EaD
UFMS DIGITAL
Graduação

AGEAD
Agência de Educação
Digital e a Distância

UFMS

Algoritmos e Programação II

Prof. Dr. Rafael dos Passos Canteri

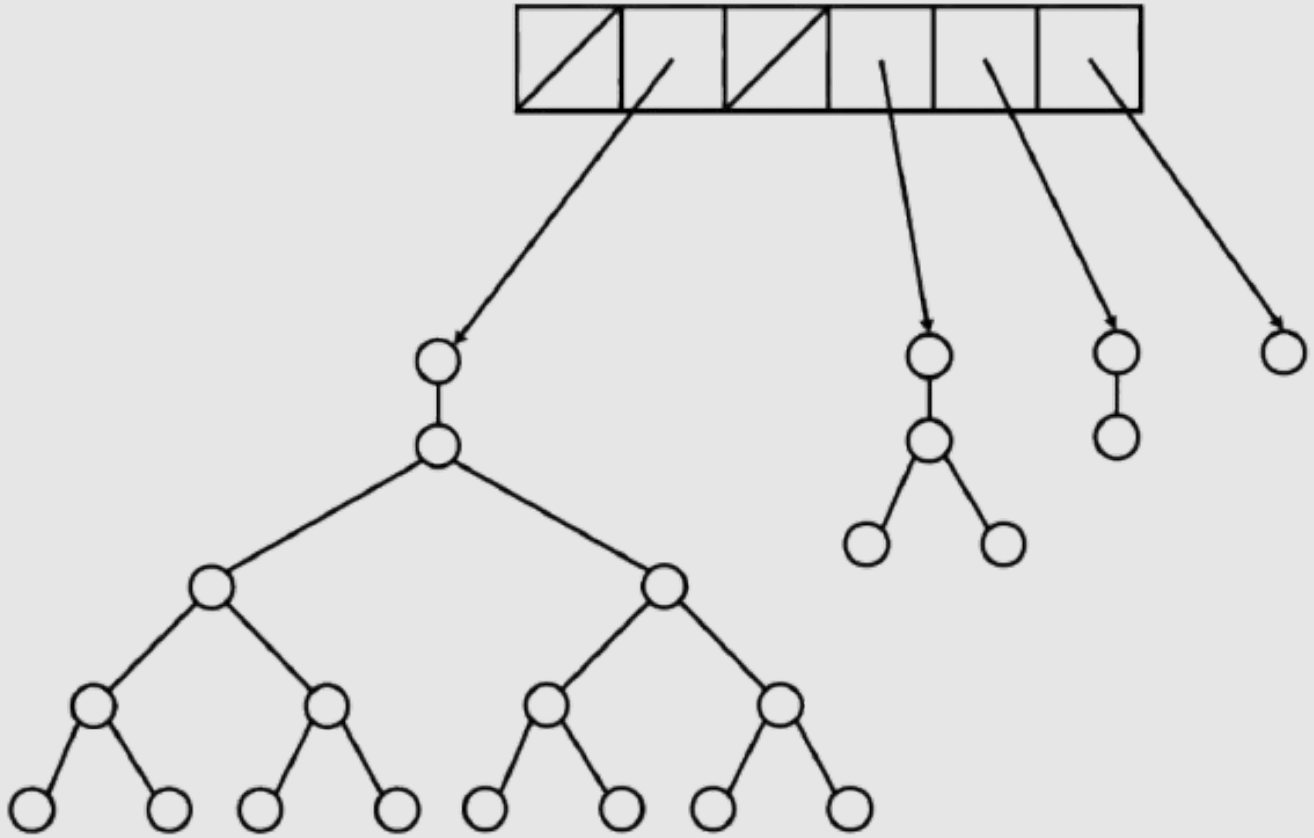
EaD
UFMS DIGITAL
Graduação

Módulo 3 - Estruturas de dados

Unidade 1 - Listas lineares

UFMS
AGEAD
Agência de Educação
Digital e a Distância

Estruturas de Dados



Fonte: Adaptado de Wikimedia Commons

Estruturas de Dados

- Também conhecidas como Tipos Abstratos de Dados (TADs).
- Estruturas de dados são formas específicas de organizar, armazenar e manipular dados em um computador.
- Tem o objetivo de tornar essas operações mais eficientes e adequadas ao problema que se deseja resolver.

Operações em Estruturas de Dados

- Uma estrutura de dados define regras claras para:
  - Inserção de dados (como e onde adicionar novos elementos),
  - Remoção de dados (como excluir e com que impacto),
  - Acesso e busca (como recuperar ou localizar valores).

Principais Estruturas de Dados

- Listas
- Pilhas
- Filas
- Grafos,
- Deques,
- *Heaps*,
- Árvores..

Fonte: PixaBay

O que são Listas

- Uma lista é uma estrutura de dados que armazena uma coleção ordenada de elementos.
- Cada elemento possui uma posição (índice), e pode ser acessado, modificado, inserido ou removido com base nessa posição.
- As listas podem ser sequenciais (como *arrays*) ou encadeadas (com ponteiros), dependendo de como estão implementadas.

Para que servem Listas?

- As listas são usadas para:
  - Guardar informações de forma organizada.
  - Iterar sobre elementos, como em laços.
  - Inserir e remover elementos facilmente.
  - Buscar dados com base em algum critério.
  - Implementar outras estruturas mais avançadas.

Operações em Listas

- Inserir (início, entre elementos, final)
- Remover (início, entre elementos, final)
- Buscar (elemento, posição)

Implementações de Listas

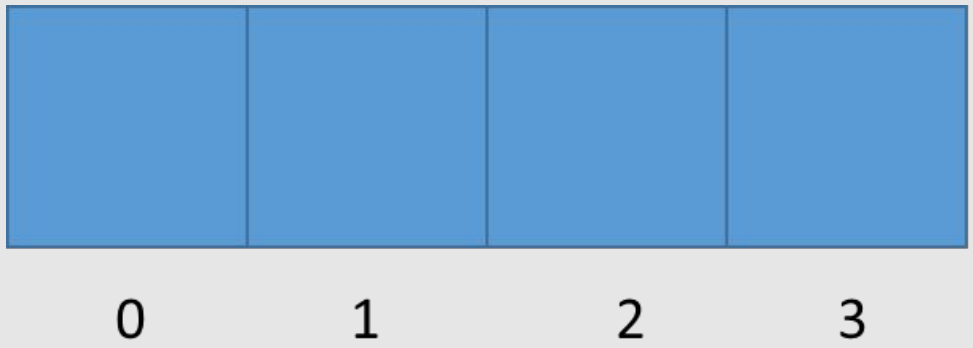
- Lista Sequencial (Linear)
  - Memória contígua: os elementos são armazenados em posições consecutivas.
  - Estrutura base: normalmente implementada usando vetores (*arrays*).
  - Tamanho fixo: geralmente exige definir a capacidade máxima antecipadamente.
  - Acesso rápido: acesso direto a qualquer elemento com índice.

Implementações de Listas

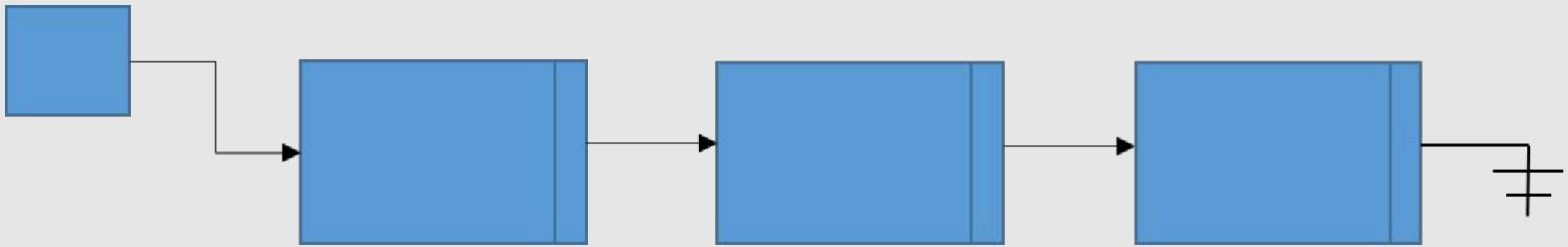
- Lista Encadeada (Dinâmica)
  - Memória não contígua: os elementos podem estar espalhados na memória.
  - Crescimento dinâmico: os nós são criados conforme necessário, sem limite fixo pré-definido.
  - Flexibilidade: mais fácil inserir ou remover elementos em qualquer posição.
  - Composição: cada nó possui um dado e um ponteiro (referência) para o próximo nó.

Implementações de Listas

- Sequencial

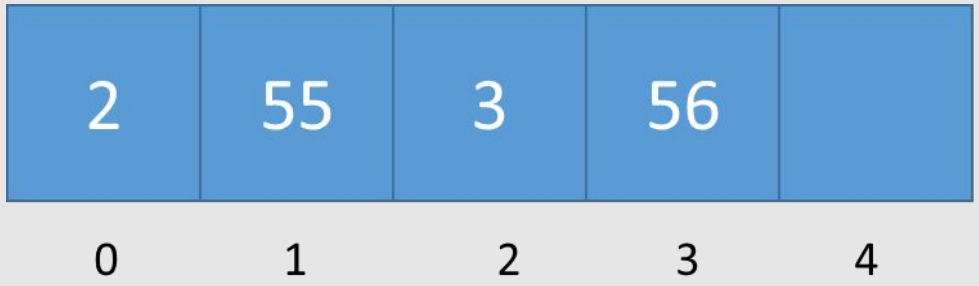


- Encadeada



Inserção - Lista sequencial

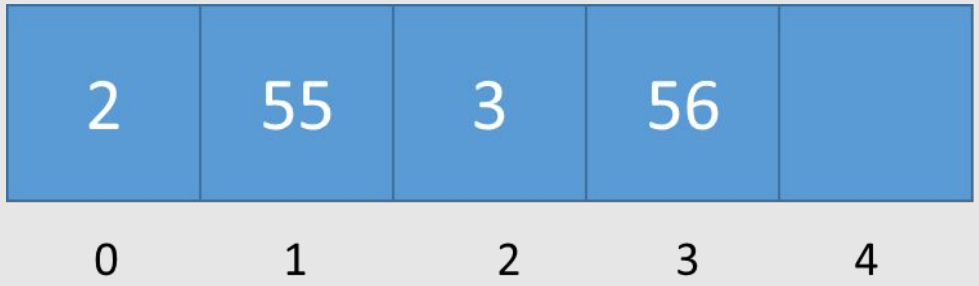
- No fim:



Quantidade: 4

Inserção - Lista sequencial

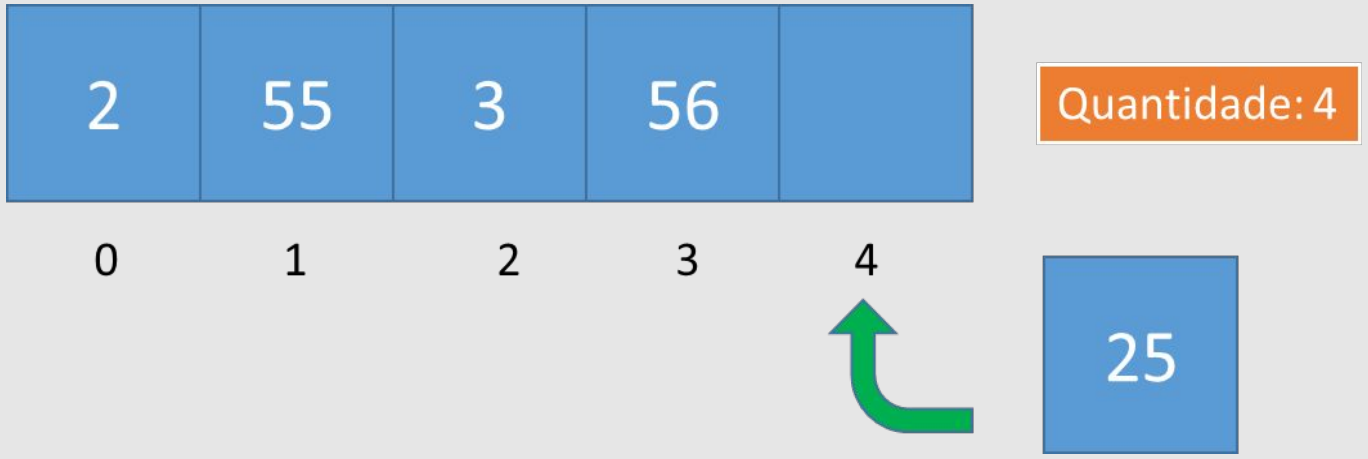
- No fim:



- E se quisermos inserir o elemento 25 no fim da Lista?

Inserção - Lista sequencial

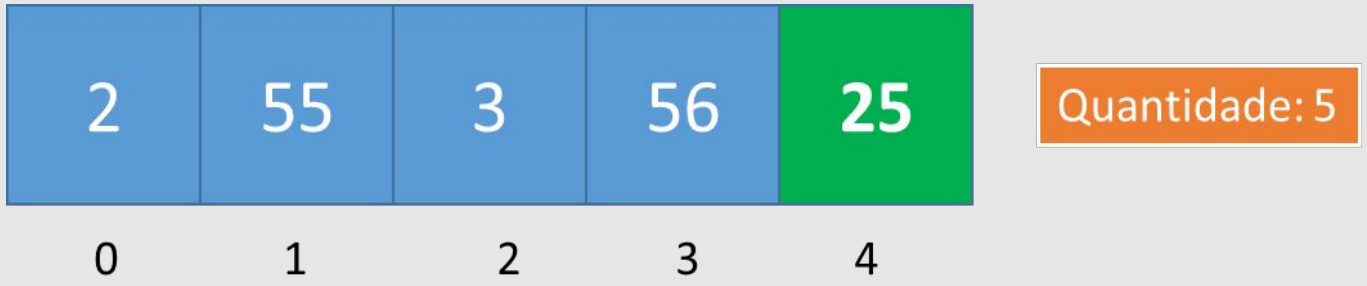
- No fim:



- E se quisermos inserir o elemento 25 no fim da Lista?

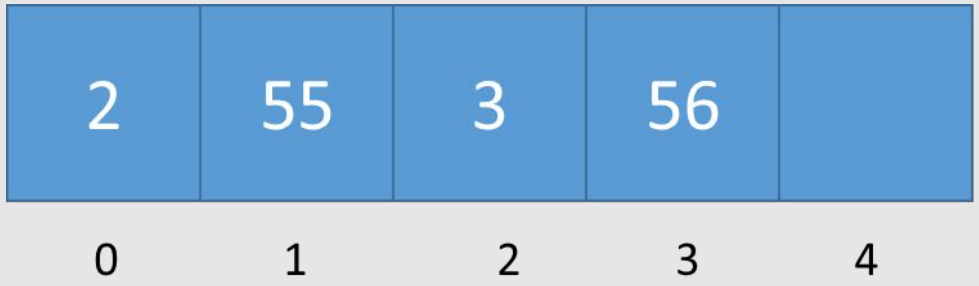
Inserção - Lista sequencial

- No fim:



Remoção - Lista sequencial

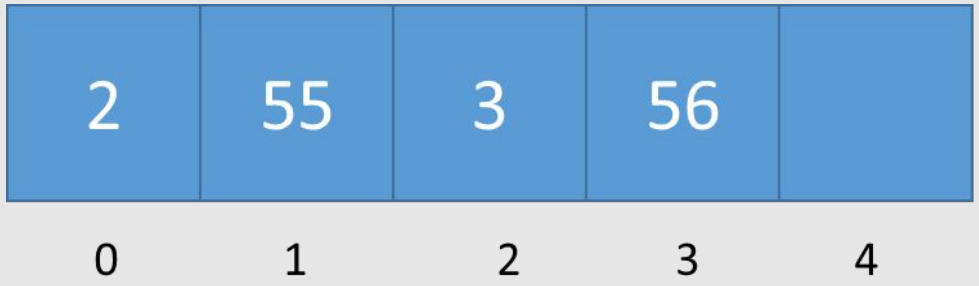
- No fim:



Quantidade: 4

Remoção - Lista sequencial

- No fim:

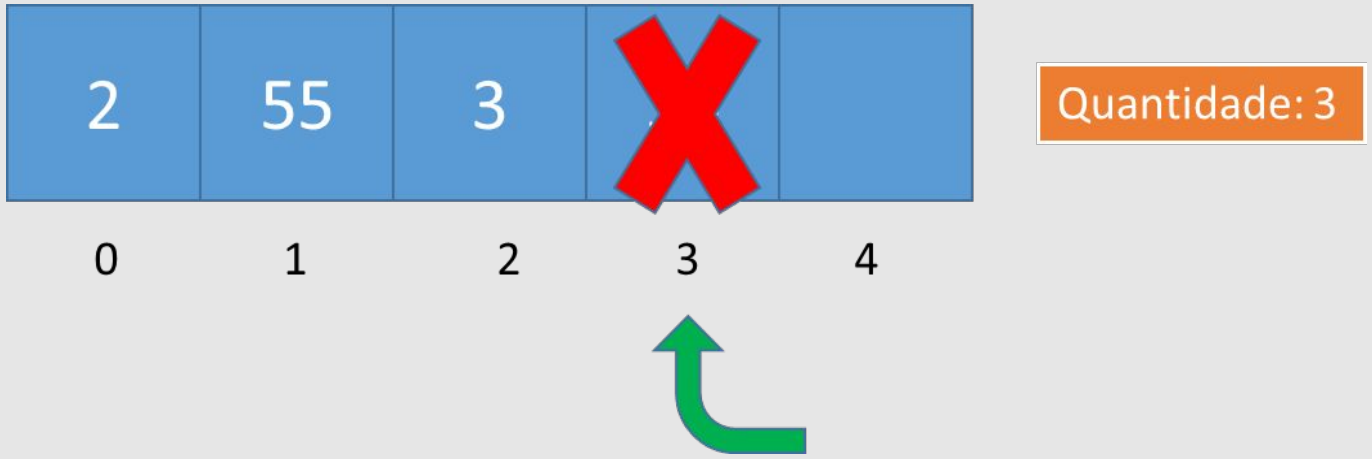


Quantidade: 4

- E se quisermos remover o elemento do final?

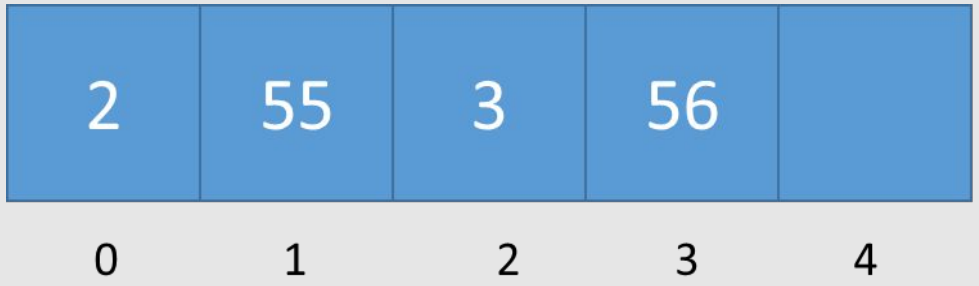
Remoção - Lista sequencial

- No fim:



- E se quisermos remover o elemento do final?

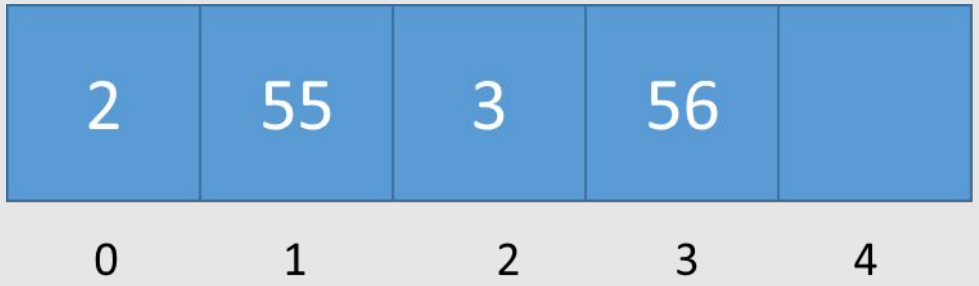
Busca - Lista sequencial



- Pode ser:
  - Por elemento (chave)
  - Por posição (índice)

Busca - Lista sequencial

- Por elemento:

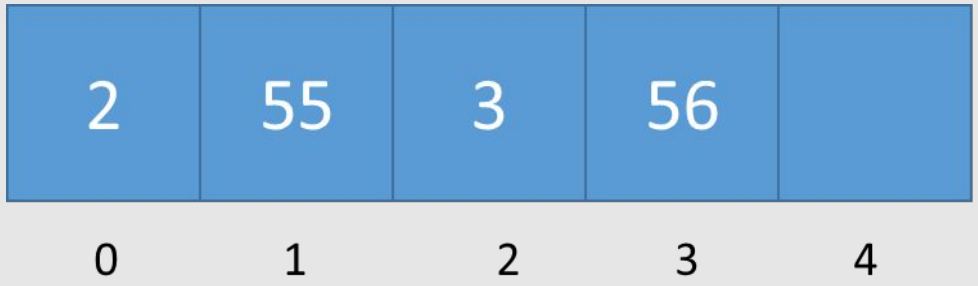


Quantidade: 4

- E se quisermos buscar o elemento 56?
  - Busca Sequencial,
  - Busca Binária (se estiver ordenada).

Busca - Lista sequencial

- Por posição:

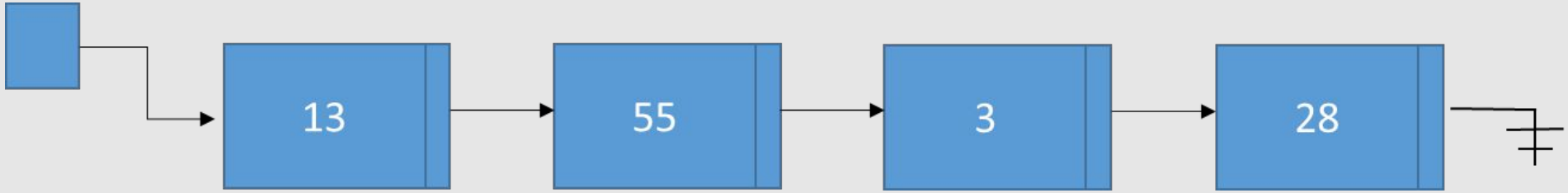


Quantidade: 4

- E se quisermos buscar o elemento da posição 2?
  - Acesso direto pelo índice.

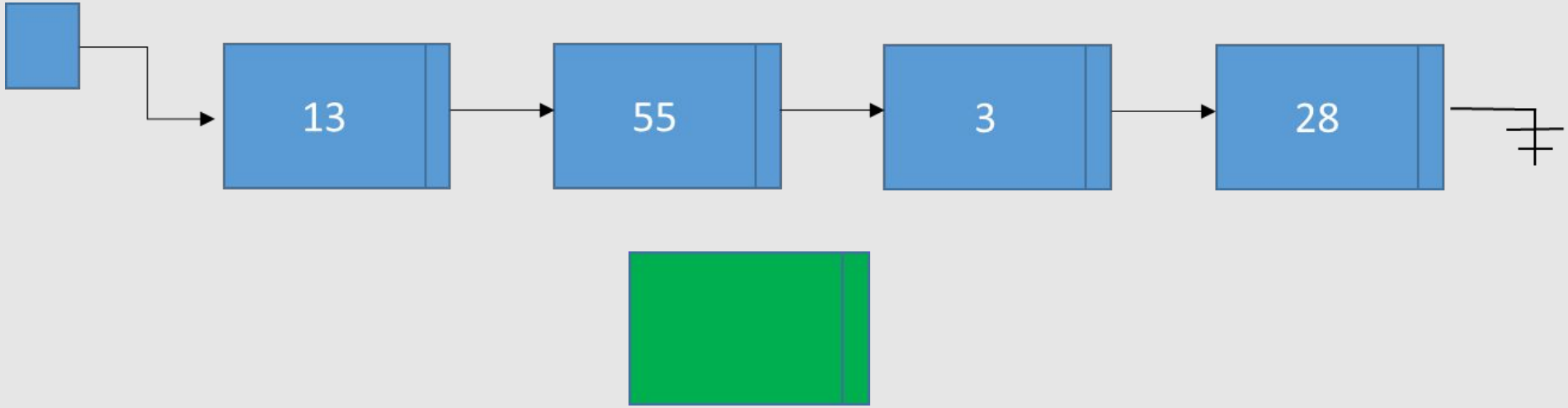
Inserção - Lista encadeada

- No final:



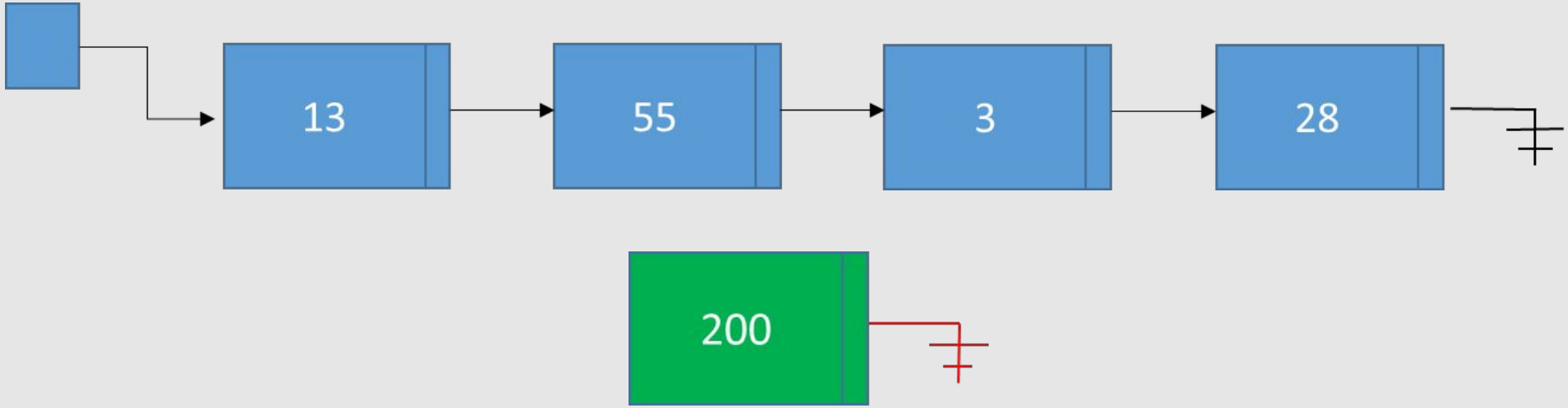
Inserção - Lista encadeada

- No final:



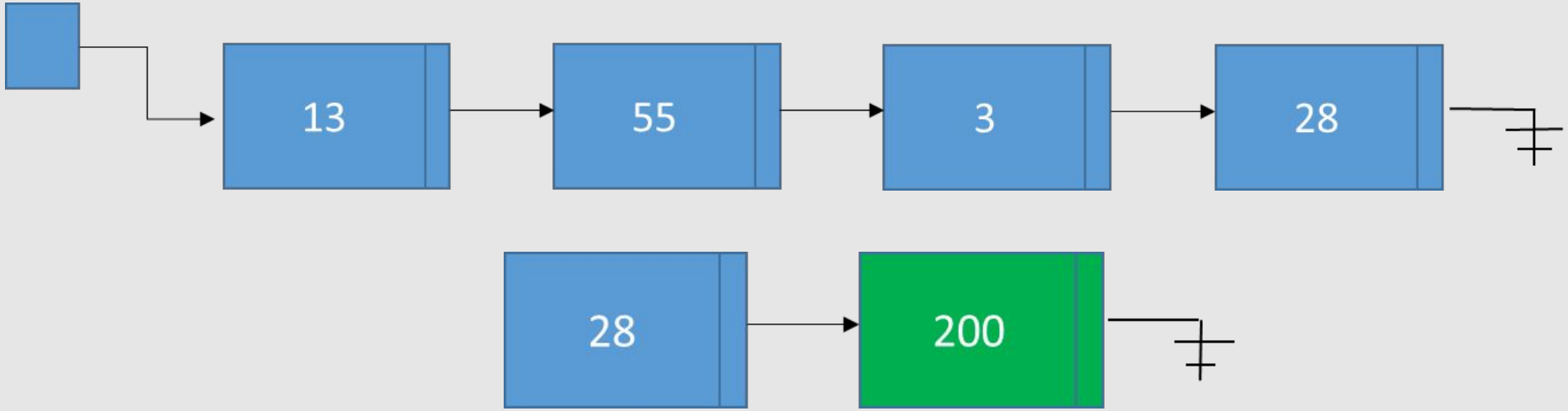
Inserção - Lista encadeada

- No final:



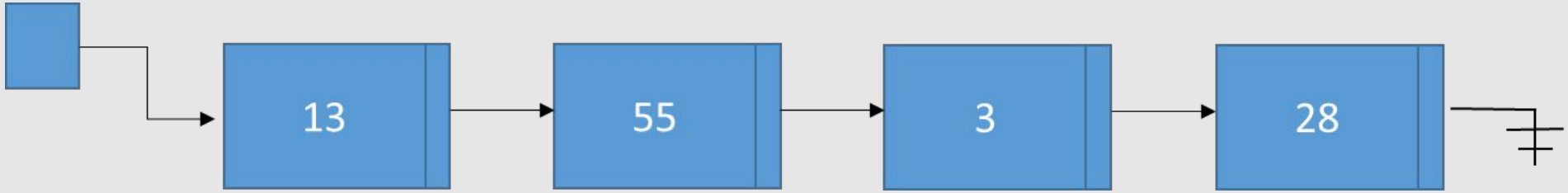
Inserção - Lista encadeada

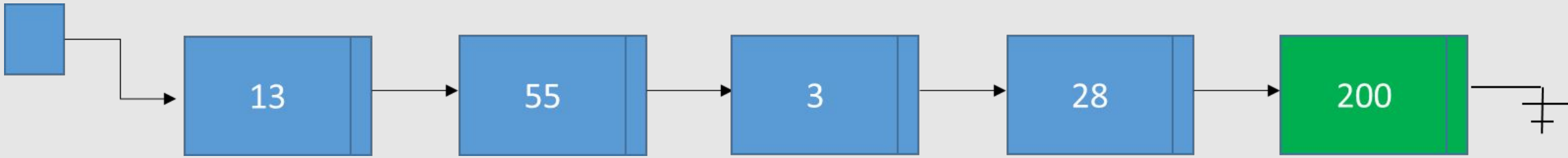
- No final:



Inserção - Lista encadeada

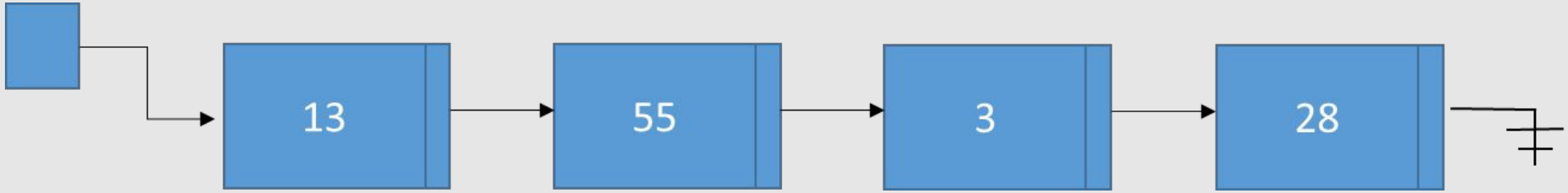
- No final:





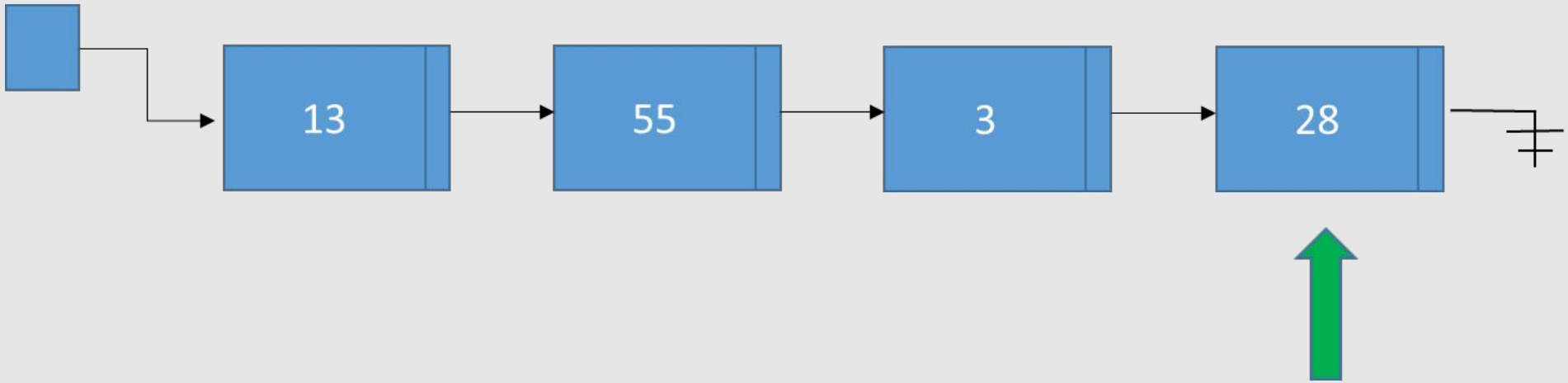
Remoção - Lista encadeada

- No final:



Remoção - Lista encadeada

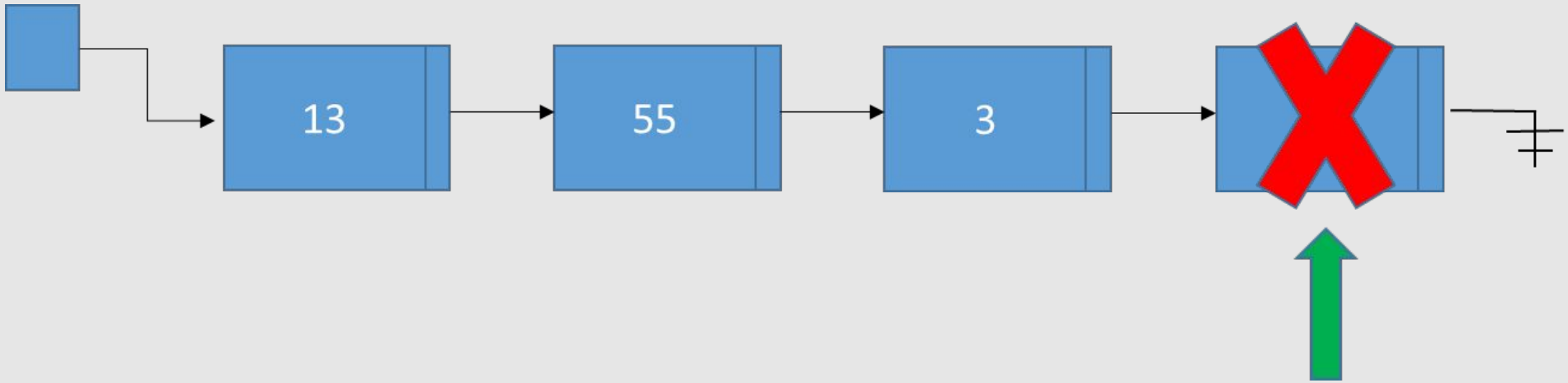
- No final:



- E se quisermos remover o elemento 28?

Remoção - Lista encadeada

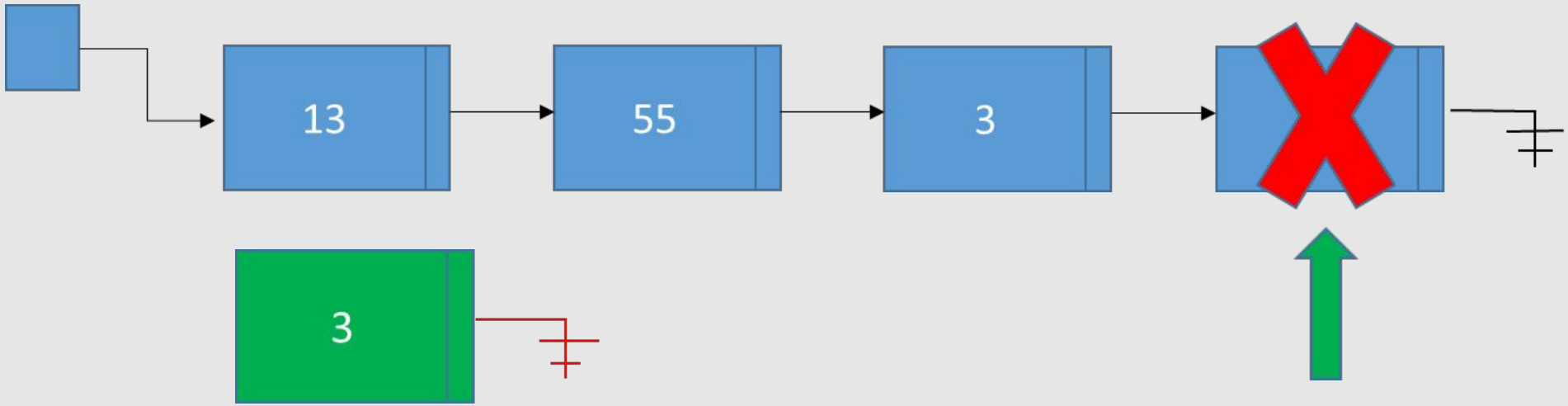
- No final:



- E se quisermos remover o elemento 28?

Remoção - Lista encadeada

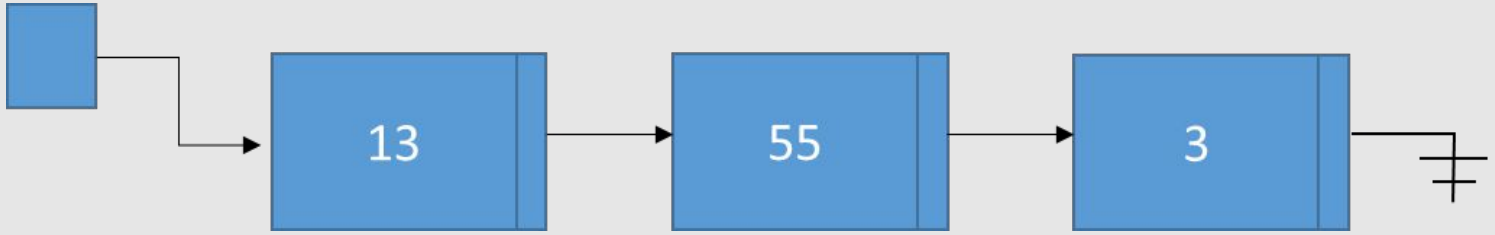
- No final:



- E se quisermos remover o elemento 28?

Remoção - Lista encadeada

- No final:



Busca - Lista encadeada

- Busca por Elemento:
  - Percorre a Lista comparando a chave de busca com cada elemento da estrutura, até o final.
- Busca por Posição:
  - Utiliza um contador para verificação da posição atual.

Implementação de Lista sequencial

```
# Classe que representa a Lista e seus elementos
class ListaSequencial:
    def __init__(self, capacidade):
        self.capacidade = capacidade
        self.dados = [None] * capacidade  # Array fixo
        self.tamanho = 0  # Quantidade atual de elementos
```

Implementação de Lista sequencial

```
def inserir(self, elemento):
    if self.tamanho < self.capacidade:
        self.dados[self.tamanho] = elemento
        self.tamanho += 1
    else:
        print("Lista está cheia!")
```

Implementação de Lista sequencial

```
def remover(self, indice):
    if 0 <= indice < self.tamanho:
        for i in range(indice, self.tamanho - 1):
            self.dados[i] = self.dados[i + 1]
        self.dados[self.tamanho - 1] = None
        self.tamanho -= 1
    else:
        print("Índice inválido!")
```

Implementação de Lista sequencial

```
def buscar(self, elemento):
    for i in range(self.tamanho):
        if self.dados[i] == elemento:
            return i
    return -1
```

Implementação de Lista sequencial

```
def imprimir(self):
    for i in range(self.tamanho):
        print(self.dados[i], end=" | ")
    print()
```

Implementação de Lista sequencial

```
# Exemplo de uso
lista = ListaSequencial(5)
lista.inserir(10)
lista.inserir(20)
lista.inserir(30)
lista.imprimir()
print("Índice do 20:", lista.buscar(20))
lista.remover(1)
lista.imprimir()
```

Implementação de Lista encadeada

```
# Classe que representa os nós (elementos) da Lista
class No:
    def __init__(self, dados):
        self.dados = dados
        self.proximo = None
```

Implementação de Lista encadeada

```
# Classe que representa a Lista
class ListaEncadeada:
    def __init__(self):
        self.cabeca = None
```

Implementação de Lista encadeada

```
# Adiciona um novo nó no início da lista
 def inserir_inicio(self, dado):
     novo_no = No(dado)
     novo_no.proximo = self.cabeca
     self.cabeca = novo_no
```

Implementação de Lista encadeada

```
# Remove um novo nó no início da lista
  def remover_inicio(self):
      if self.cabeca:
          self.cabeca = self.cabeca.proximo
```

Implementação de Lista encadeada

```
# Exibe os elementos da lista
  def imprimir(self):
      atual = self.cabeca
      while atual:
          print(atual.dados, end=" -> ")
          atual = atual.proximo
      print("None")
```

Implementação de Lista encadeada

```
# Exemplo de uso
lista = ListaEncadeada()
lista.inserir_inicio(10)
lista.inserir_inicio(20)
lista.inserir_inicio(30)
lista.imprimir()
lista.remover_inicio()
lista.imprimir()
```

Recapitulação

- TADs definem o comportamento e as operações de uma estrutura de dados, independentemente de sua implementação.
- Lista Linear: armazena elementos de forma contígua.
  - Útil para situações com tamanho fixo e acesso direto.
- Lista Encadeada: cada elemento aponta para o próximo.
  - Permite inserções e remoções eficientes, sem necessidade de deslocar elementos.

Referências

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

PIXABAY. **Pixabay**. [S.d.]. Disponível em: https://link.ufms.br/1niQd. Acesso em: 2 jun. 2025.

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

WIKIMEDIA FOUNDATION. **Wikimedia Commons**. [S.d.]. Disponível em: https://link.ufms.br/ySkva. Acesso em: 2 jun. 2025.

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