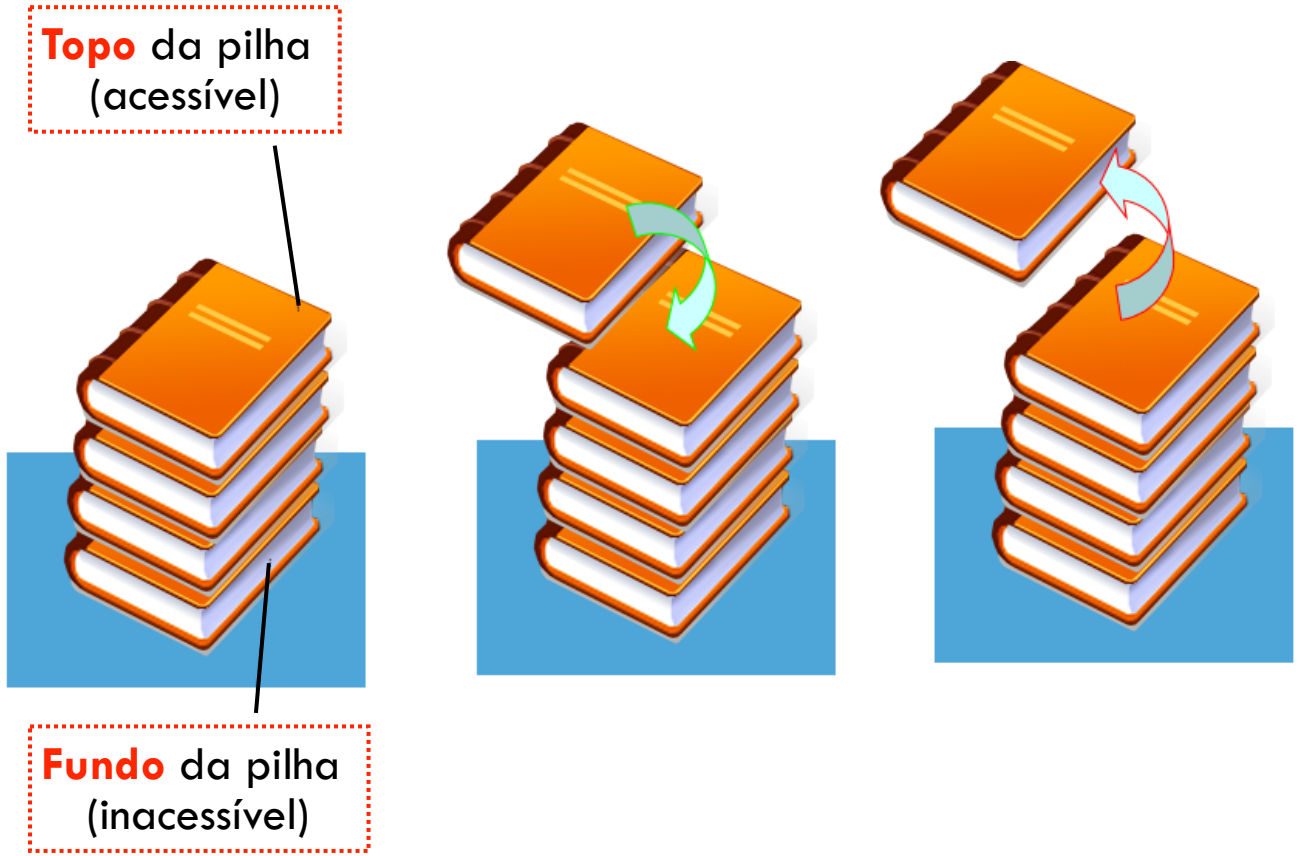
Pilhas

Pilhas

LIFO (**L**ast **I**n, **F**irst **O**ut)

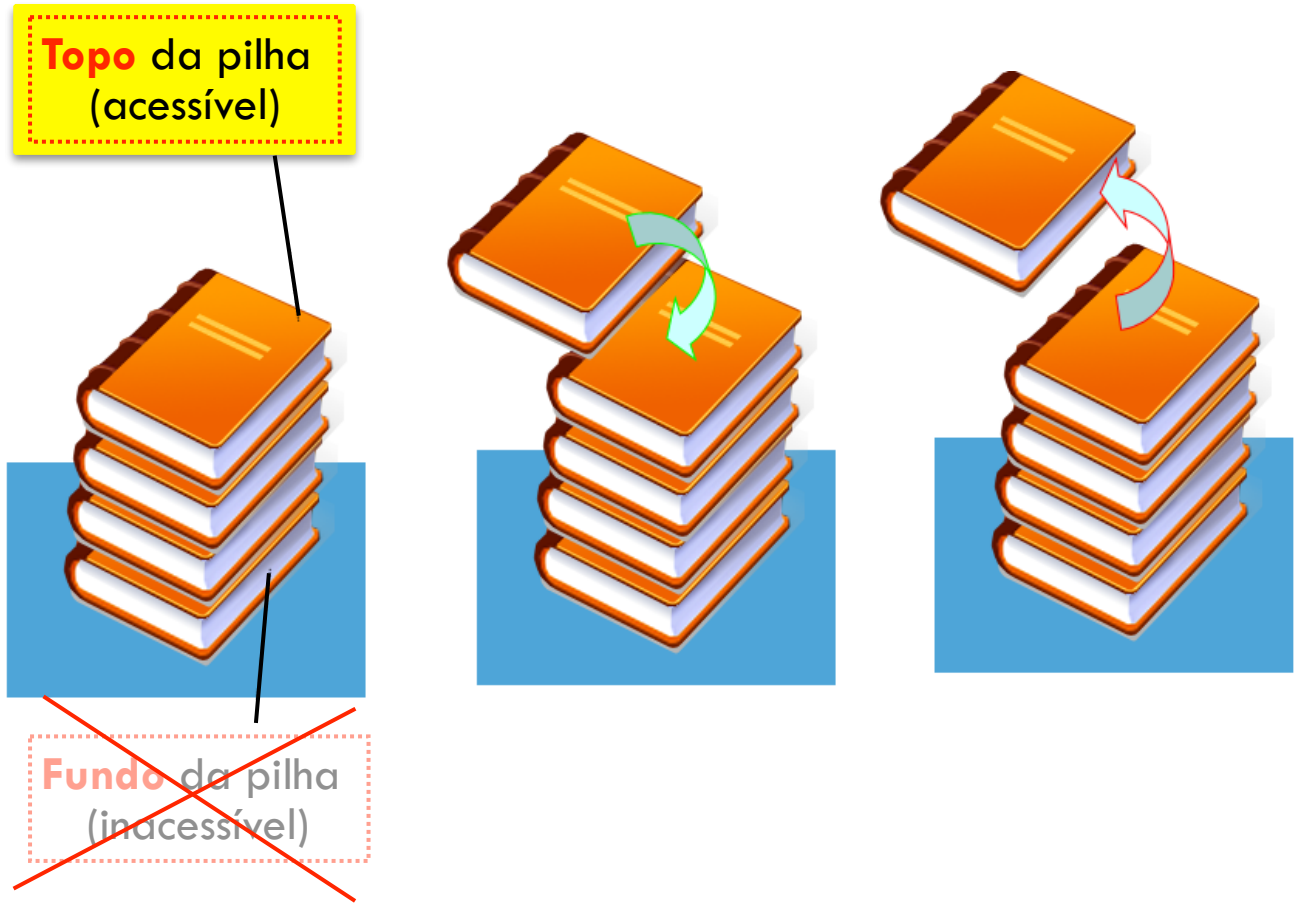
"Último elemento a entrar é o primeiro a sair"

Pilhas


Topo da pilha
(acessível)
Pilha de livros
(Stack)
Fundo da pilha
(inacessível)

Pilhas

Pilha de livros
(Stack)



Pilhas

Pilha de livros (Stack)

Empilha um novo livro no topo (**push**)

Remove um livro do topo (**pop**)

Pilhas

Pilha de livros (Stack)

Operações

Empilha um novo livro no topo (**push**)

Remove um livro do topo (**pop**)

Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

Operações em Pilhas

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

- ~~search(S, k) → procurar k em S [TRUE/FALSE]~~
- **insert(S, k)** → inserir k em S
- **delete(S, k)** → remover k em S
- ~~minimum(S) → menor valor armazenado em S~~
- ~~maximum(X) → maior valor armazenado em S~~
- ~~next(S, x) → elemento sucessor a x~~
- ~~previous(S, x) → elemento antecessor a x~~
- **size(S)** → tamanho de S
- **empty(S)** → S está vazia? [TRUE/FALSE]
- **full(S)** → S está cheia? [TRUE/FALSE]

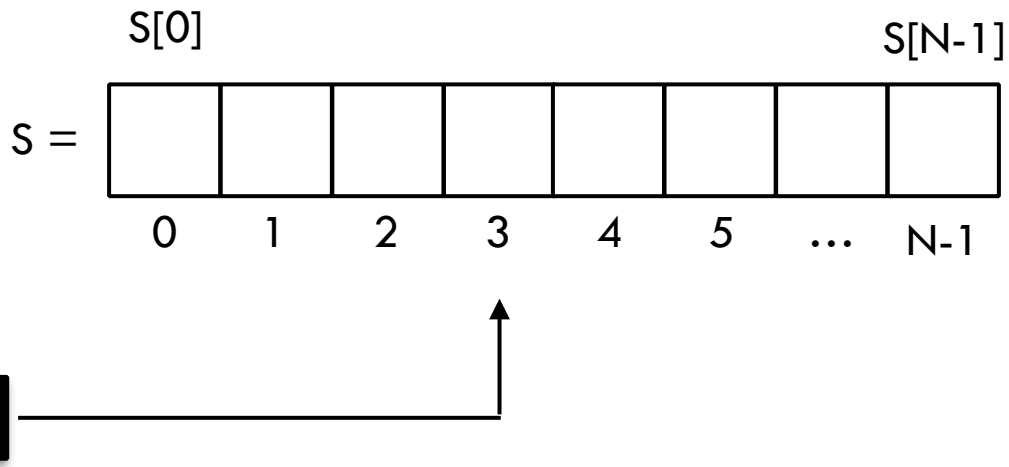
Operações em Pilhas

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

- ~~search(S, k) → procurar k em S [TRUE/FALSE]~~
- **push(S, k)** → empilhar k em S
- **pop(S, k)** → desempilhar k em S
- ~~minimum(S) → menor valor armazenado em S~~
- ~~maximum(X) → maior valor armazenado em S~~
- ~~next(S, x) → elemento sucessor a x~~
- ~~previous(S, x) → elemento antecessor a x~~
- **size(S)** → tamanho de S
- **empty(S)** → S está vazia? [TRUE/FALSE]
- **full(S)** → S está cheia? [TRUE/FALSE]

Pilhas

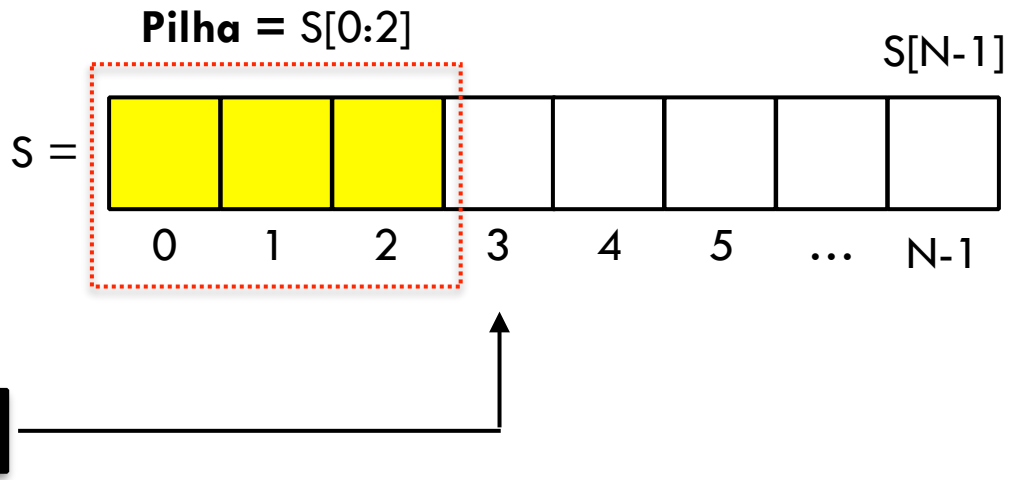
S = Arranjo de N elementos



Indexa a posição disponível para inserção

Pilhas

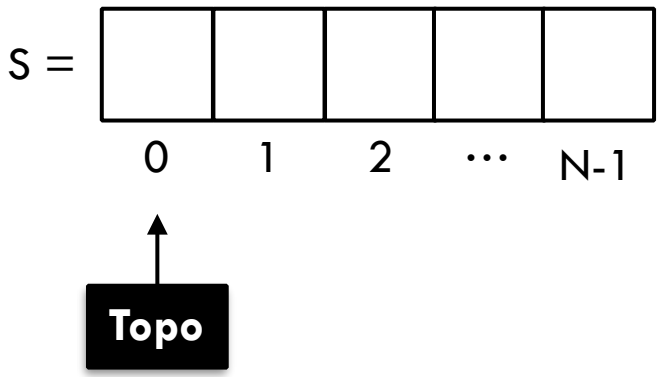
S = Arranjo de N elementos



Indexa a posição disponível para inserção (topo =3)

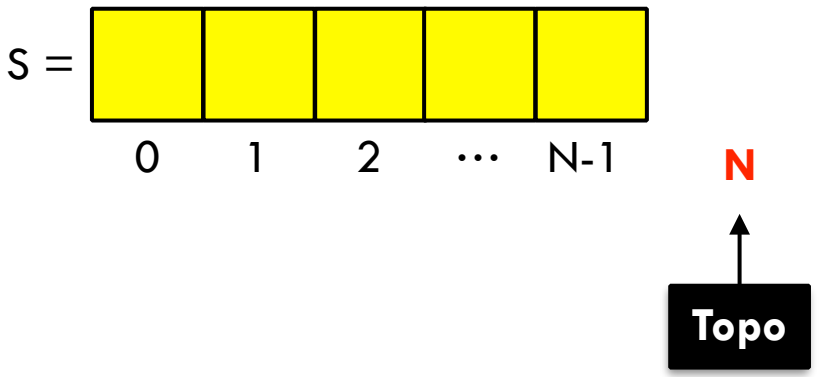
Pilhas

- S[S.topo] == 0 —> pilha está vazia




```
isEmpty (S)
1. return(S.topo == 0)
```

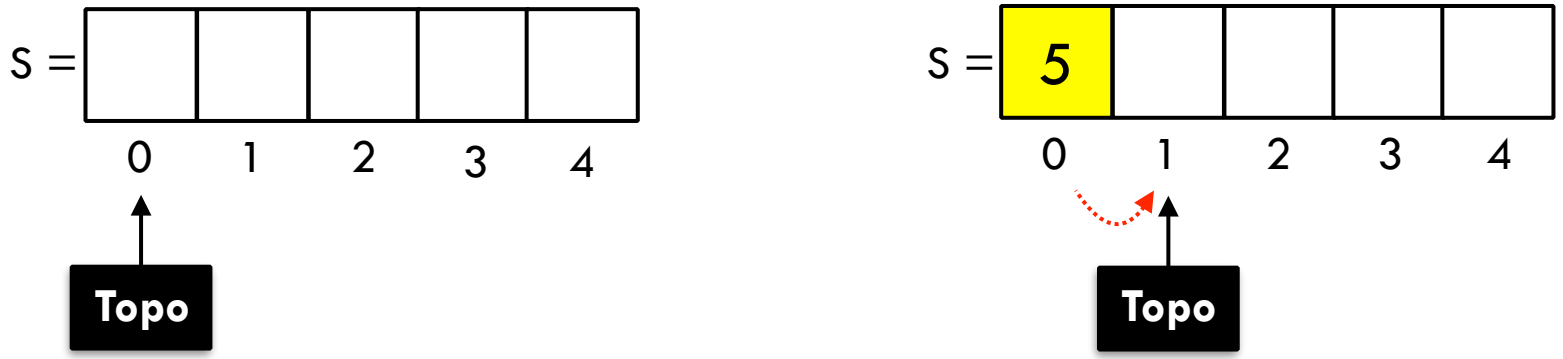
- S [S.topo] == N —> pilha está cheia




```
isFull (S)
1. return(S.topo == N)
```

Empilhar

- Empilhar (inserir) elemento x = 5

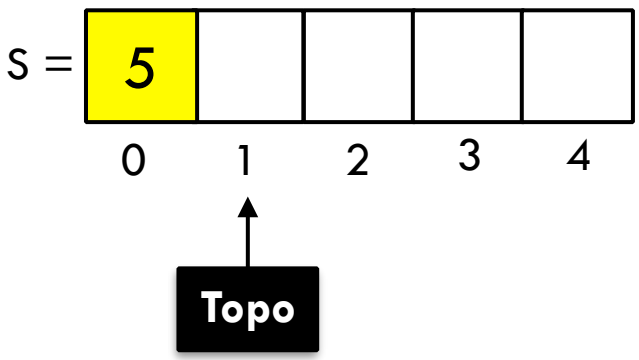


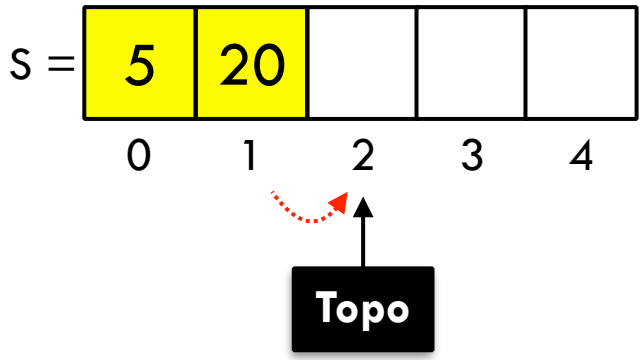
- **Pseudocódigo**

```
Push (S, x)
1. S[S.topo] = x;
2. S.topo = S.topo + 1;
```

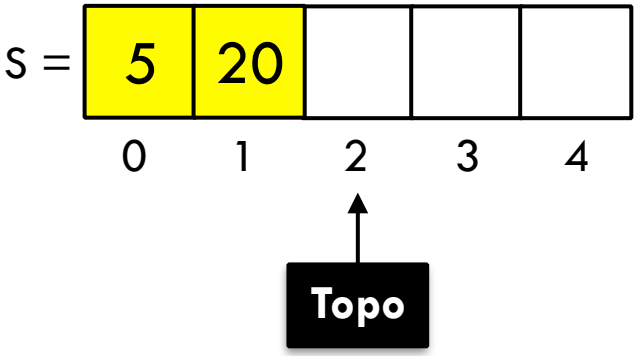
Empilhar (Push)

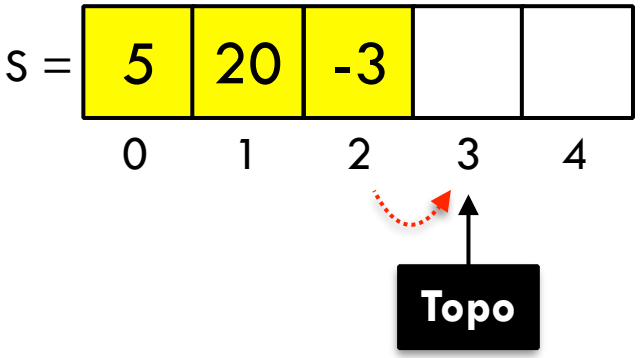
- Empilhar (inserir) elemento x = 20





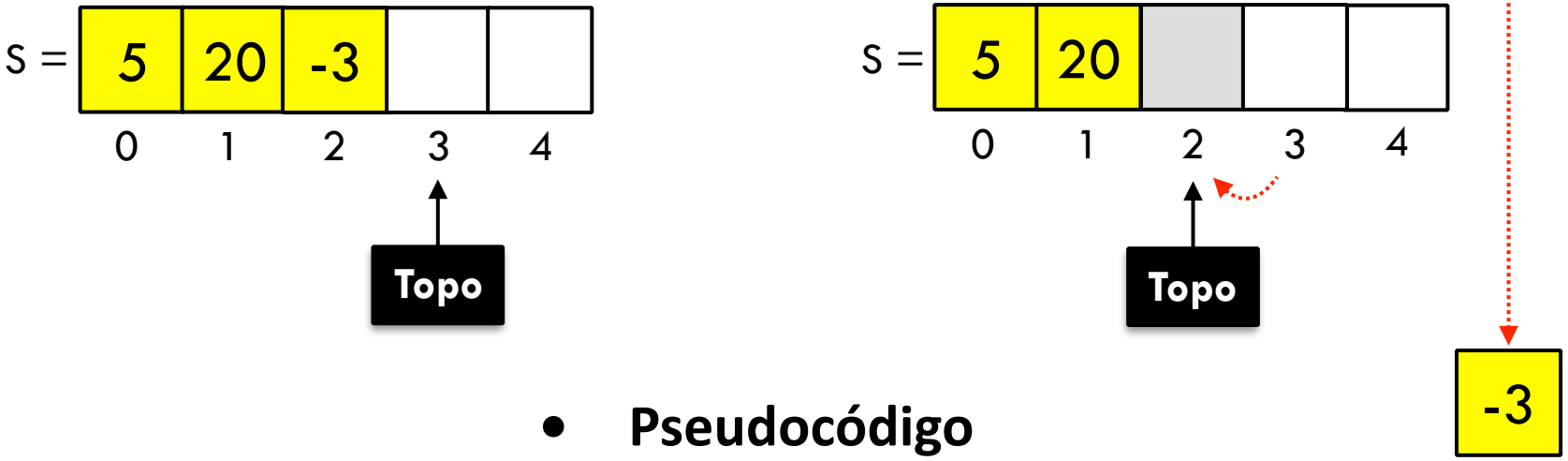
- Empilhar (inserir) elemento x = -3





Desempilhar

- desempilhar (remover) elemento

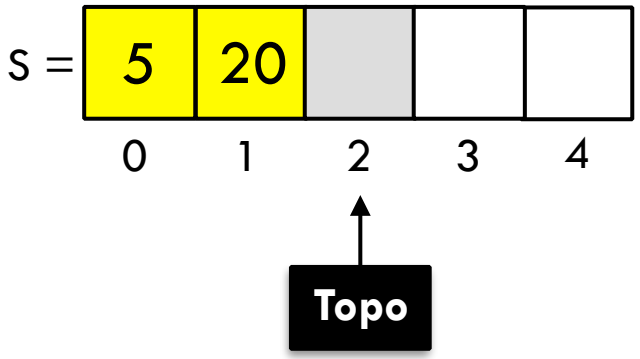


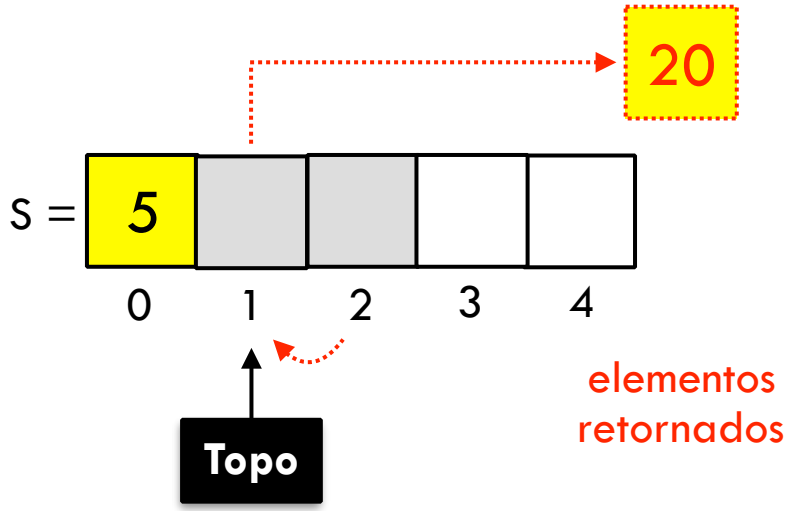
- **Pseudocódigo**

```
Pop (S)
1. x = S[S.topo - 1];
2. S.topo = S.topo - 1;
3. return(x);
```

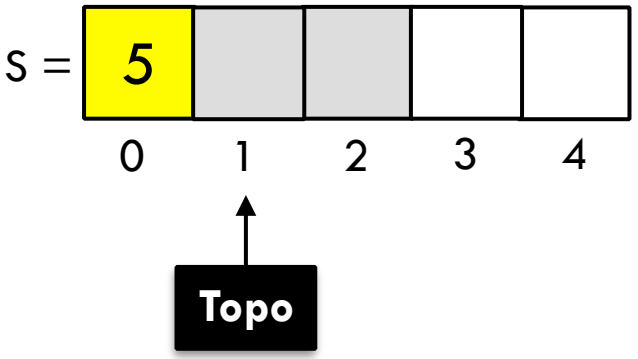
Desempilhar

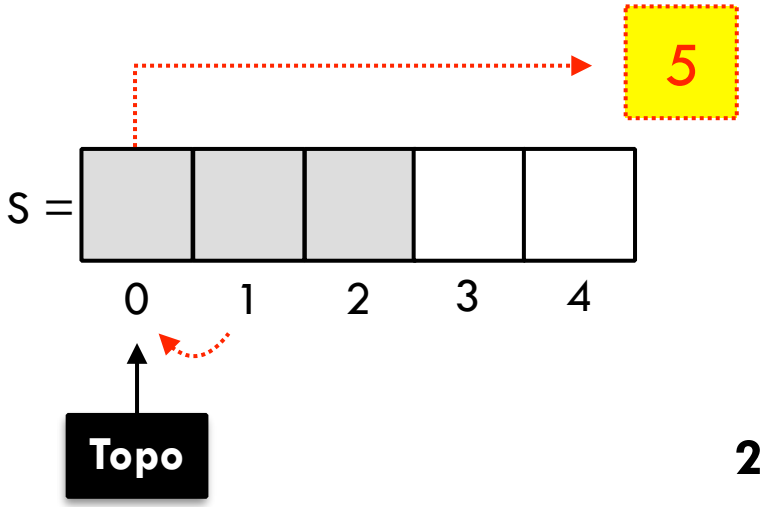
- desempilhar (remover) elemento



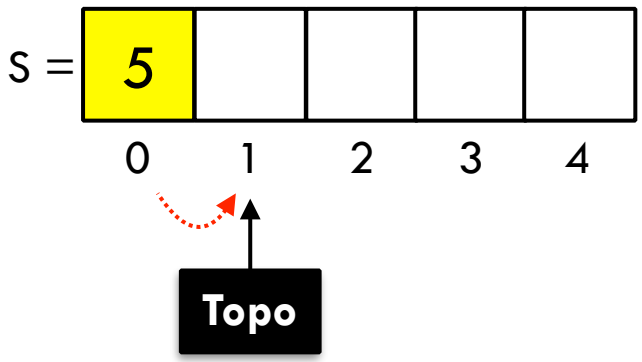


- desempilhar (remover) elemento





Acessar topo (sem remoção)



- Pseudocódigo

```
Top (S)
1. x = [S.topo-1];
2. return(x);
```

Exercício 01

- Ilustre cada estado de uma pilha após realizar as seguintes operações (em ordem)
  - Push(S, 4)
  - Push(S, 1)
  - Push(S, 3)
  - Pop(S)
  - Push(S, 8)
  - Pop(S)

- Considere que a pilha está inicialmente vazia e é armazenada em um arranjo S[1 .. 6]

Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

Implementação (Estática)

- Dois tipos abstratos de dados

```
#define N 100

typedef struct {
    int key;
   /* pode ter mais elementos */
} Item;


typedef struct {
    Item vet[N];
    int topo;
} StaticStack;
```

implementa o nosso objeto

implementa o TDA para Pilha

Implementação (Estática)

- Dois tipos abstratos de dados

```
void init(StaticStack *pilha);
int isEmpty(StaticStack *pilha);
int isFull(StaticStack *pilha);
void push(Item item, StaticStack *pilha);
void pop(StaticStack *pilha, Item *item);
int size(StaticStack *pilha);
Item top(StaticStack *pilha);
void print(StaticStack *pilha);
```

Exercício 02

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

Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

Revisão

- Pilhas
  - o que é
  - operações
  - implementação estática

Próximas Aulas

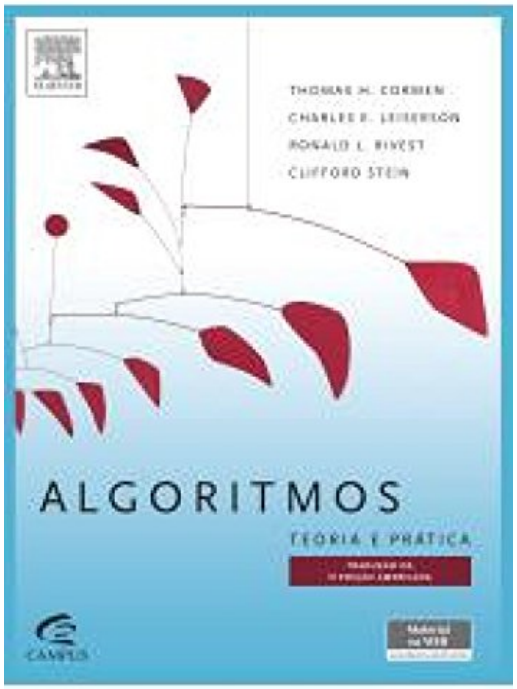
- Pilhas → implementação dinâmica
- Filas/ Deques

- Implementação de Listas Lineares
  - single-linked
  - double-linked

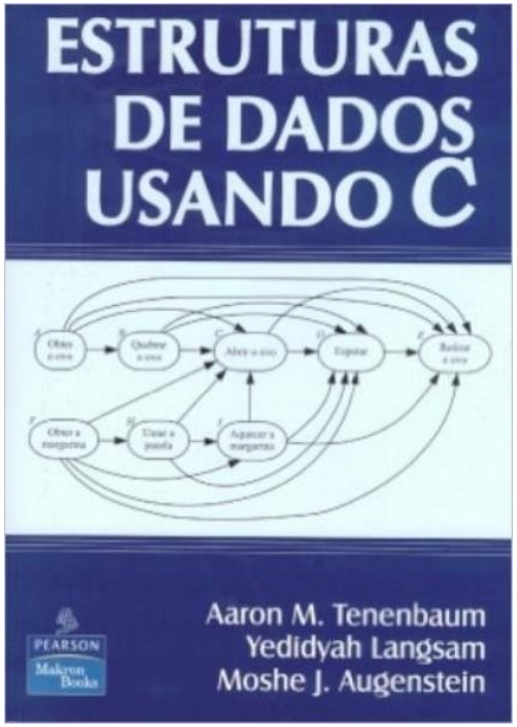
Roteiro

1. Introdução
2. Pilhas
3. Operações
4. Implementação com memória estática
5. Síntese / Revisão
6. Referências

Referências sugeridas

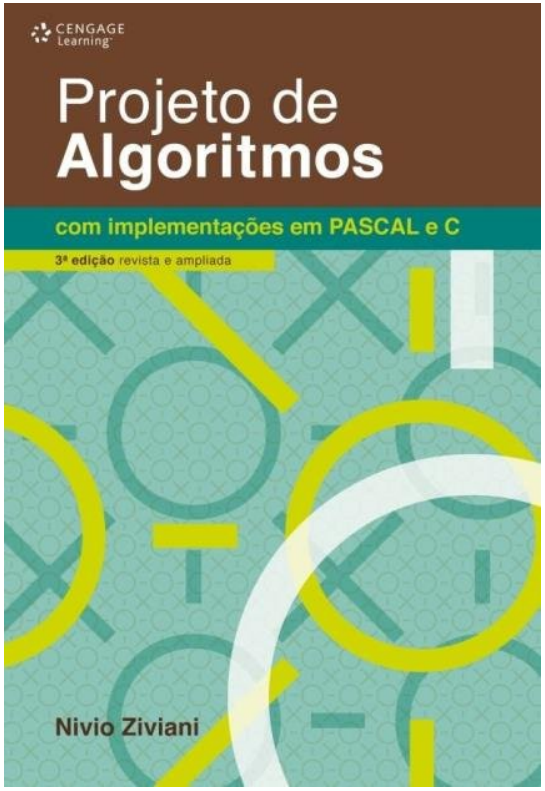


[Cormen et al, 2018]



[Tenenbaum et al, 1995]

Referências sugeridas



[Ziviani, 2010]

[Drozdek, 2017]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br