3-Pilhas

CCA0916 - ESTRUTURA DE DADOS I
BCC

Prof. Dr. Paulo César Rodacki Gomes
paulo.gomes@ifc.edu.br

Blumenau, 2017

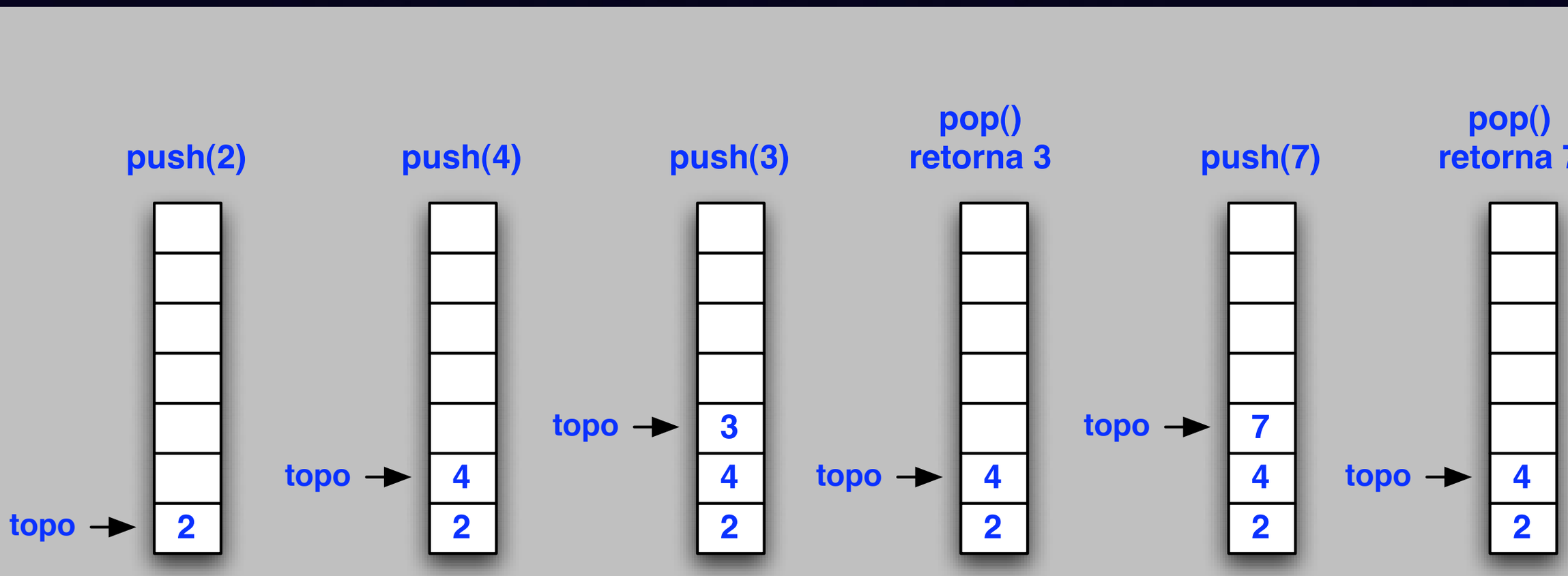
Tópicos

- Motivação
- Interface do tipo pilha
- Implementação de pilha com vetor
- Implementação de pilha com lista

Introdução - Pilha

- Novo elemento é inserido no topo e acesso é apenas no topo
- o primeiro que sai é o último que entrou (LIFO - *last in first out*)
- operações básicas:
  - empilhar (*push*) um novo elemento, inserindo-o no topo
  - desempilhar (pop) um elemento, removendo-o do topo

Introdução - Pilha



Interface Pilha

- Implementação usando vetor:
  - simplicidade
  - limite máximo na quantidade de elementos
- Implementação usando lista:
  - não há limite fixo para quantidade de elementos

Interface Pilha

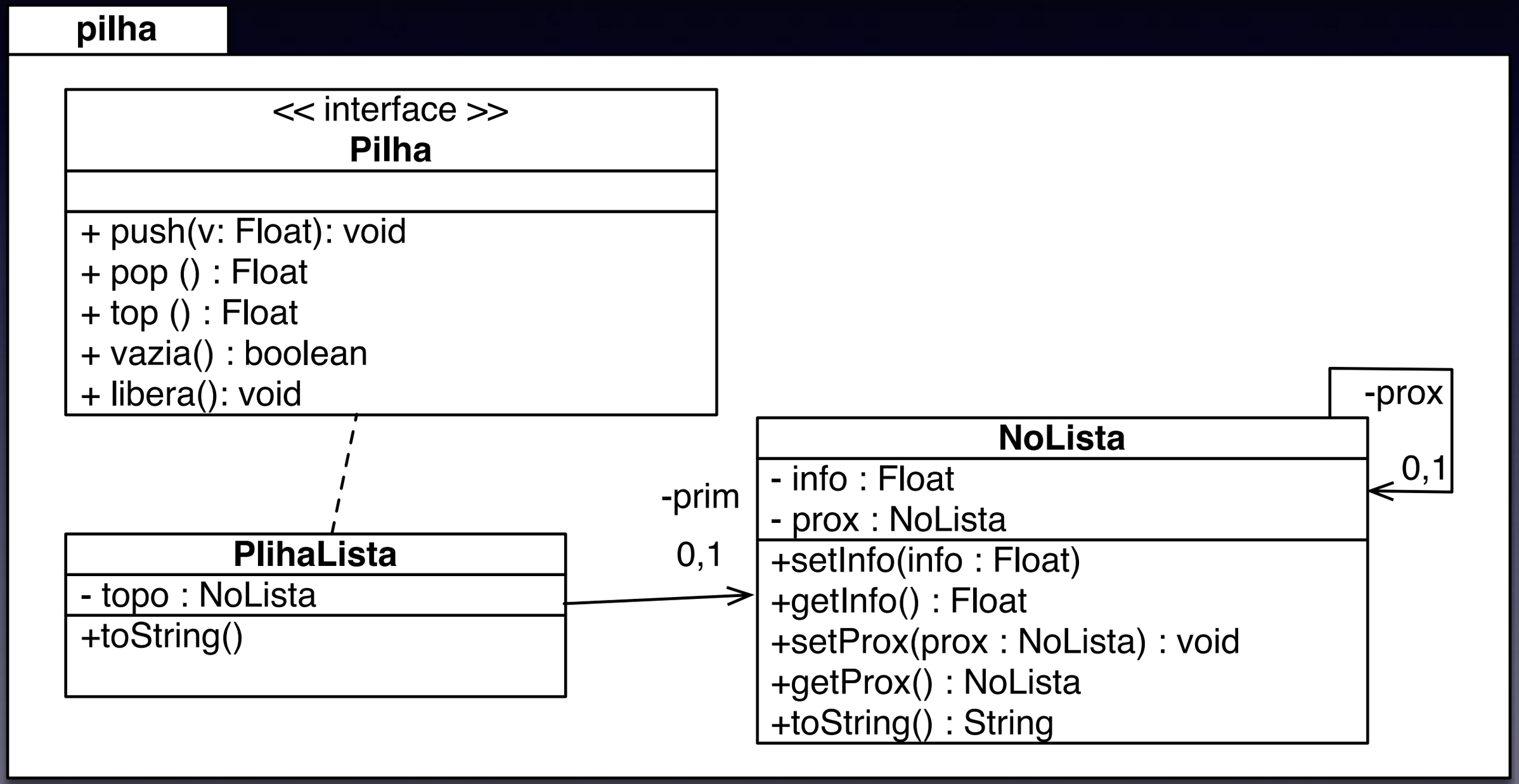
push	insere um valor (inteiro) na pilha, lança exceção caso chegue ao limite
pop	retira um valor da pilha, lança exceção se a pilha estiver vazia
vazia	verifica se a pilha está vazia
libera	libera os recursos utilizados pela pilha

Interface Pilha

<< interface >>
Pilha

+ push(v: float): void : throws Exception
+ pop () : float : throws Exception
+ top () : float : throws Exception
+ vazia() : boolean
+ libera(): void

Diagrama de classes



Interface Pilha

```
public interface Pilha {
   void push(int v) throws Exception;
   int pop () throws Exception;
   boolean vazia();
   void libera();
}
```

Implementação de Pilha com Vetor

- Classe PilhaVetor
- Vetor *vet* armazena os elementos da pilha
- tam: tamanho máximo da pilha (para alocação dinâmica de *vet* no construtor da classe
- n: quantidade atual de elementos inseridos
- elementos inseridos ocupam as primeiras posições livres no vetor (elemento `vet[m-1]` represena o elemento do topo)

Implementação de Pilha com Vetor

```
public class PilhaVetor implements Pilha {

    // quantidade de valores armazenados
    private int n;

    // tamanho do vetor
    private int tam;

    // vetor que armazena elementos
    private int[] vet;

    // implementação dos métodos...
}
```

Implementação de Pilha com Vetor

Construtor da classe:

- recebe tamanho da pilha;
- aloca dinamicamente o vetor
- inicializa atributos de tamanho e quantidade atual de elementos na pilha

Algoritmo: PilhaVetor(int tam)

$this.vet \leftarrow new\ int[tam]$;
$this.tam \leftarrow tam$;
$this.n \leftarrow 0$;

Algoritmo 2.1: Construtor da classe PilhaVetor

Implementação de Pilha com Vetor

Algoritmo: push(int v)

se $n == tam$ **então**
  throw new Exception("ERRO: a capacidade da pilha estourou!");
senão
  `// insere elemento na prox.  posição livre`
  $this.vet[n] \leftarrow v$;
  `// incrementa o número de elementos`
  $this.n++$;

Algoritmo 2.2: Método push da classe PilhaVetor

Implementação de Pilha com Vetor

Método pop:

- verifica se a pilha está vazia
- retira o elemento do topo da pilha, retornando seu valor

Algoritmo: int pop()

```
int v;
se this.vazia() então
    throw new Exception("ERRO: pilha vazia!");
// retira elemento do topo
v ← this.vet[n − 1];
// decrementa o número de elementos
this.n − −;
retorna v;
```

Algoritmo 2.3: Método pop da classe PilhaVetor

Implementação de Pilha com Lista

Classe PilhaLista

- elementos da pilha armazenados na lista
- a classe implementa uma “fachada” para lista, implementando a interface Pilha

```
public class PilhaLista implements Pilha {

    private NoLista l;

    // implementacao de métodos da interface Pilha
}
```

Implementação de Pilha com Lista

- Método construtor, chama o construtor da classe lista

Algoritmo: PilhaLista(int tam)

this.topo $\leftarrow$ *null*;

Algoritmo 2.5: Construtor da classe PilhaLista

Implementação de Pilha com Lista

- Método push: insere elemento no início da lista

```
public void push(int v) throws Exception {
    insereInicio(v);
}
```

Implementação de Pilha com Lista

- Método pop: retira o primeiro elemento da lista

```
public int pop() throws Exception {
    if (!vazia()) {
       // retire o primeiro nó da lista;
       // retorne a informação armazenada no nó;
    } else {
        throw new Exception("ERRO: pilha vazia!");
    }
}
```

Exemplo: calculadora

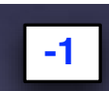
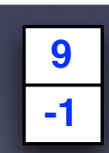
Notação para expressões aritméticas

- infixa, operador entre operandos: (1-2)*(4+5)
- pós-fixa, operador após operandos: 1 2 - 4 5 + *
- pré-fixa, operador antes dos operandos: * - 1 2 + 4 5

Exemplo: calculadora

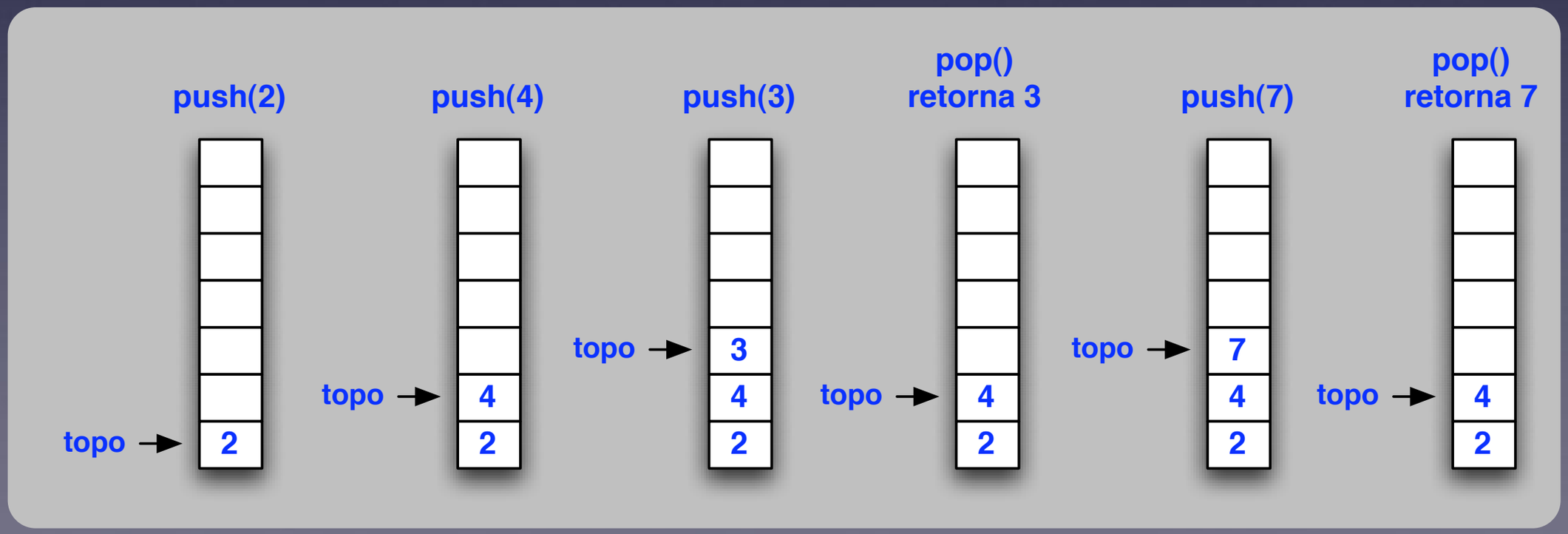
Avaliação de expressões aritméticas pós-fixadas:

- cada operando é empilhado numa pilha de valores
- quando encontra um operador:
  - desempilha-se o número apropriado de operandos
  - realiza-se a operação devida
  - empilha-se o resultado
- Avaliação da expressão: 1 2 - 4 5 + *

empilha valores 1 e 2	1 2 - 4 5 + *	
quando aparece operador “-”	1 2 - 4 5 + *	
desempilhe 1 e 2		
empilhe -1 : resultado da operação (1 - 2)		
empilhe os valores 4 e 5	1 2 - 4 5 + *	
quando aparecer o operador “+”	1 2 - 4 5 + *	
desempilhe 4 e 5		
empilhe 9 : resultado da operação (4 + 5)		
quando aparecer o operador “*”	1 2 - 4 5 + *	
desempilhe -1 e 9		
empilhe -9 : resultado da operação (-1 * 9)		

Resumo: Pilha

- top: retorna o topo da pilha
- push: insere novo elemento no topo
- pop remove e retorna o elemento do topo da pilha



3-Pilhas

CCA0916 - ESTRUTURA DE DADOS I
BCC

Prof. Dr. Paulo César Rodacki Gomes
paulo.gomes@ifc.edu.br

Blumenau, 2017