EaD
UFMS DIGITAL
Graduação

AGEAD
Agência de Educação
Digital e a Distância

UFMS

Algoritmos e Programação II

Prof. Dr. Rafael dos Passos Canteri

EaD
UFMS DIGITAL
Graduação

Módulo 3 - Estruturas de dados

Unidade 2 - Pilhas e Filas

UFMS
AGEAD
Agência de Educação
Digital e a Distância

Pilha

Fonte: OpenClipArt

O que é uma Pilha?

- A Pilha é uma estrutura de dados linear que segue o princípio LIFO — *Last In, First Out* (o último a entrar é o primeiro a sair).
- Ou seja, os elementos são inseridos e removidos do topo da Pilha.

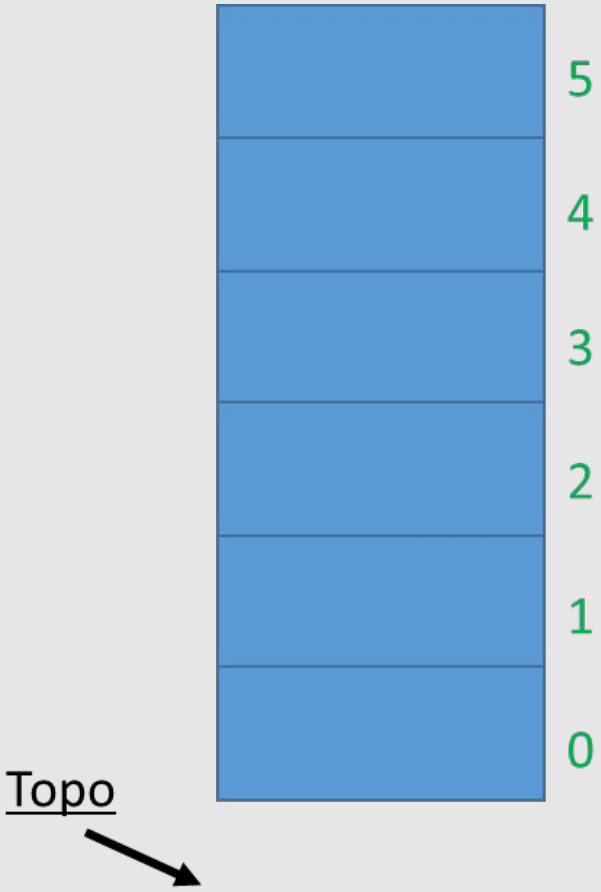
Utilidade da Pilha

- O conceito de Pilha é amplamente utilizado em diversas situações, como:
  - Execução de sub-rotinas;
  - Conversão de bases numéricas;
  - Avaliação de expressões matemáticas;
  - Desfazer/refazer em editores de texto.

Operações da Pilha

- *Push* (Inserir).
- *Pop* (Remover).
- *Top* (Buscar).

Inserção na Pilha



Quantidade: 0

- A Pilha se encontra vazia, isto é, não possui elementos.
- Assim, o marcador de topo está fora da Pilha.
- E se quisermos inserir um elemento qualquer.
- Por exemplo, 2.7?

Inserção na Pilha

Quantidade: 1

- Nesse caso, o marcador do topo sobe uma posição e o elemento é inserido no topo da Pilha.
- E se quisermos inserir mais um elemento.
- Por exemplo, -8.5?

Inserção na Pilha

Quantidade: 2

- Segue-se o mesmo processo até que não se deseje mais inserir elementos.

Remoção na Pilha

Quantidade: 5

- E se quisermos remover um elemento qualquer?
- Por exemplo -8.5.

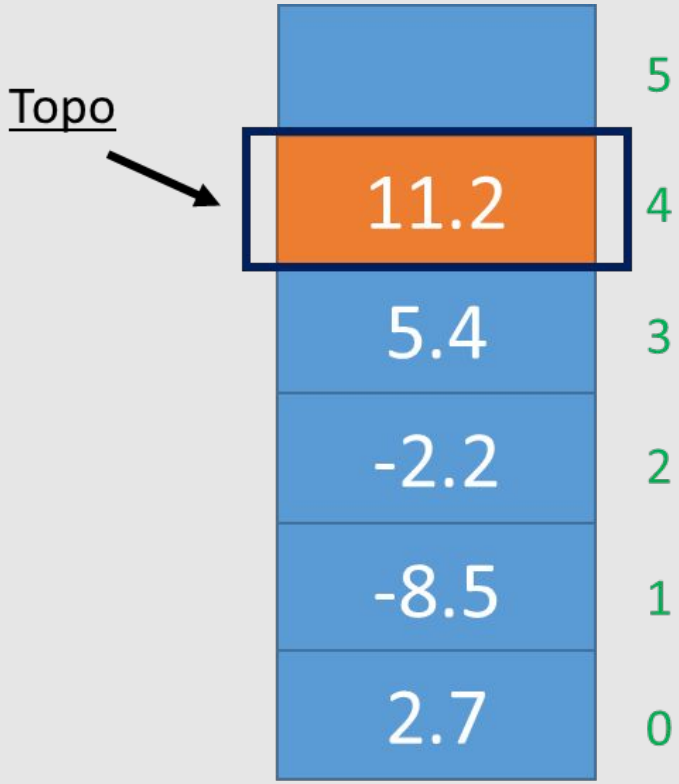
  NÃO PODEMOS!
- Todas as operações de inserção e remoção sempre acontecem no topo da Pilha!

Remoção na Pilha

Quantidade: 4

- O marcador do topo desce uma posição.

Busca na Pilha



- Só pode ser realizada no topo da Pilha.
- Implementação mais simples do que em uma Lista.
- Basta acessar o elemento do topo.

Implementação de Pilha

```
class Pilha:
    def __init__(self, tamanho):
        self.tamanho = tamanho          # Tamanho máximo da Pilha
        self.dados = [None] * tamanho # Array fixo
        self.topo = -1                  # Índice do topo da Pilha
```

Implementação de Pilha

```
def push(self, elemento):
    if self.topo < self.tamanho - 1:
        self.topo += 1
        self.dados[self.topo] = elemento
        print(f"Elemento {elemento} inserido.")
    else:
        print("Erro: Pilha cheia!")
```

Implementação de Pilha

```
def pop(self):
    if self.topo >= 0:
        elemento = self.dados[self.topo]
        self.dados[self.topo] = None
        self.topo -= 1
        print(f"Elemento {elemento} removido.")
        return elemento
    else:
        print("Erro: Pilha vazia!")
        return None
```

Implementação de Pilha

```
def top(self):
    if self.topo >= 0:
        return self.dados[self.topo]
    else:
        print("Pilha vazia!")
        return None
```

Implementação de Pilha

```
def imprimir(self):
    print("Elementos da Pilha:", self.dados)
```

Implementação de Pilha

```
# Exemplo de uso
pilha = Pilha(5)
pilha.push(10)
pilha.push(20)
pilha.push(30)
pilha.imprimir()
print("Topo da pilha:", pilha.top())
pilha.pop()
pilha.imprimir()
```

Fila

Fonte: Flickr

O que é uma Fila?

- A Fila é uma estrutura de dados linear que segue o princípio FIFO – *First In, First Out* (o primeiro a entrar é o primeiro a sair).
- Ou seja, os elementos são inseridos no final e removidos do início, assim como em uma fila no mundo real.

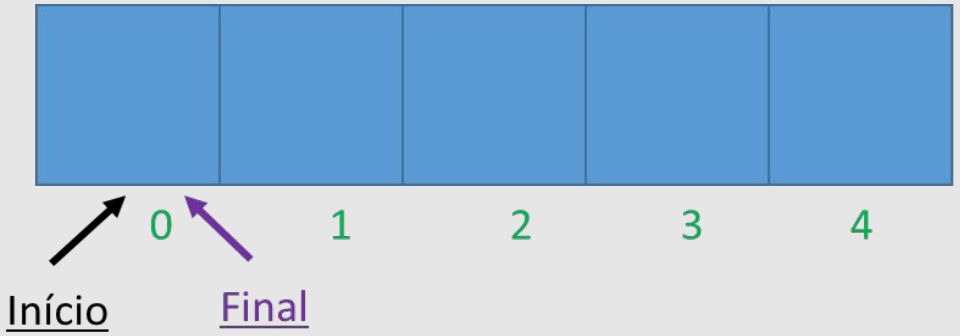
Utilidade da Fila

- Filas de impressão.
- Escalonamento de processos.
- Comunicação entre sistemas.
- Transmissão de pacotes de rede.

Operações da Fila

- Enfileirar (Inserir).
- Desenfileirar (Remover).
- Primeiro (Buscar).

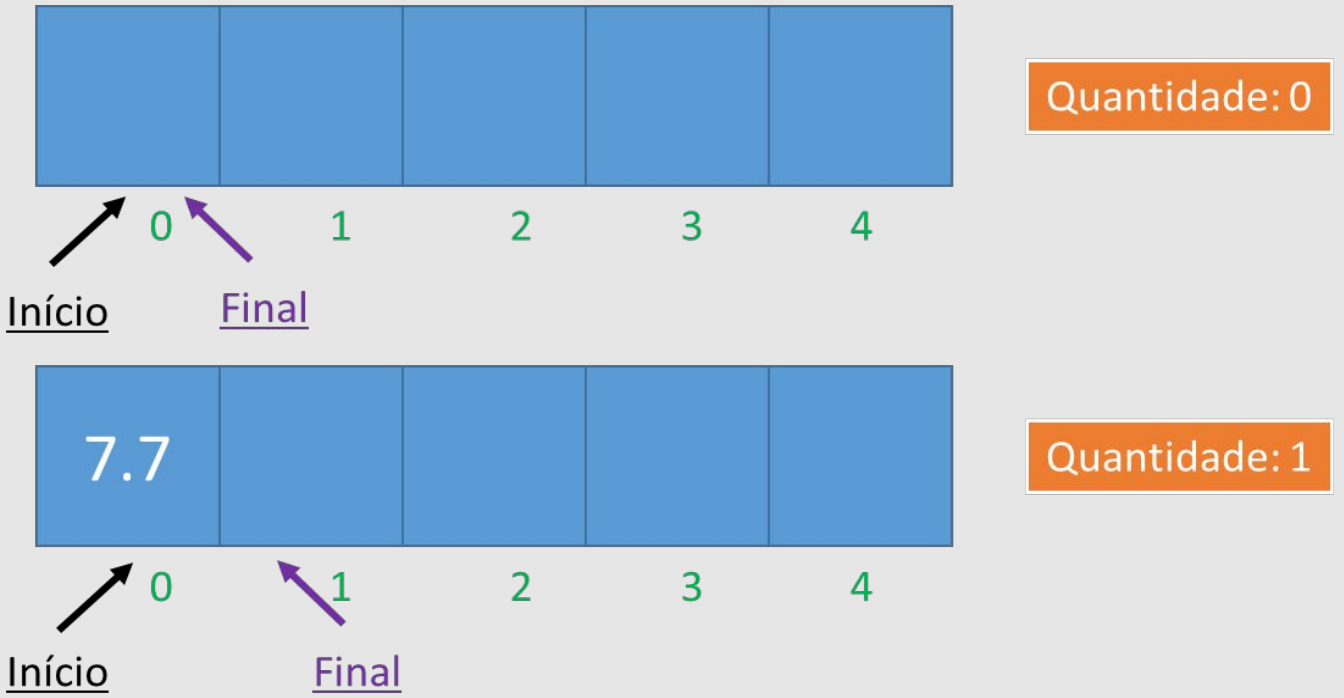
Inserção na Fila



Quantidade: 0

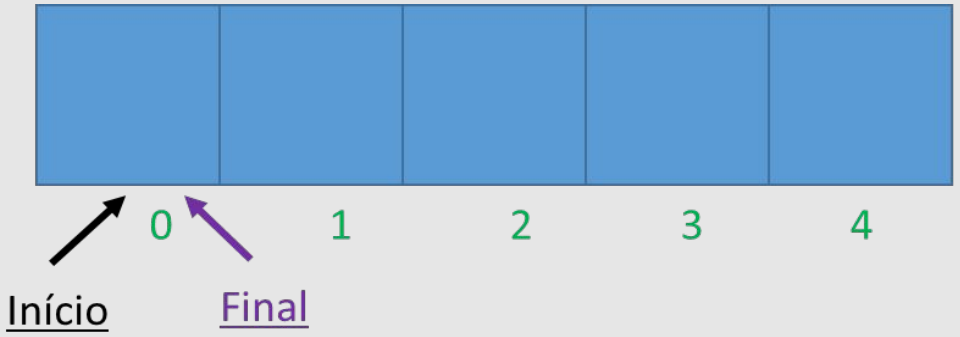
- A Fila se encontra vazia, isto é, não possui elementos.
- Assim, o início e final dela estão na mesma posição.
- E se quisermos inserir um elemento qualquer, por exemplo, 7.7?

Inserção na Fila

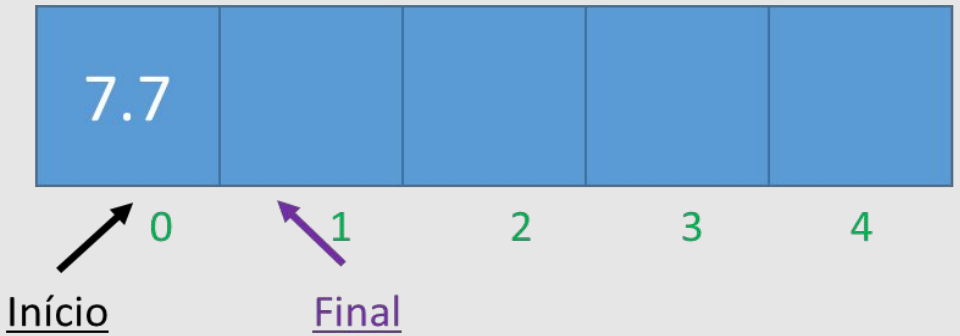


- E se quisermos inserir mais um elemento qualquer, por exemplo, 1.1?

Inserção na Fila



Quantidade: 0



Quantidade: 1

Quantidade: 2

Inserção na Fila

- Segue-se o mesmo processo até que não se deseje mais inserir elementos.

Inserção na Fila

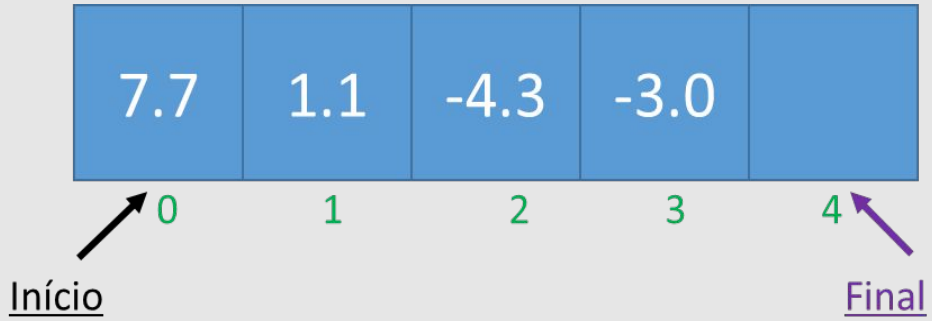
Quantidade: 2

- E se quisermos inserir um elemento entre o 7.7 e o 1.1?

NÃO PODEMOS!

- A inserção sempre acontece no final da Fila!

Remoção na Fila



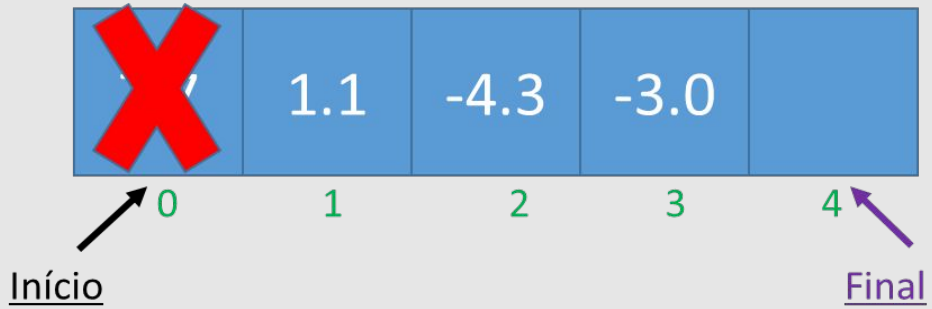
Quantidade: 4

- E se quisermos remover um elemento qualquer? Por exemplo -4.3.

NÃO PODEMOS!

- A remoção sempre acontece a partir do início da Fila!

Remoção na Fila



Quantidade: 4

Remoção na Fila

Busca na Fila

- Só pode ser realizada no início da Fila.

- Implementação mais simples do que em uma Lista.
  - Basta acessar o elemento do início.

Implementação de Fila

```
class Fila:
    def __init__(self, tamanho):
        self.tamanho = tamanho
        self.dados = [None] * tamanho      # Array fixo
        self.inicio = 0                    # Índice onde removemos
        self.fim = 0                       # Índice onde inserimos
        self.quantidade = 0                # Quantidade de elementos na Fila
```

Implementação de Fila

```
def enfileirar(self, elemento):
    if self.quantidade < self.tamanho:
        self.dados[self.fim] = elemento
        self.fim += 1
        self.quantidade += 1
        print(f"Elemento {elemento} inserido.")
    else:
        print("Erro: Fila cheia!")
```

Implementação de Fila

```
def desenfileirar(self):
    if self.quantidade > 0:
        elemento = self.dados[self.inicio]
        self.dados[self.inicio] = None
        self.inicio += 1
        self.quantidade -= 1
        return elemento
    else:
        print("Erro: Fila vazia!")
        return None
```

Implementação de Fila

```
def primeiro(self):
    if self.quantidade > 0:
        return self.dados[self.inicio]
    else:
        print("Fila vazia!")
        return None
```

Implementação de Fila

```
def imprimir(self):
    print("Elementos da Fila:", self.dados)
```

Implementação de Fila

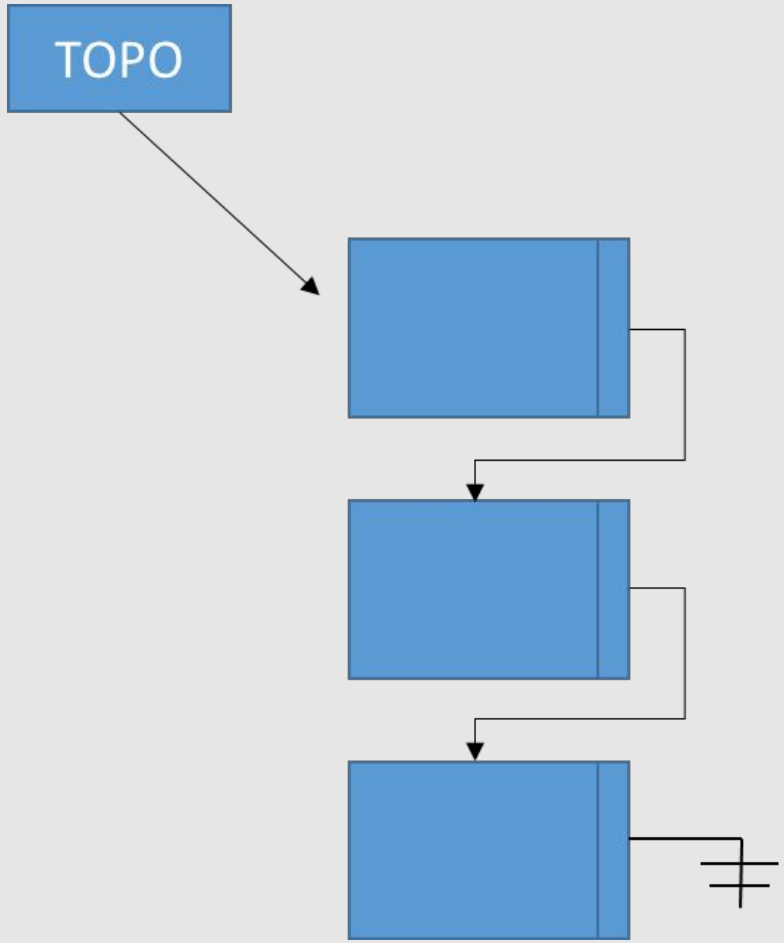
```
# Exemplo de uso
fila = Fila(5)
fila.enfileirar(10)
fila.enfileirar(20)
fila.enfileirar(30)
fila.imprimir()
print("Primeiro da fila:", fila.primeiro())
fila.desenfileirar()
fila.imprimir()
```

Estruturas sequenciais e encadeadas

- Assim como no caso das Listas, as Pilhas e as Filas também podem ser implementadas de maneira sequencial ou encadeada.

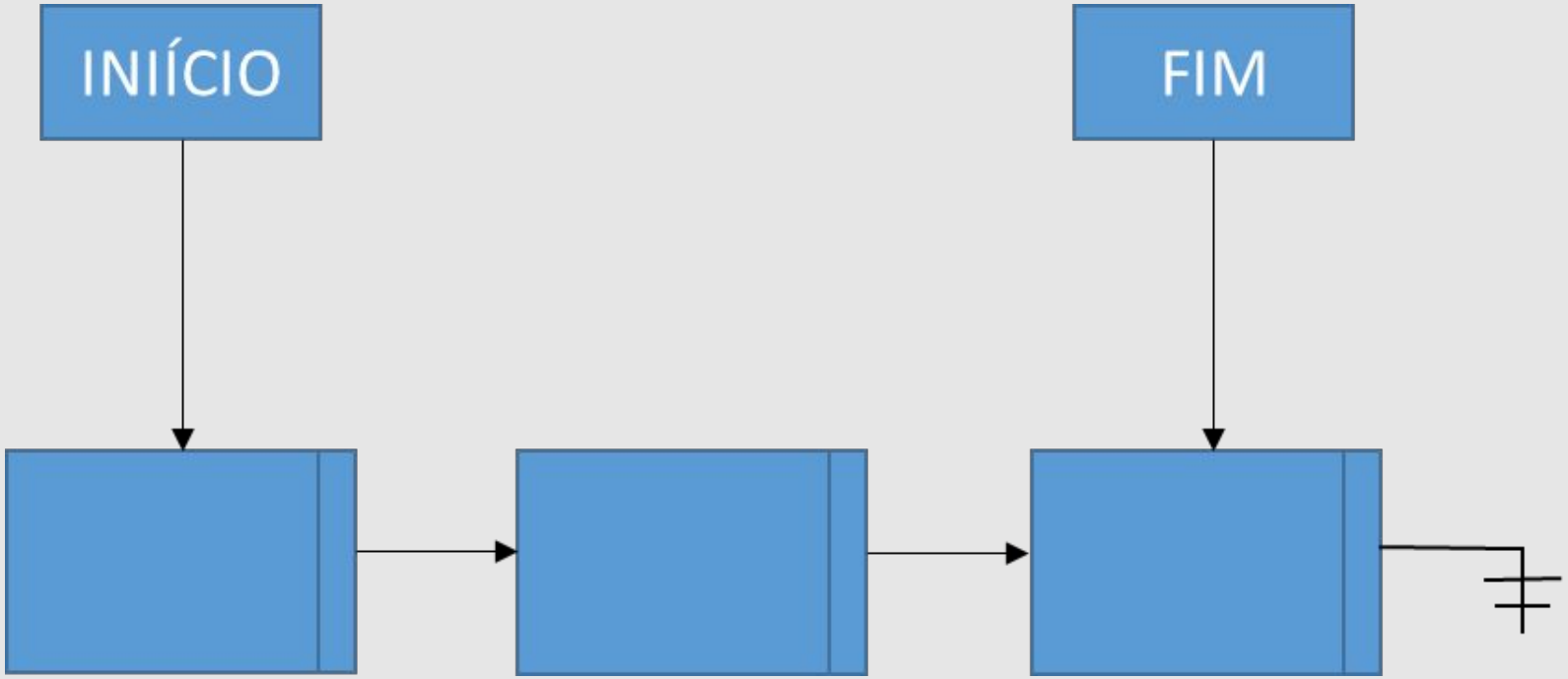
Estruturas encadeadas

- Pilha encadeada



Estruturas encadeadas

- Fila encadeada



Estruturas encadeadas - vantagens

- Pilha Encadeada
  - Sem limite fixo de tamanho;
  - Eficiência de inserção e remoção;
  - Uso dinâmico da memória.
- Fila Encadeada
  - Sem necessidade de gerenciamento de índices;
  - Crescimento flexível;
  - Evita deslocamentos e circularidade.

Estruturas encadeadas - desvantagens

- Pilha Encadeada
  - Mais complexa para implementar;
  - Pode ter *overhead*.
- Fila Encadeada
  - Necessidade de controle cuidadoso dos ponteiros;
  - Menor previsibilidade no uso da memória.

Lista vs Pilha vs Fila

Característica	Lista	Pilha	Fila
Ordem de acesso	Livre	LIFO	FIFO
Inserção	Em qualquer posição	No topo	No final
Remoção	De qualquer posição	Do topo	Do início
Busca	Em qualquer posição	Elemento do topo	Elemento do início

Lista vs Pilha vs Fila

Característica	Lista	Pilha	Fila
Uso comum	Manipulação geral de dados	Desfazer ações	Processamento de tarefas
Estrutura	*Array* e Encadeada	*Array* e Encadeada	*Array* e Encadeada
Complexidade	O(n)	O(1)	O(1)
Flexibilidade de acesso	Flexível (complexa)	Simples e reversa	Simples e em ordem

Recapitulação

- Conceitos e diferenças fundamentais entre as estruturas.
- Fila (FIFO) e Pilha (LIFO).
- Implementações sequenciais e encadeadas.
- Operações principais:
  - Pilha: *push*, *pop*, *top*.
  - Fila: enfileirar, desenfileirar, primeiro.

Referências

CORMEN, Thomas H.; LEISERSON, Charles E.; Ronald L. Rivest; et al. **Algoritmos**. 4. ed. Rio de Janeiro: GEN LTC, 2024. ISBN 9788595159914.

OPENCLIPART. OpenClipArt. [S.d.]. Disponível em: https://link.ufms.br/yDIR0. Acesso em: 6 jun. 2025.

SZWARCFITER, Jayme L.; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro: LTC, 2010. ISBN 978852162995-5.

Licenciamento

Respeitadas as formas de citação formal de autores de acordo com as normas da ABNT NBR 6023 (2018), a não ser que esteja indicado de outra forma, todo material desta apresentação está licenciado sob uma Licença Creative Commons - Atribuição 4.0 Internacional.

EaD

UFMS DIGITAL

Graduação