

ED62A-COM2A

ESTRUTURAS DE DADOS

Aula 03 - Pilha
(Implementação estática)

Prof. Rafael G. Mantovani

Roteiro



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

Roteiro

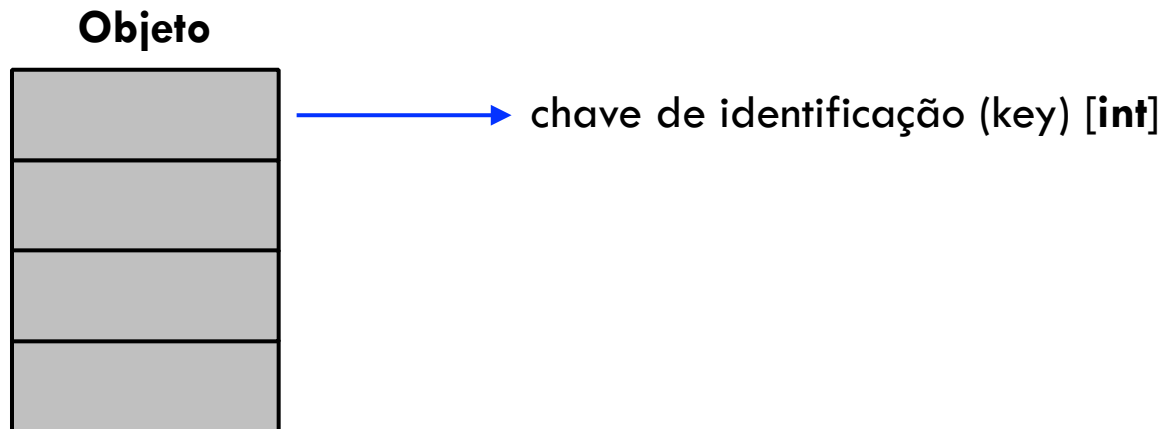
- 1 Introdução**
- 2 Pilhas**
- 3 Operações**
- 4 Implementação com memória estática**
- 5 Síntese / Revisão**
- 6 Referências**

Introdução

- Conjuntos são fundamentais para Computação / Matemática
 - Matemática = conjuntos são invariáveis
 - Computação = são dinâmicos
- Operações
 - diferentes operações em um conjunto
 - operações de "dicionários"
 - inserir, eliminar e testar a existência de um elemento
 - a melhor forma de implementar depende das operações

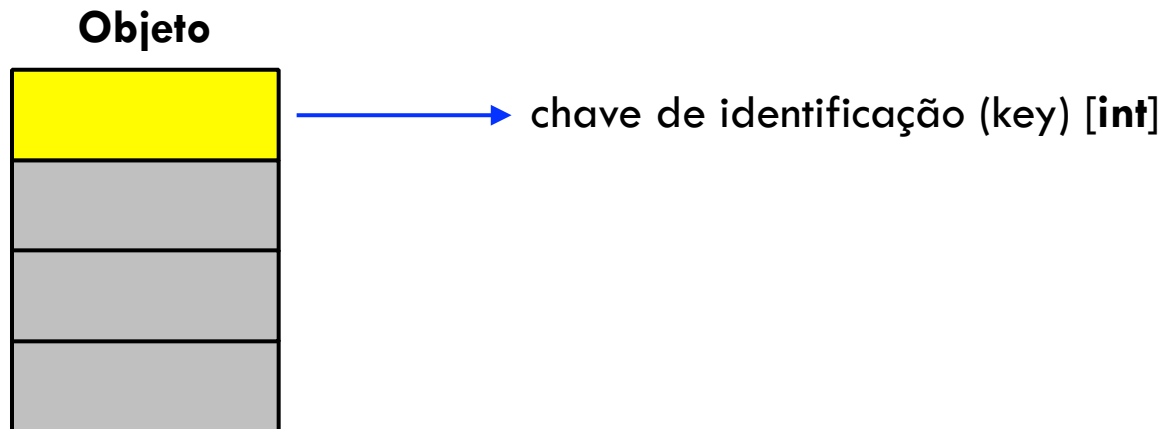
Introdução

- Elemento (objeto) → vários atributos



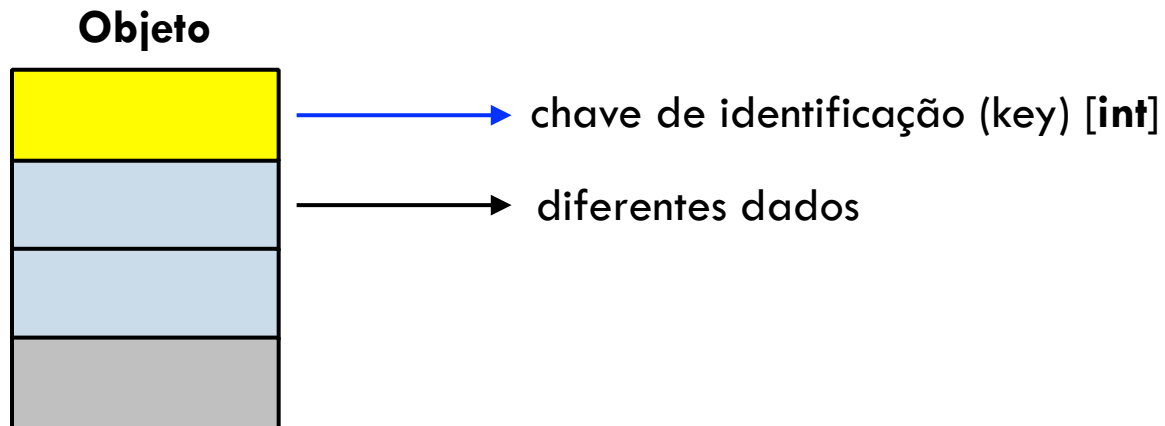
Introdução

- Elemento (objeto) → vários atributos



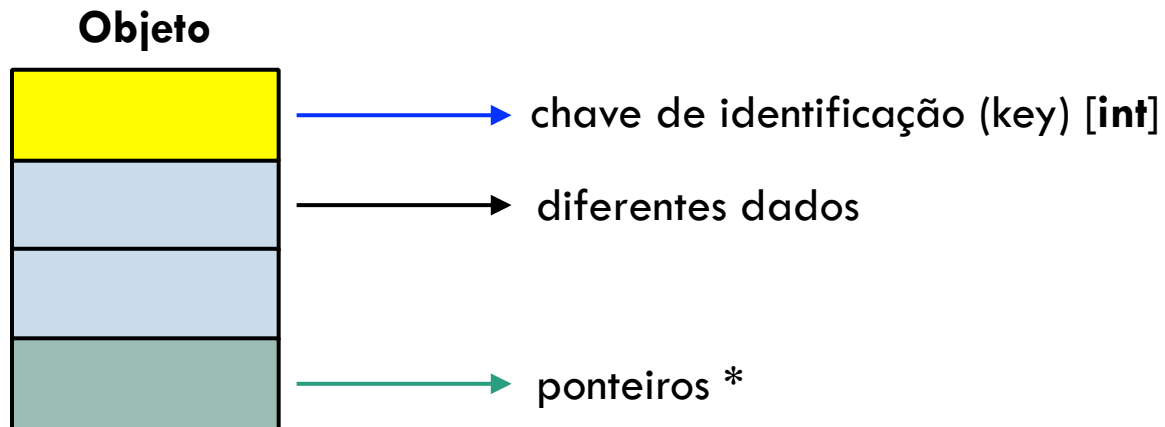
Introdução

- Elemento (objeto) → vários atributos



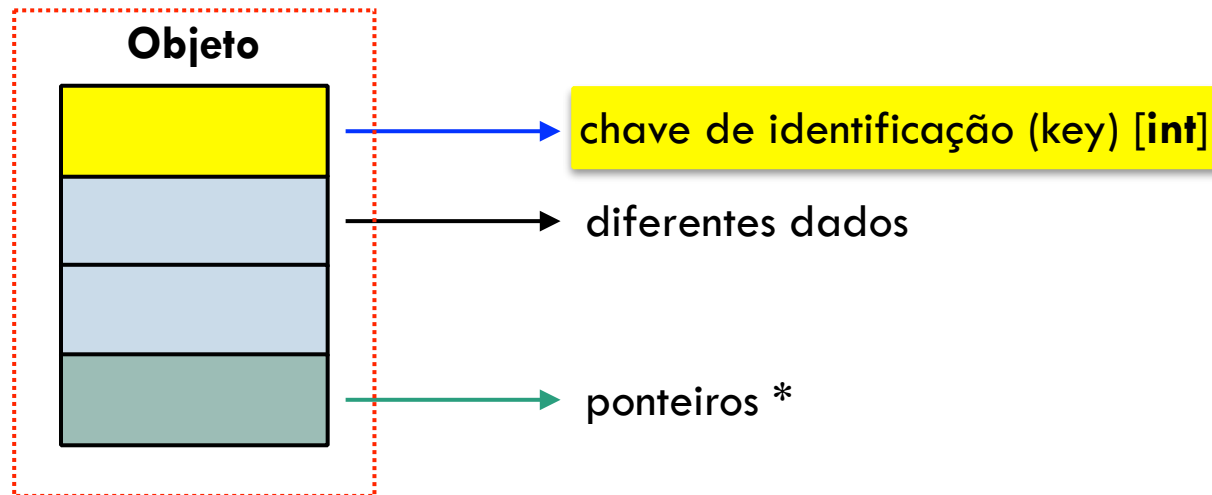
Introdução

- Elemento (objeto) → vários atributos



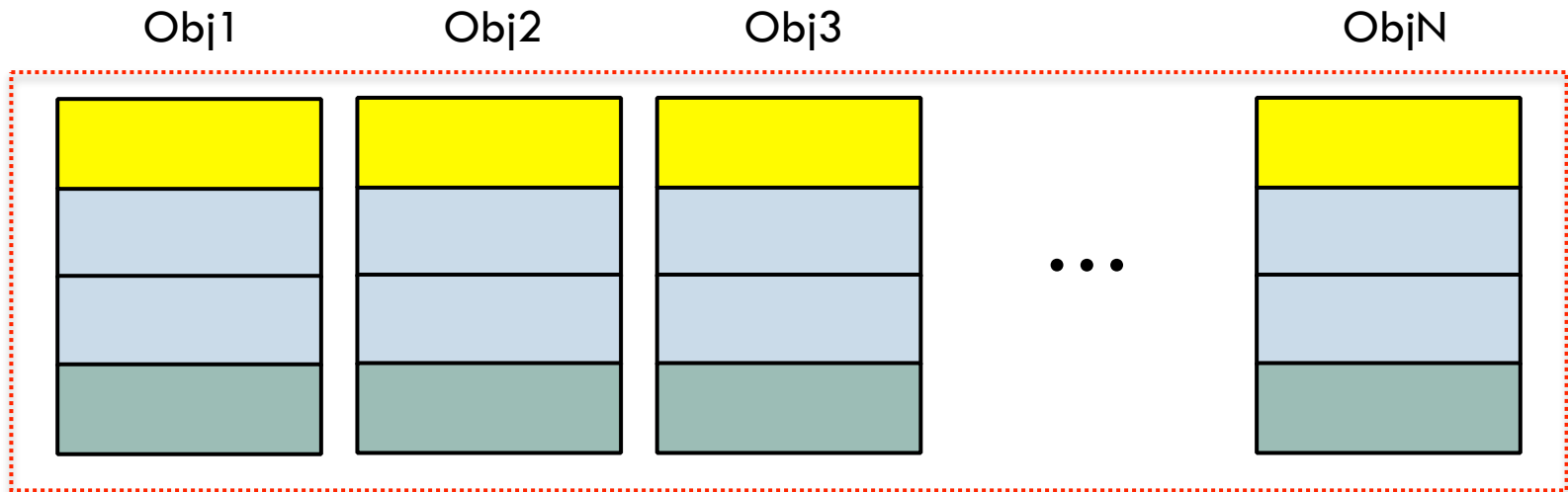
Introdução

- Elemento (objeto) → vários atributos



Introdução

- Estrutura = Arranjo de N Objetos



Estrutura de Dados

Operações

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

iniciar (**S**, **x**)

Inserir (**S**, **k**)

Remover (**S**, **k**)

pesquisar (**S**, **k**)

destruir (**S**, **x**)

**Operações de
modificação**

Operações

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

iniciar (S, x)

Cria e inicia a estrutura

Inserir (S, k)

Insere um novo elemento na estrutura

Remover (S, k)

Remove um elemento da estrutura

pesquisar (S, k)

Pesquisa um elemento existente na estrutura

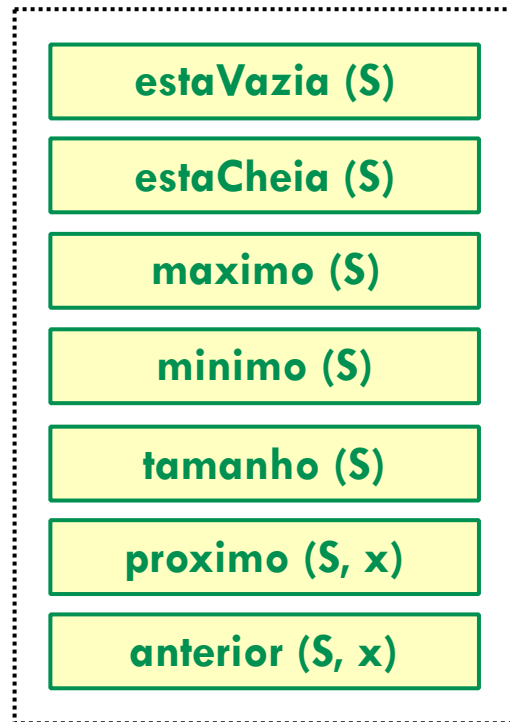
destruir (S, x)

Destrói a estrutura e desloca a memória

**Operações de
modificação**

Operações

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



**Operações adicionais
de consulta**

Operações

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

estaVazia (S)

Verifica se a estrutura está vazia

estaCheia (S)

Verifica se a estrutura está cheia

maximo (S)

Retorna o elemento de maior valor

minimo (S)

Retorna o elemento de menor valor

tamanho (S)

Retorna a quantidade de elementos na estrutura

proximo (S, x)

Retorna o próximo elemento (segundo critério)

anterior (S, x)

Retorna o elemento anterior (segundo critério)

**Operações adicionais
de consulta**

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 (Last In, First Out)

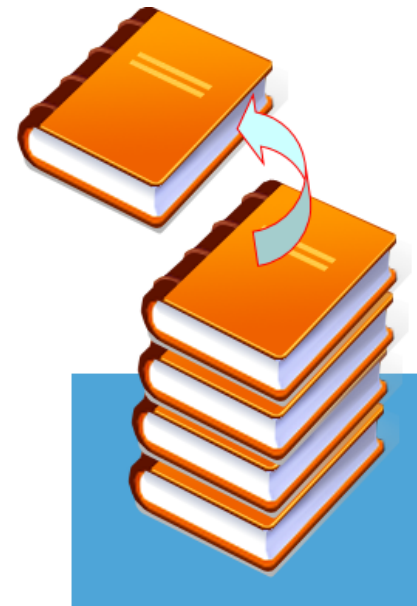
"Último elemento a entrar é o primeiro a sair"

Pilhas

Topo da pilha
(acessível)

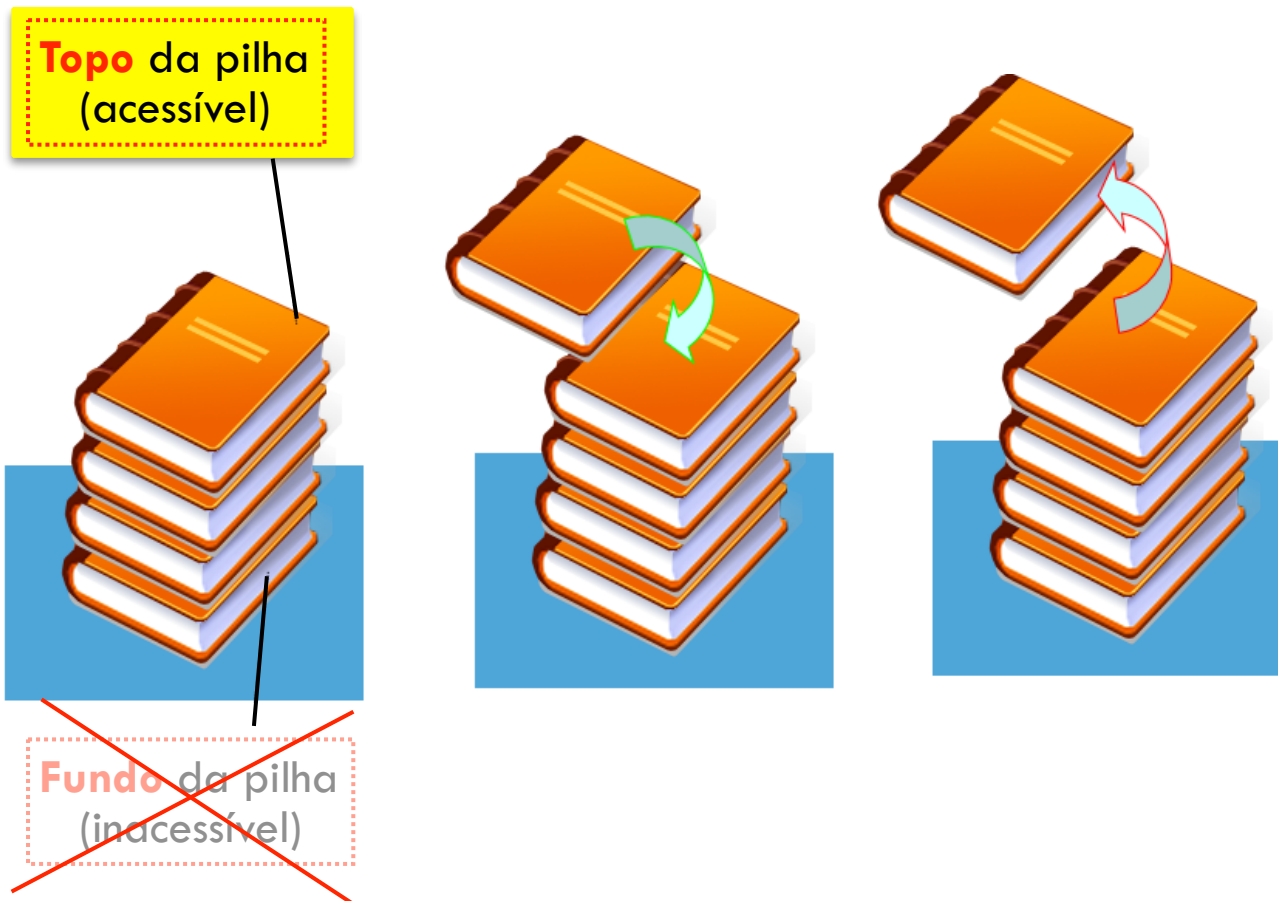


Fundo da pilha
(inacessível)



Pilha de livros
(Stack)

Pilhas



Pilhas

Topo da pilha
(acessível)



Pilha de livros
(Stack)

~~Fundo~~ da pilha
(inacessível)



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



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

Pilhas

Topo da pilha
(acessível)



Pilha de livros
(Stack)

~~Fundo~~ da pilha
(inacessível)

Operações



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



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

Roteiro

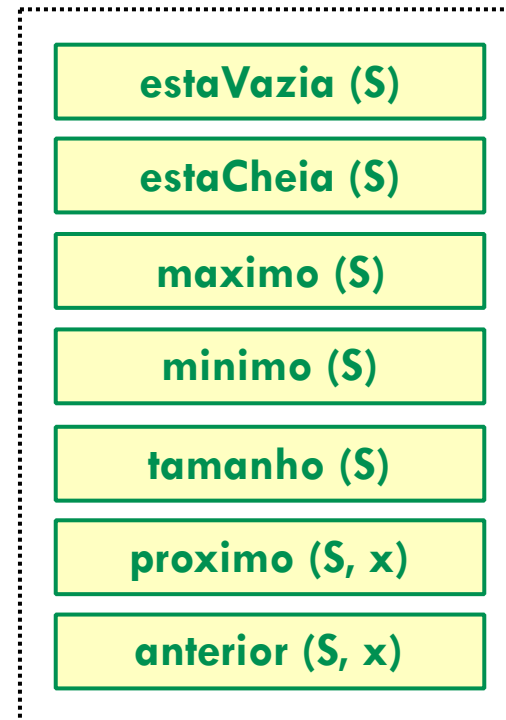
- 1 Introdução
- 2 Pilhas
- 3 Operações
- 4 Implementação com memória estática
- 5 Síntese / Revisão
- 6 Referências

Operações em Pilhas Estáticas

Dada uma estrutura S , chave k , elemento x :



**Operações de
modificação**



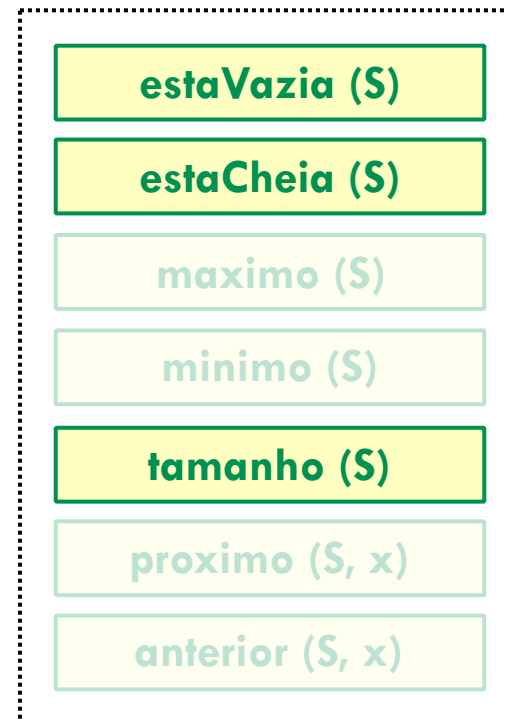
**Operações adicionais
de consulta**

Operações em Pilhas Estáticas

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



**Operações de
modificação**



**Operações adicionais
de consulta**

Operações em Pilhas Estáticas

iniciar (S, x)

Inicializa a pilha e suas variáveis

Inserir (S, k)

Inserir objeto na pilha (empilhar)

Remover (S, k)

Remover objeto da pilha (desempilhar)

Topo (S)

Retorna o objeto do topo, sem remover

estaVazia (S)

Retorna booleano indicando se a pilha está vazia

estaCheia (S)

Retorna booleano indicando se a pilha está cheia

tamanho (S)

Retorna a quantidade de elementos na pilha

Pseudocódigos

Função (param1, ...)

1. Instrução 1

2. ...

3. Instrução N

4. return (x)

Pseudocódigos

Função (param1, ...)

1. Instrução 1
2. ...
3. Instrução N
4. **return** (x)

Inputs (entradas)
- parâmetros

Pseudocódigos

Função (param1, ...)

1. Instrução 1

2. ...

3. Instrução N

4. return (retornos)

Sequência de
Instruções

Pseudocódigos

Função (param1, ...)

1. Instrução 1

2. ...

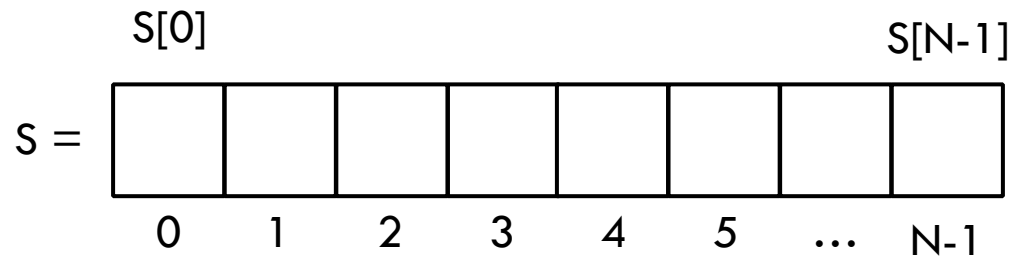
3. Instrução N

4. **return (x)**

Se necessário,
retornos da função

Pilhas

S = Arranjo de N elementos

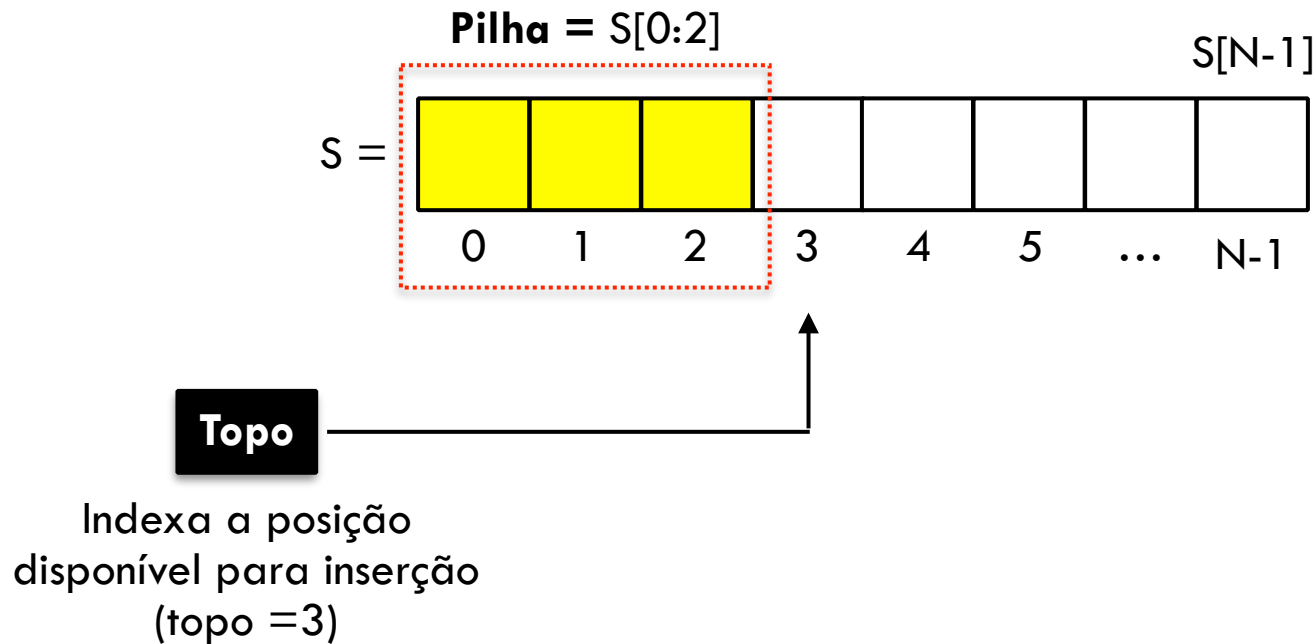


Topo

Indexa a posição
disponível para inserção

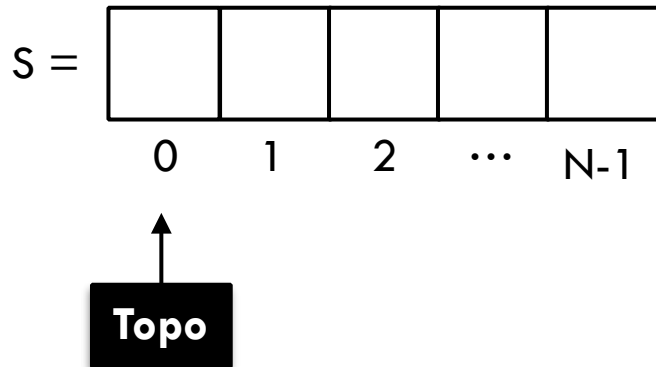
Pilhas

S = Arranjo de N elementos



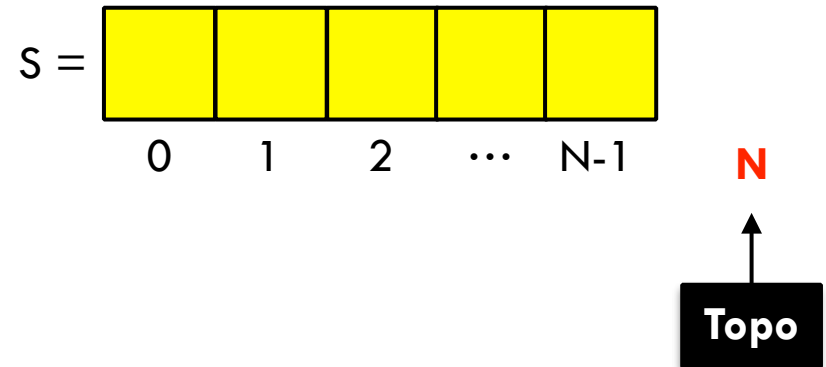
Pilhas

- $S[S.topo] == 0 \rightarrow$ pilha está vazia
- $S[S.topo] == N \rightarrow$ pilha está cheia



estaVazia (S)

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

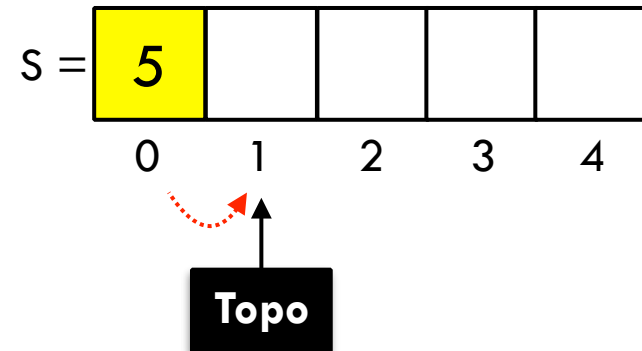
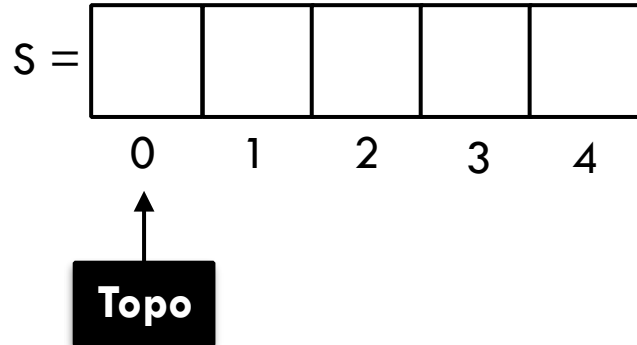


estaCheia (S)

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

Empilhar

- Empilhar (inserir) elemento $x = 5$

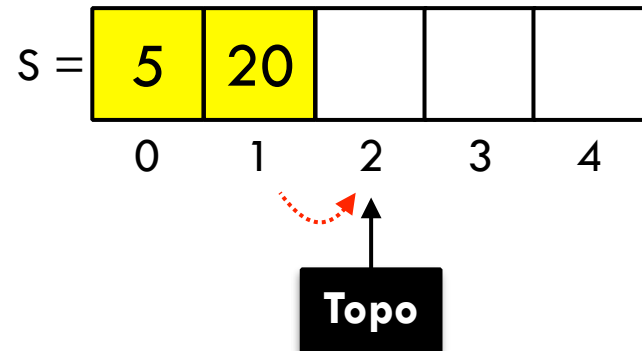
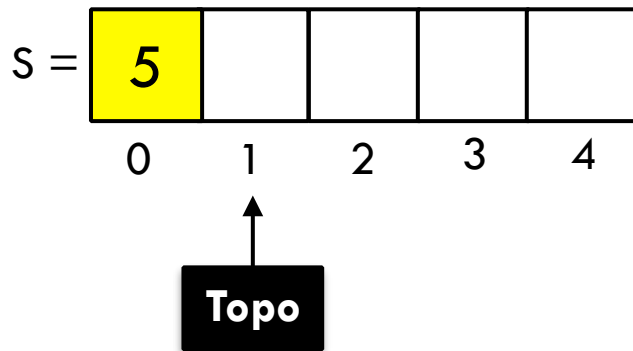


- Pseudocódigo**

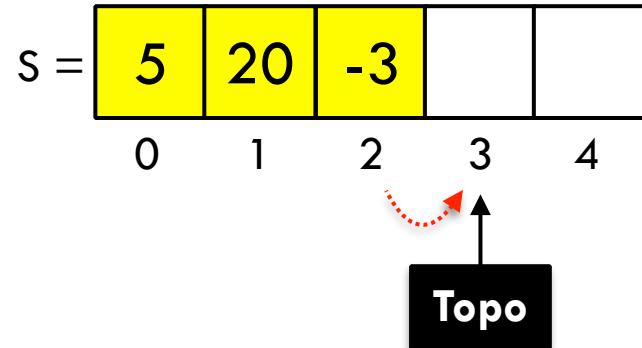
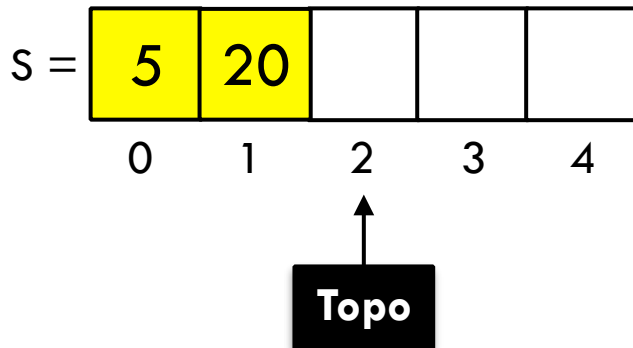
```
Empilha (S, x)
1. S[S.topo] = x;
2. S.topo = S.topo + 1;
```

Empilhar (Push)

- Empilhar (inserir) elemento $x = 20$

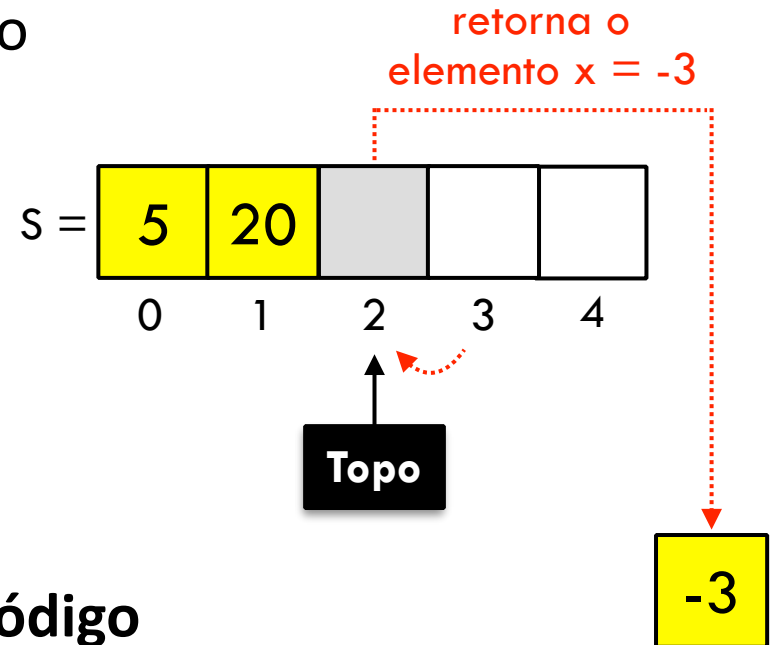
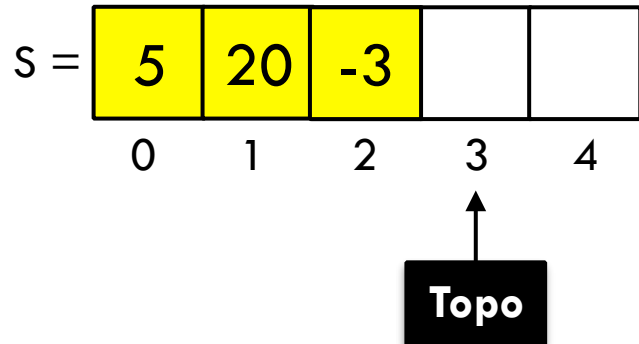


- Empilhar (inserir) elemento $x = -3$



Desempilhar

- desempilhar (remove) elemento



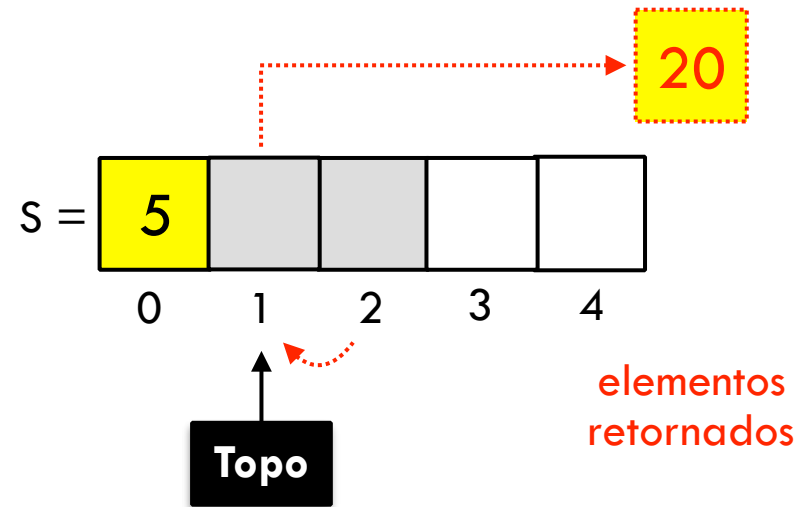
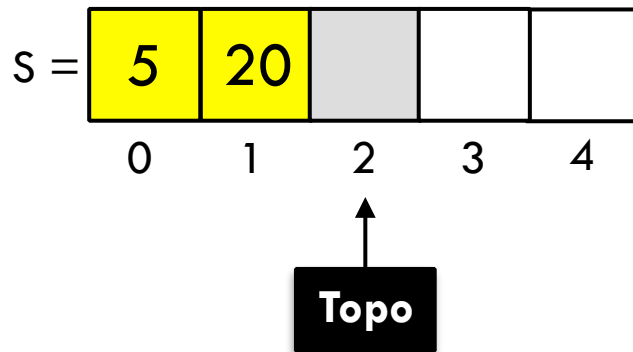
- Pseudocódigo

Desempilha (S)

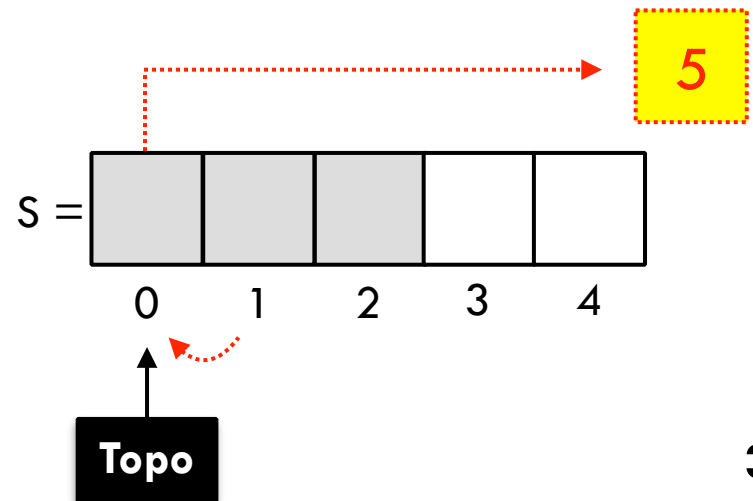
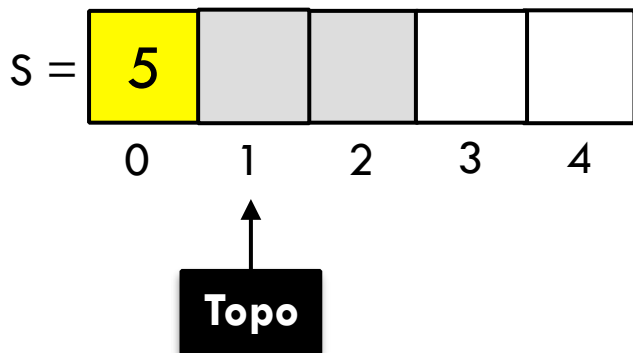
```
1.  $x = S[S.topo - 1];$   
2.  $S.topo = S.topo - 1;$   
3. return(x);
```

Desempilhar

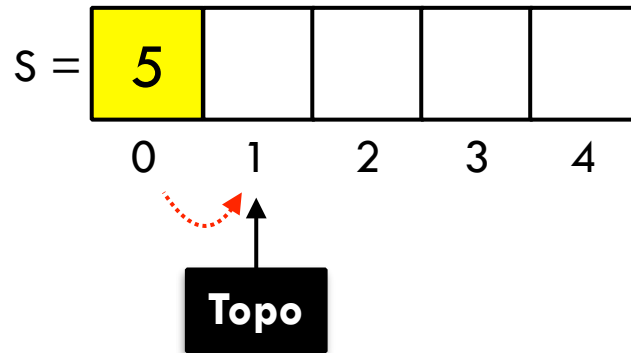
- desempilhar (remove) elemento



- desempilhar (remove) elemento



Acessar topo (sem remoção)



- Pseudocódigo

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

Exercício 01

- Ilustre cada estado de uma pilha após realizar as seguintes operações (em ordem)
 - (A) `Push(S, 4)` *// empilha o elemento 4*
 - (B) `Push(S, 1)` *// empilha o elemento 1*
 - (C) `Push(S, 3)` *// empilha o elemento 3*
 - (D) `Pop(S)` *// desempilha o valor do topo*
 - (E) `Push(S, 8)` *// empilha o elemento 8*
 - (F) `Pop(S)` *// desempilha o valor do topo*
- 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 */
} Objeto;
```

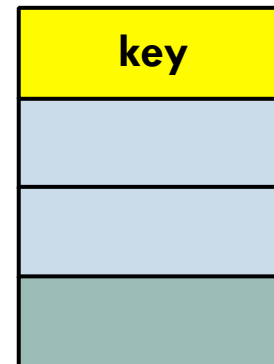

Implementação (Estática)

- Dois tipos abstratos de dados

```
#define N 100

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

implementa o nosso
objeto



Implementação (Estática)

- Dois tipos abstratos de dados

```
#define N 100

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

.....→ implementa o nosso
objeto

```
typedef struct {
    Objeto vetor[N];
    int topo;
} PilhaEstatica;
```

Implementação (Estática)

- Dois tipos abstratos de dados

```
#define N 100

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

implementa o nosso
objeto

```
typedef struct {
    Objeto vetor[N];
    int topo;
} PilhaEstatica;
```

implementa o TAD
para Pilha Estática
(armazena Objetos)

Implementação (Estática)

- Dois tipos abstratos de dados

```
void iniciaPilha(PilhaEstatica *pilha);
bool estaVazia(PilhaEstatica *pilha);
bool estaCheia(PilhaEstatica *pilha);
void push(Objeto obj, PilhaEstatica *pilha);
void pop(PilhaEstatica *pilha, Objeto *obj);
int size(PilhaEstatica *pilha);
Objeto top(PilhaEstatica *pilha);
void print(PilhaEstatica *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

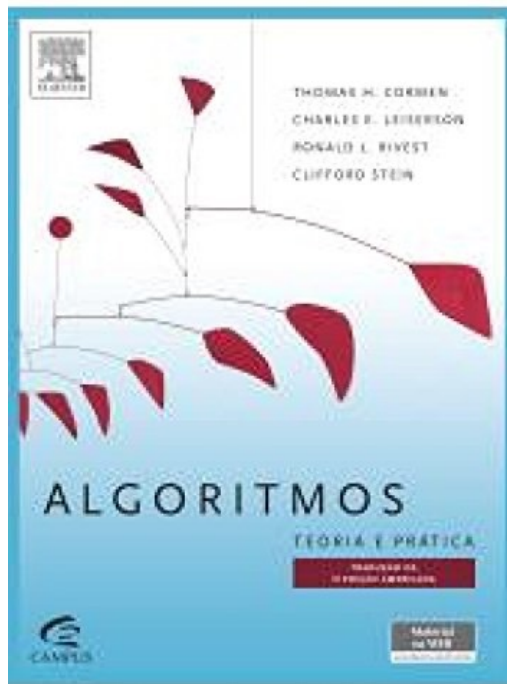


- Pilhas → implementação dinâmica
- Filas/ Deques
- Implementação de Listas Lineares
 - single-linked
 - double-linked

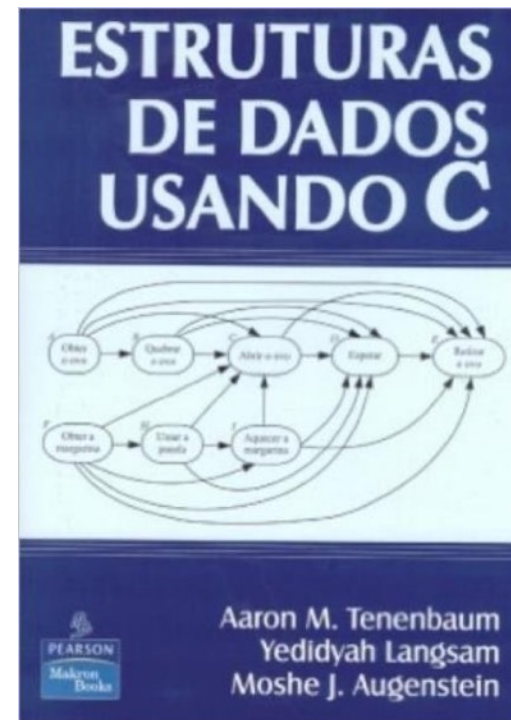
Roteiro

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

Referências sugeridas

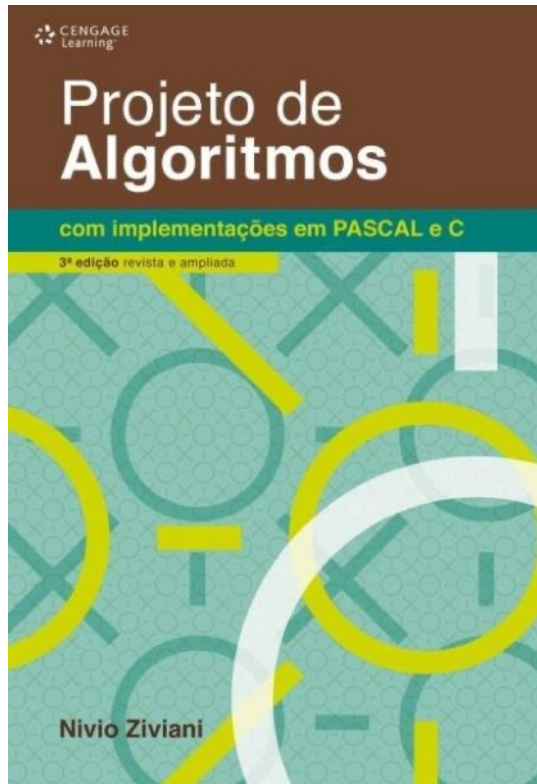


[Cormen et al, 2018]



[Tenenbaum et al, 1995]

Referências sugeridas



[Ziviani, 2010]



[Drozdek, 2017]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br