

# Estrutura de Dados Pilha

Prof. Helder Pestana

# Pilha

- Pilha (LIFO -LastIn, FirstOut):
- A pilha é uma estrutura de dados fundamental em algoritmos e programas.
- Funciona com o princípio LIFO, ou seja, o último elemento inserido é o primeiro a ser retirado.
- Analogia com uma pilha de livros: o último livro colocado no topo é o primeiro a ser retirado.
- Outra analogia seria o carregamento de um caminhão de entregas: se aplicarmos a lógica da pilha, a última entrega colocada no caminhão deve ser a primeira a sair. Isso evita a necessidade de reorganizar as entregas durante o processo de descarregamento.

# Pilha

- Pilhas e filas são casos especiais das listas encadeadas.
- Ambas possuem regras rigorosas para acessar os dados armazenados nelas.
- As operações de recuperação de dados são destrutivas, ou seja, para se alcançar dados intermediários a tais estruturas, é necessário destruir sequencialmente os dados anteriores.
- É uma das estruturas de dados mais simples
- É a estrutura de dados mais utilizada em programação, sendo inclusive implementada diretamente pelo hardware da maioria das máquinas modernas.
- A idéia fundamental da pilha é que todo o acesso a seus elementos é feito através do seu topo.
- Assim, quando um elemento novo é introduzido na pilha, passa a ser o elemento do topo, e o único elemento que pode ser removido da pilha é o do topo.

# Atividades realizadas em computadores que utilizam estruturas do tipo pilha:

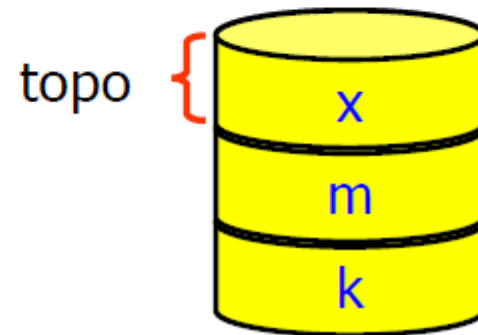
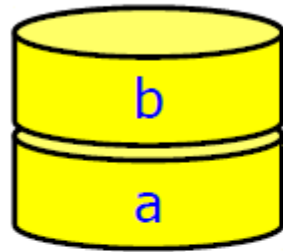
- a) Controle feito pelo sistema operacional a chamadas sucessivas a procedimentos ou funções de uma aplicação (pilha de execução). Essas chamadas, referenciadas por endereços de memória, são empilhadas, quando requisitadas, e desempilhadas à medida que finalizam a execução, permitindo retornar ao ponto de origem no programa principal.
- b) Compiladores utilizam estruturas do tipo pilha para a transformação de expressões da notação infixa, em que os operadores estão entre os operandos, e para a pós-fixa, em que os operadores são colocados após os operandos, facilitando a geração do código.
- c) Implementação no próprio hardware, como em calculadoras programáveis que usam a notação pós-fixa ou polonesa.

# Pilha

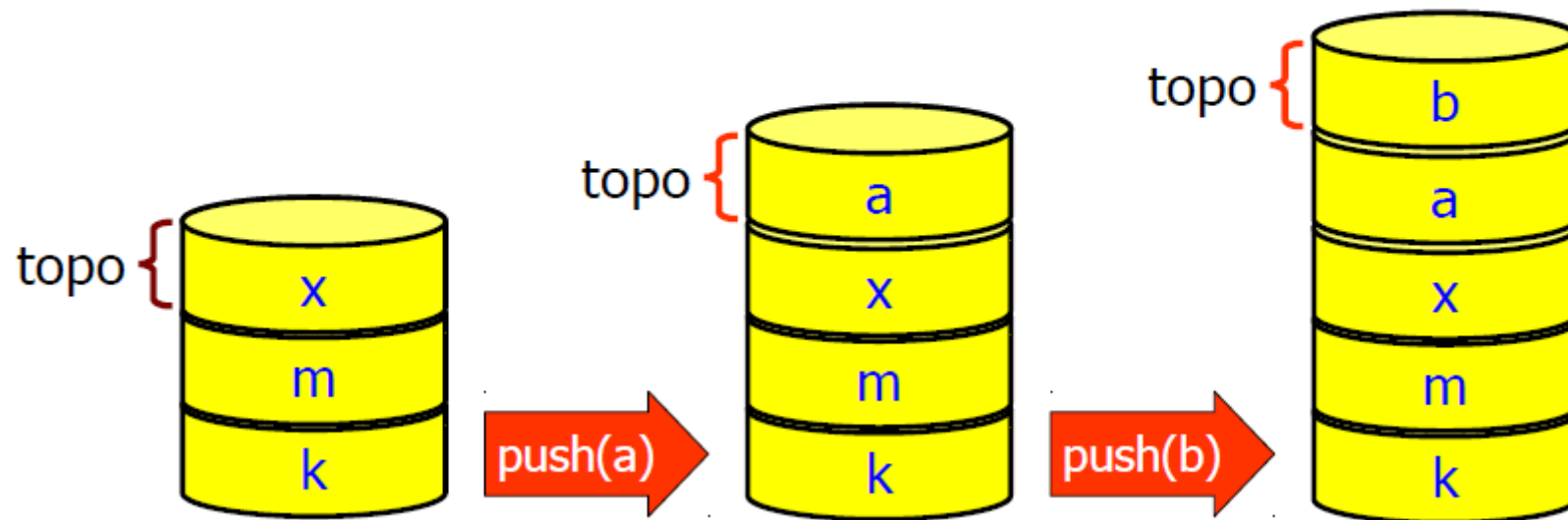
- Os elementos da pilha são retirados na ordem inversa à ordem em que foram introduzidos: o primeiro que sai é o último que entrou (LIFO – last in, first out).
- Existem duas operações básicas que devem ser implementadas numa estrutura de pilha:
  - operação para **empilhar** (**push**) um novo elemento, inserindo no topo,
  - operação para **desempilhar** (**pop**) um elemento, removendo-o do topo

# Push

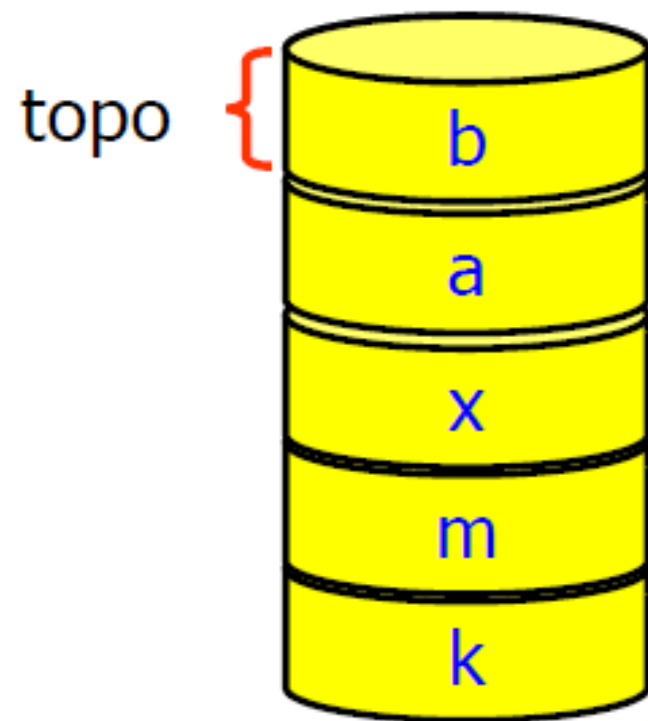
**push(b)**



# Push

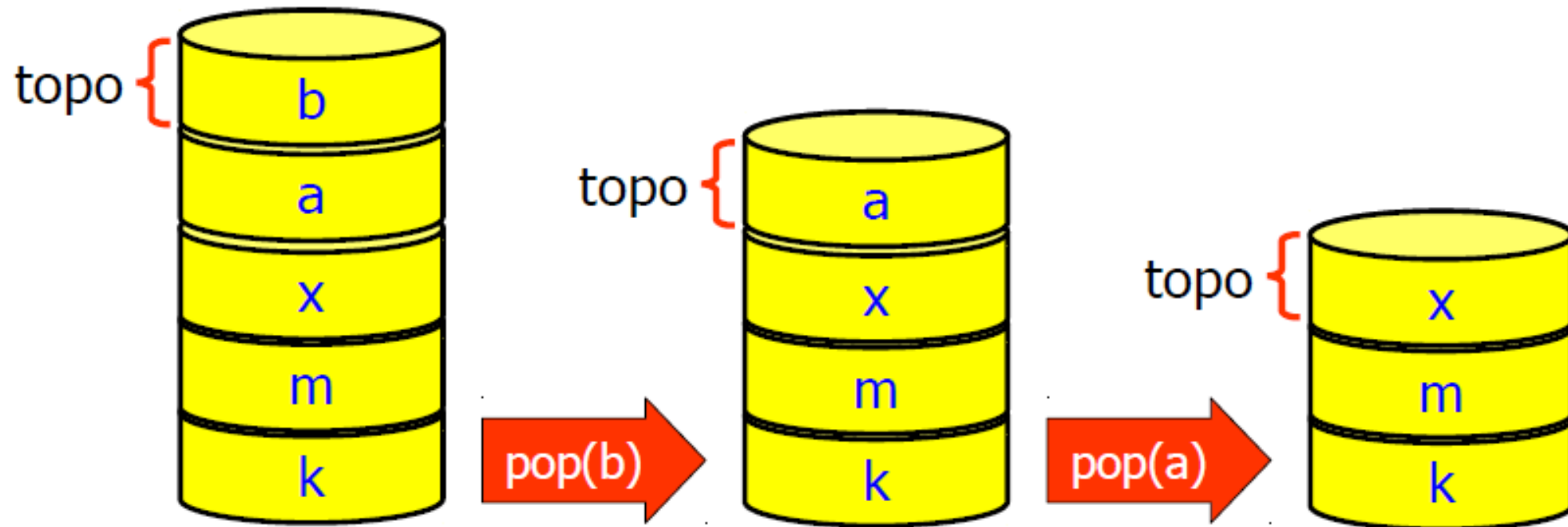


pop(~~a~~)





# Pop



# Implementação de pilha usando vetor




- Supondo uma pilha armazenada em um vetor `pilha[0..n-1]`.
- Considerando que os elementos são inteiros (isso é só um exemplo, os elementos de pilha poderiam ser quaisquer outros objetos).
- A parte do vetor ocupada pela pilha será:



# Operações Básicas




- Criar uma estrutura de pilha;
- Inserir um elemento no topo (push);
- Remover o elemento do topo (pop);
- Verificar se a pilha está vazia;
- Liberar a estrutura de pilha

# Exemplo 1 (Nó)

 No.java  
 Pilha.java  
 palindromo.java

```
class No {  
    char dado;  
    No proximo;  
  
    public No(char dado) {  
        this.dado = dado;  
        this.proximo = null;  
    }  
}
```

# Exemplo 1 (Pilha)

 No.java  
 Pilha.java  
 palindromo.java

```
class Pilha {  
    private No topo;  
  
    public Pilha() {  
        this.topo = null;  
    }  
  
    public void empilhar(char dado) {  
        No novoNo = new No(dado);  
        novoNo.proximo = topo;  
        topo = novoNo;  
    }  
  
    public char desempilhar() {  
        if (topo == null) {  
            throw new RuntimeException("Pilha vazia");  
        }  
        char valor = topo.dado;  
        topo = topo.proximo;  
        return valor;  
    }  
  
    public boolean estaVazia() {  
        return topo == null;  
    }  
}
```

# Exemplo 1 (palindromo)



No.java



Pilha.java



palindromo.java

```
public class palindromo {  
    public static boolean ehPalindromo(String palavra) {  
        Pilha pilha = new Pilha();  
        String palavraInvertida = "";  
  
        // Empilhamos os caracteres da palavra  
        for (char letra : palavra.toCharArray()) {  
            pilha.empilhar(letra);  
        }  
  
        // Desempilhamos e construímos a palavra invertida  
        while (!pilha.estaVazia()) {  
            palavraInvertida += pilha.desempilhar(); // Concatenando caracteres  
        }  
  
        // Compara a palavra original com a invertida  
        return palavra.equals(palavraInvertida);  
    }  
  
    public static void main(String[] args) {  
        System.out.println(ehPalindromo("radar")); // true  
        System.out.println(ehPalindromo("java")); // false  
        System.out.println(ehPalindromo("arara")); // true  
    }  
}
```

# Nó genérico

// Armazena qualquer tipo de dado.

```
class No<T> {  
    T dado;  
    No<T> proximo;  
  
    public No(T dado) {  
        this.dado = dado;  
        this.proximo = null;  
    }  
}
```

```
class No {  
    char dado;  
    No proximo;  
  
    public No(char dado) {  
        this.dado = dado;  
        this.proximo = null;  
    }  
}
```

```
// Definindo a classe Pilha como genérica
class Pilha<T> {
    private No<T> topo; // A pilha terá um topo que é do tipo No<T>

    public Pilha() {
        this.topo = null;
    }

    // Empilhando um valor do tipo T
    public void empilhar(T dado) {
        No<T> novoNo = new No<>(dado);
        novoNo.proximo = topo;
        topo = novoNo;
    }

    // Desempilhando um valor do tipo T
    public T desempilhar() {
        if (topo == null) {
            throw new RuntimeException("Pilha vazia");
        }
        T valor = topo.dado;
        topo = topo.proximo;
        return valor;
    }

    // Verificando se a pilha está vazia
    public boolean estavazia() {
        return topo == null;
    }
}
```

Pilha Genérico

```
class Pilha {
    private No topo;

    public Pilha() {
        this.topo = null;
    }

    public void empilhar(char dado) {
        No novoNo = new No(dado);
        novoNo.proximo = topo;
        topo = novoNo;
    }

    public char desempilhar() {
        if (topo == null) {
            throw new RuntimeException("Pilha vazia");
        }
        char valor = topo.dado;
        topo = topo.proximo;
        return valor;
    }

    public boolean estaVazia() {
        return topo == null;
    }
}
```