

Universidade Federal da Paraíba

Centro de Informática

---

Departamento de Informática

# Estrutura de Dados

## Pilhas

▶ Tiago Maritan

▶ [tiago@ci.ufpb.br](mailto:tiago@ci.ufpb.br)

---

# Conteúdos Abordados

---

- ▶ O Conceito de Pilha
- ▶ Pilhas com Representação Sequencial
- ▶ Pilhas com Representação Dinâmica

# Tipos Particulares de Listas

---

## ▶ **Pilha**

- ▶ Inserções e remoções ocorrem somente em um extremo da lista
- ▶ **LIFO: Last In First Out**

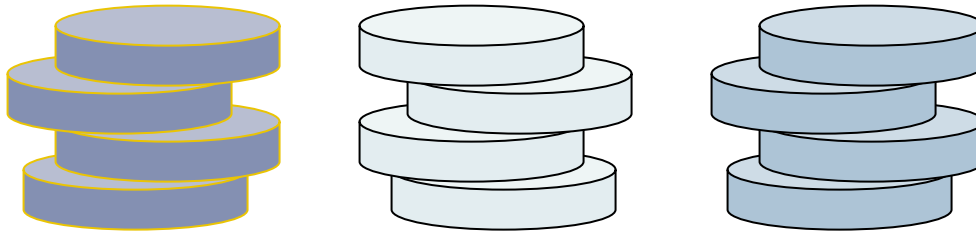
## ▶ **Fila**

- ▶ Inserções são realizadas em um extremo e remoções em outro extremo da lista.
- ▶ **FIFO: First In First Out**

# Definição de Pilha

---

- ▶ **Ideia fundamental:** Acesso a elementos da pilha é sempre feito através do seu **topo**.
- ▶ Quando um **elemento novo** é **introduzido** na pilha, ele passa a ser o elemento do **topo**.
- ▶ O único elemento que pode ser **removido da pilha** é o do **topo**.
  - ▶ Metáfora da pilha de pratos

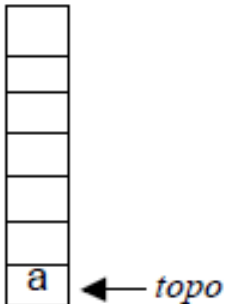


# Operações

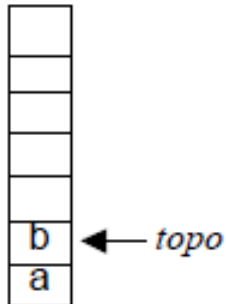
---

- ▶ **Inserção e remoção acontecem no topo da pilha.**
- ▶ Operações:
  - ▶ push (empilhar) e pop (desempilhar)

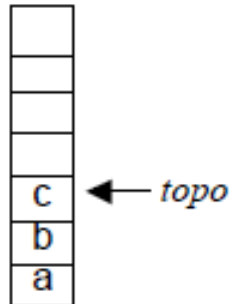
push (a)



push (b)

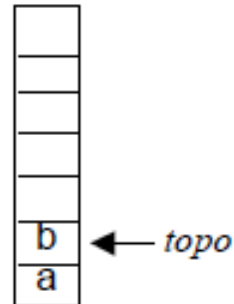


push (c)

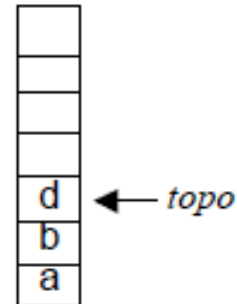


pop ()

*retorna-se C*

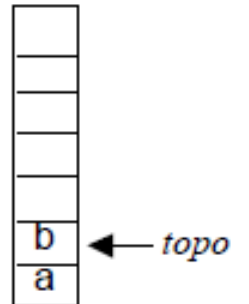


push (d)



pop ()

*retorna-se d*



# Aplicações de Pilha

---

- ▶ Estrutura de dados mais utilizada em programação
  - 1. Implementada diretamente pelo HW da maioria das máquinas modernas.
  - 2. Implementação de compiladores
    - ▶ Pilha de execução de funções chamadas
    - ▶ Avaliação de expressões
  - 3. Navegadores web
    - ▶ Usam pilhas para armazenar os endereços mais recentemente visitados.
  - 4. Mecanismo de reversão de operações (“undo”) dos editores de texto
    - ▶ Armazena as alterações em uma pilha



# Pilhas

---

- ▶ Definição:

Dada uma pilha  $P = (a(1), a(2), \dots, a(n))$ , dizemos que:

- ▶  $a(1)$  é o elemento da base da pilha;
- ▶  $a(n)$  é o elemento topo da pilha; e
- ▶  $a(i + 1)$  está acima de  $a(i)$ .

# Formas de Representação de Pilhas

---

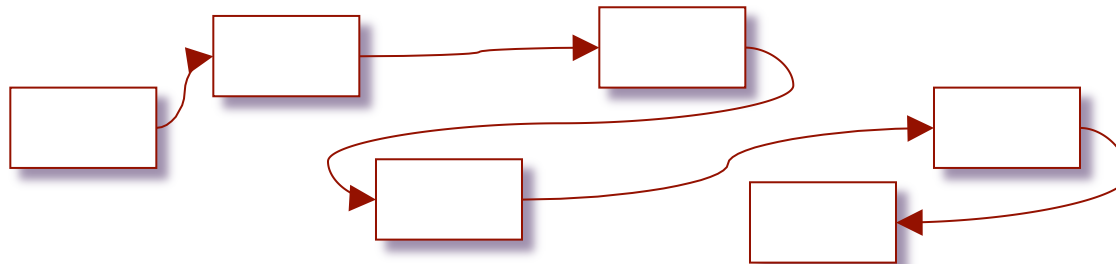
## ▶ Alocação Sequencial

- ▶ Elementos dispostos em posições contíguas de memória



## ▶ Alocação Encadeada

- ▶ Elementos dispostos aleatoriamente na memória, encadeados por ponteiros





---

# Pilhas com Representação Sequencial

# Implementação de Pilhas Sequenciais

---

## ▶ Interface:

- ▶ Criar uma pilha vazia;
- ▶ Testar se a pilha está vazia;
- ▶ Verificar se a pilha está cheia;
- ▶ Obter o tamanho da pilha;
- ▶ Consultar o elemento do topo da pilha (*top*);
- ▶ Inserir um novo elemento no topo da pilha (*push*);
- ▶ Remover o elemento do topo da pilha (*pop*).

# Implementação de Pilhas Seqüenciais

---

```
public class PilhaSeq {  
  
    private int dados[];  
    private int topo;  
    private int tamMax;  
  
    public PilhaSeq() {  
        tamMax = 100;  
        dados = new int[tamMax];  
        topo = -1;  
    }  
  
    ...  
}
```

# Implementação de Pilhas Seqüenciais

---

```
/** Definição das Operações */  
  
/** Verifica se a Pilha está vazia */  
public boolean vazia() {  
    if (topo == -1)  
        return true;  
    else  
        return false;  
}  
  
// continua...
```

# Implementação de Pilhas Seqüenciais

---

```
/**Verifica se a Pilha está cheia */  
public boolean cheia() {  
    if (topo == (tamMax-1))  
        return true;  
    else  
        return false;  
}  
  
/**Obtém o tamanho da Pilha*/  
public int tamanho() {  
    return topo+1;  
}  
  
// continua...
```

# Implementação de Pilhas Seqüenciais

---

```
/** Retorna o elemento do topo da Pilha.  
    -1 se a pilha estiver vazia. */  
public int top () {  
    if (vazia())  
        return -1; // pilha vazia  
  
    return dados[topo];  
}
```

# Implementação de Pilhas Seqüenciais

---

```
/** Insere um elemento no topo da pilha.  
    Retorna false se a pilha estiver cheia.  
    Caso contrário retorna true */  
public boolean push(int valor) {  
    if (cheia())  
        return false; // err: pilha cheia  
  
    topo++;  
    dados[topo] = valor;  
    return true;  
}
```

# Implementação de Pilhas Seqüenciais

---

```
/** Retira o elemento do topo da pilha.  
    Retorna -1 se a pilha estiver vazia. */  
public int pop() {  
    if (vazia())  
        return -1; // Pilha vazia  
  
    int valor = dados[topo];  
    topo--;  
    return valor;  
}
```



---

# Pilhas Encadeadas

# Implementação de Pilhas Encadeadas

---

## ▶ Interface:

- ▶ Criar uma pilha vazia;
- ▶ Testar se a pilha está vazia;
- ▶ Obter o tamanho da pilha;
- ▶ Consultar o elemento do topo da pilha (*top*);
- ▶ Inserir um novo elemento no topo da pilha (*push*);
- ▶ Remover o elemento do topo da pilha (*pop*).

# Implementação de Pilhas Encadeadas

---

```
public class No{  
    private int conteudo;  
    private No prox;  
  
    public No(){  
        prox = null;  
    }  
  
    // Métodos get e set  
}
```

```
public class Pilha{  
    private No topo;  
    private int nElementos;  
  
    public Pilha(){  
        topo = null;  
        nElementos = 0;  
    }  
}
```

# Implementação de Pilhas Encadeadas

---

```
/** Verifica se a Pilha está vazia*/  
public boolean vazia () {  
    if (nElementos == 0)  
        return true;  
    else  
        return false;  
}  
  
/** Obtém o tamanho da Pilha*/  
public int tamanho() {  
    return nElementos;  
}
```

# Implementação de Pilhas Encadeadas

---

```
/** Consulta o elemento do topo da Pilha
    Retorna -1 se a pilha estiver vazia.*/
public int top () {
    if (vazia()) {
        return -1; // Pilha vazia
    }

    return topo.getConteudo();
}

// continua...
```

# Implementação de Pilhas Encadeadas

---

```
/** Insere um elemento no topo da pilha.  
Retorna true se a insercao funcionar*/  
public boolean push(int valor) {  
    No novoNo = new No();  
    novoNo.setConteudo(valor);  
  
    // Faz o novo no apontar pro atual topo da pilha  
    novoNo.setProx(topo);  
  
    // Atualiza o topo da pilha para o novo nó  
    topo = novoNo;  
  
    nElementos++;  
    return true;  
}
```

# Implementação de Pilhas Encadeadas

---

```
/** Retira o elemento do topo da pilha.  
Retorna -1 se a pilha estiver vazia.  
Caso contrário retorna o valor removido */  
public int pop () {  
    if (vazia())  
        return -1; // pilha vazia  
    No p = topo;  
    int valor = p.getConteudo();  
  
    topo = p.getProx();  
    nElementos--;  
  
    p= null;  
    return valor;  
}
```

Universidade Federal da Paraíba

Centro de Informática

---

Departamento de Informática

# Estrutura de Dados

## Pilhas

▶ Tiago Maritan

▶ [tiago@ci.ufpb.br](mailto:tiago@ci.ufpb.br)

