Estrutura de Dados

Aula 07

Prof. Luiz Antonio Schalata Pacheco, Dr. Eng.

Instituto Federal de Santa Catarina
Câmpus Garopaba
Curso Superior de Tecnologia em Sistemas para Internet

schalata@ifsc.edu.br

04/05/2023

Listas Encadeadas Simples

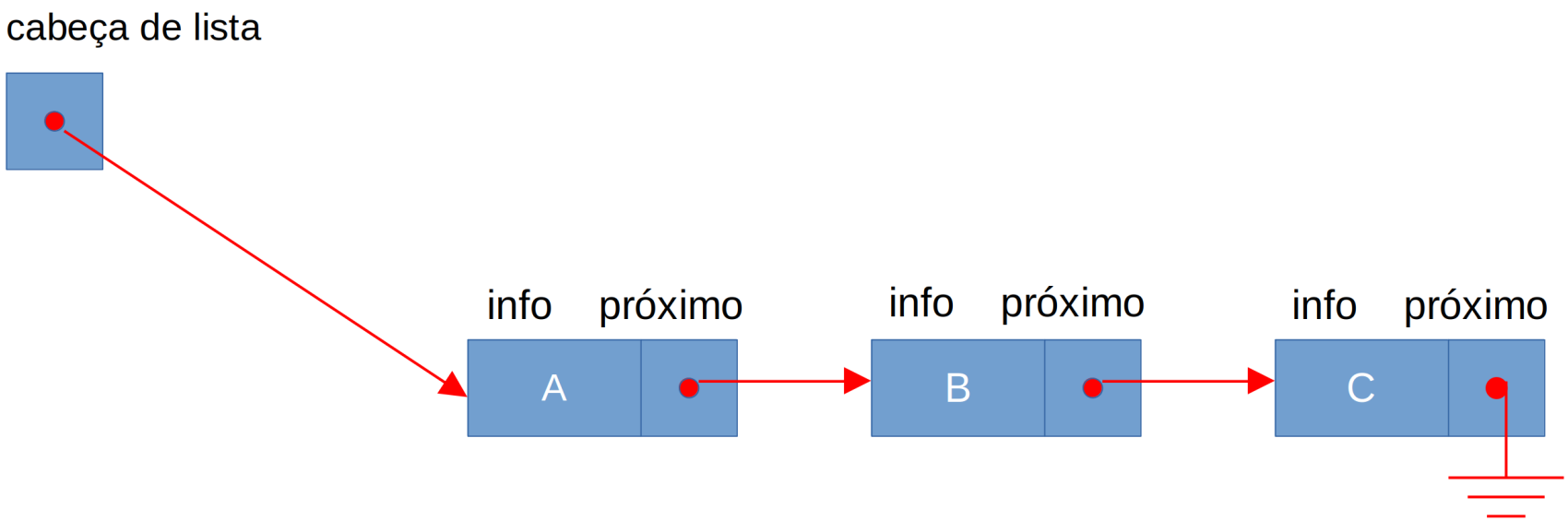
Motivação

- Surgiram para melhorar algumas limitações dos vetores:
  - Em um vetor não ordenado, a busca é lenta (usa pesquisa linear)
  - Em um vetor ordenado, a inserção é lenta (pesquisa e necessidade de remanejamento)
  - A remoção é lenta em ambos (pesquisa e remanejamento)
  - O tamanho do vetor não é alterado após a criação (vetores clássicos)
  - Mesmo vazios ocupam espaço de memória

Conceito

- São listas onde cada elemento está armazenado em uma Classe chamada *elemento de lista*
- Usamos o conceito de nó. Cada item de dado é incorporado em um nó
- Cada nó possui uma referência para o próximo nó da lista (essa referência indica um endereço de memória)
- Portanto, cada *elemento de lista* referencia o próximo
- Permite alocação dinâmica dos elementos
- A referência ao primeiro elemento é chamada de *cabeça de lista*
- A cauda da lista é um ponteiro que aponta para nulo (*NULL*)

Lista Encadeada



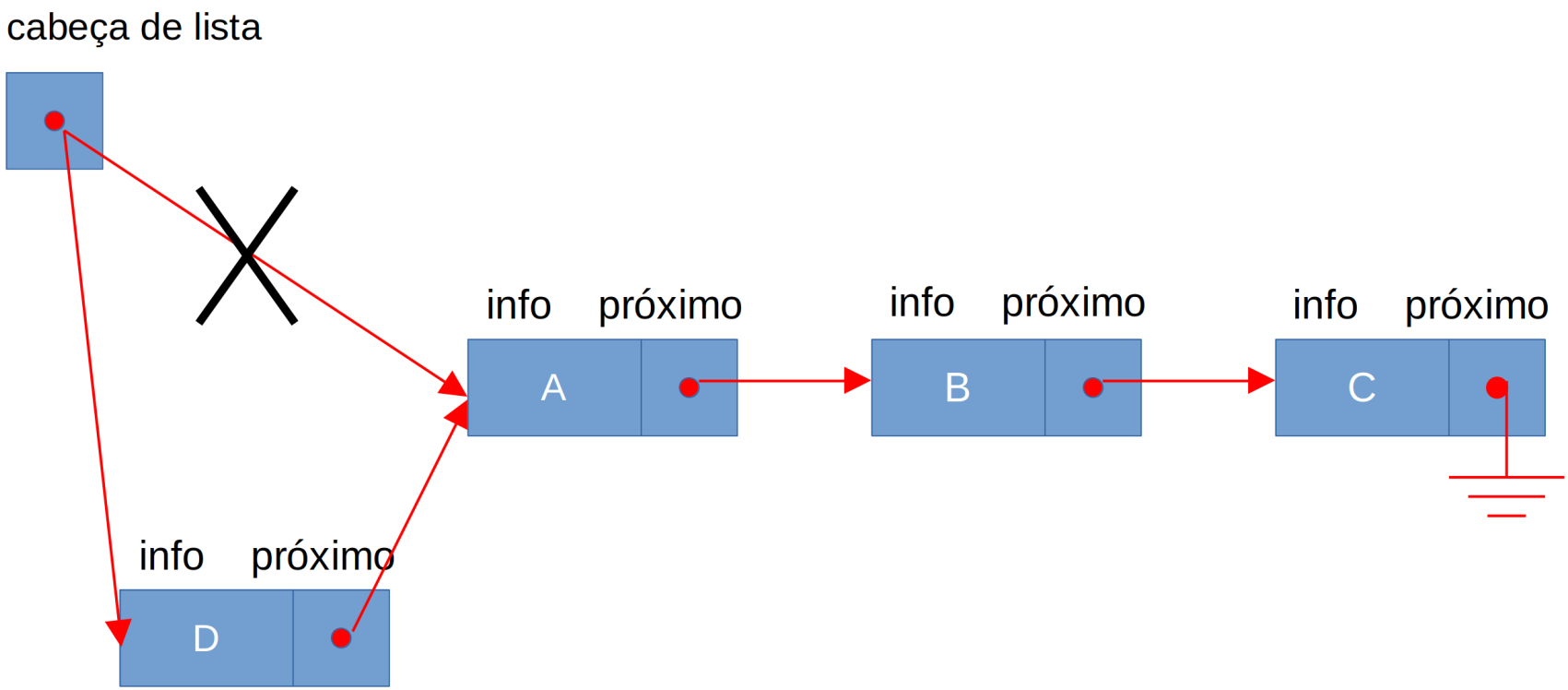
Listas Encadeadas x Vetores

- Vetor (posição)
  - Cada item ocupa uma posição
  - Cada posição pode ser acessada através de um índice
- Lista Encadeada (relacionamento)
  - Para encontrar um elemento é seguir a sequência de elementos
  - Existe referência para a cabeça de lista
  - Um item de dados não pode ser acessado diretamente, ou seja, o relacionamento entre eles deve ser utilizado
  - Inicia no primeiro item, vai para o segundo, então o terceiro, até encontrar o item pesquisado, portanto faz pesquisa linear

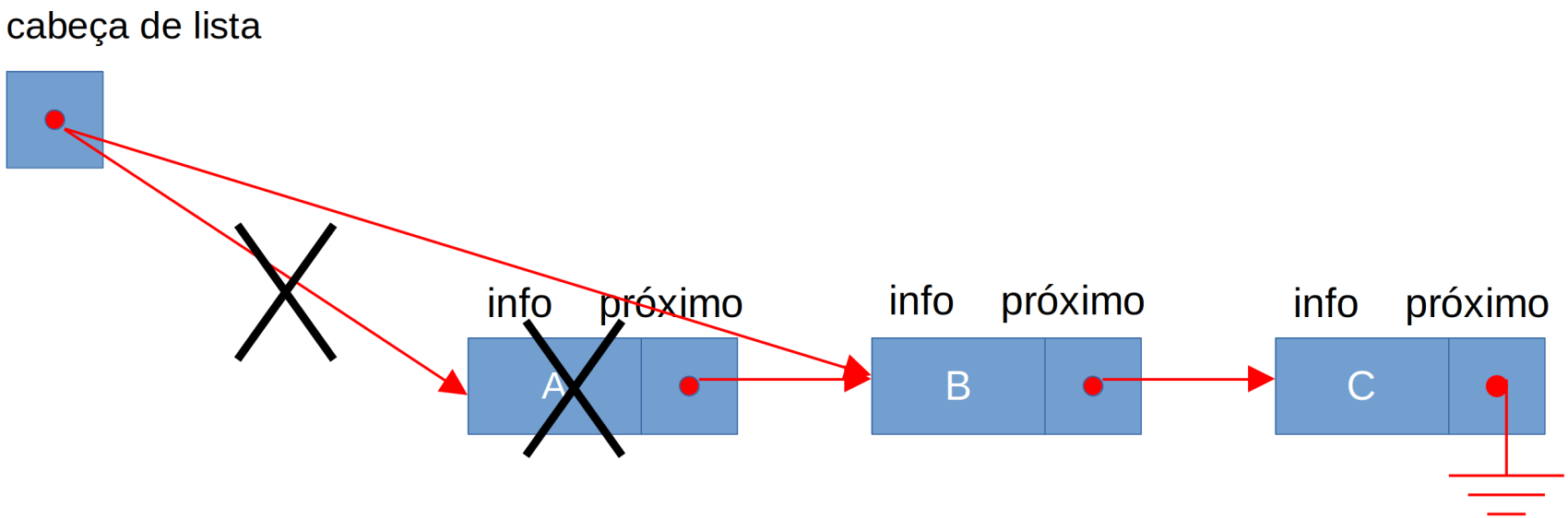
Operações

- Inserir no início
- Excluir do início
- Mostrar lista
- Pesquisar
- Excluir da posição

Lista Encadeada - Inserção no início



Lista Encadeada - Exclusão do início



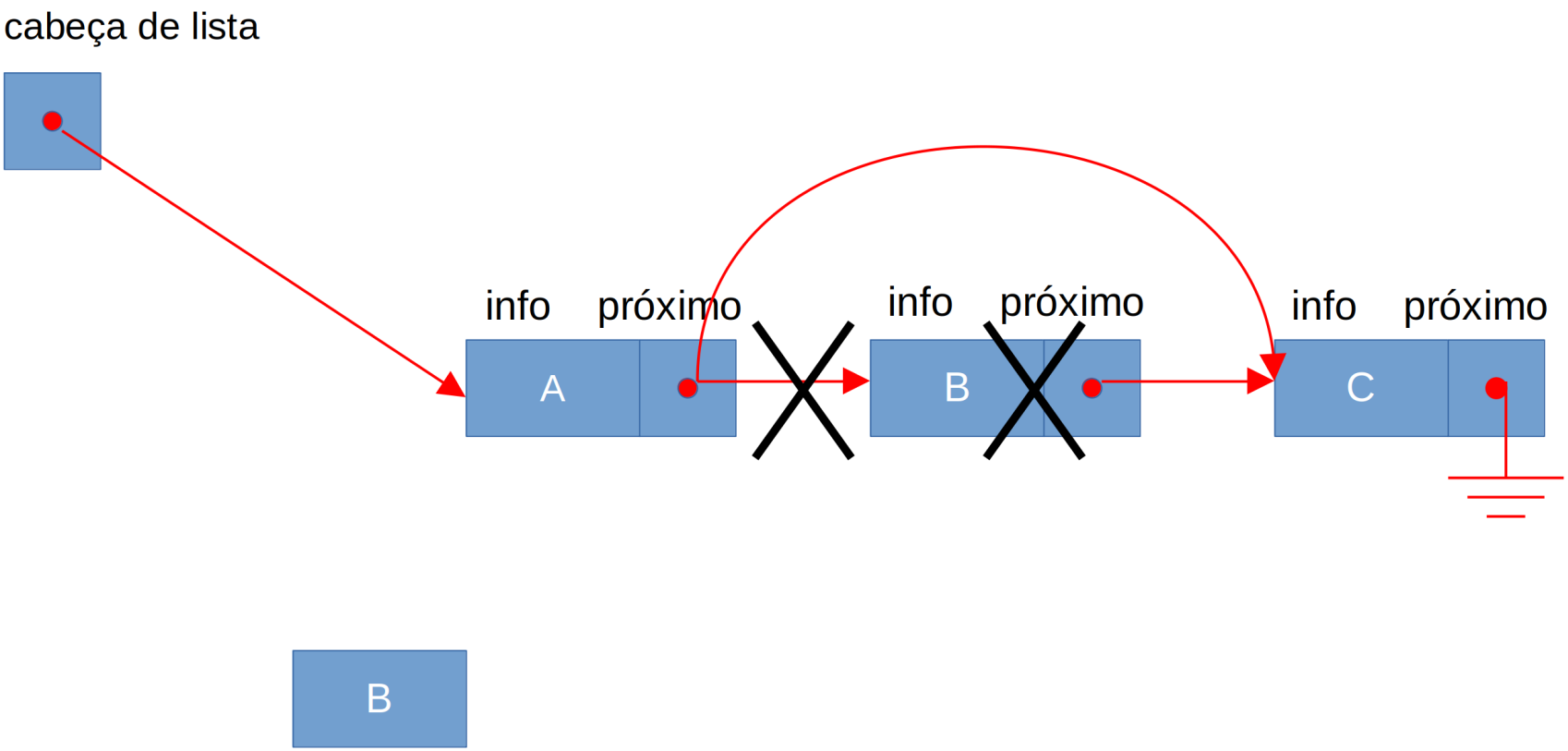
Lista Encadeada - Exclusão do início

- O elemento de lista não é, de fato, apagado num primeiro momento
- O sistema de gerenciamento de memória da linguagem vai identificar que não existe nenhuma ligação para o objeto
- O coletor de lixos vai verificar se existem variáveis que não estão sendo usadas e se encarrega de desalocar esse espaços
- Em linguagens como o C é preciso desalocar a memória, mas Java e Python, por exemplo, conseguem fazer isso automaticamente

Lista Encadeada - Mostrar lista e pesquisar

- Para exibir a lista, deverá ser iniciado no primeiro elemento, seguindo a sequência de referências dos nós
- O final da lista aponta para *null* ao invés de outro nó
- Os nós são percorridos e é verificado se o valor do elemento é aquele que está sendo procurado (pesquisa linear)
- Se atingir o final da lista sem encontrar o nó desejado, finaliza sem encontrar o elemento

Lista Encadeada - Excluir da posição



Implementação

- Implemente a classe Nó
  - Lembre-se que um nó tem um dado e um apontador para o próximo nó
  - A classe deve conter um método que mostre o valor (informação) do nó

Implementação da Classe Nó

```
1 class No:
2   def __init__(self, valor):
3     self.valor = valor
4     self.proximo = None
5
6   def mostra_no(self):
7     print(self.valor)
```


Implementação

- Implemente a classe Lista Encadeada com seus métodos:
  - Inserir no início
  - Mostrar lista
  - Pesquisar
  - Excluir do início
  - Excluir da posição

Lista Encadeada: Inserir no início

```
1  from No import No
2
3  class ListaEncadeada:
4    def __init__(self):
5      self.primeiro = None # Cabeca da lista
6
7    def insere_inicio(self, valor):
8      novo = No(valor) # Cria um novo no
9      novo.proximo = self.primeiro # O campo proximo do
       novo elemento deve apontar para o primeiro elemento
       da lista
10     self.primeiro = novo # A cabeca da lista aponta para
       o novo
```


Lista Encadeada: Mostrar

```
12   def mostrar(self):
13     if self.primeiro == None:
14       print('A lista esta vazia')
15       return None
16
17     atual = self.primeiro
18     while atual != None:
19       atual.mostra_no()
20       atual = atual.proximo # Proximo elemento da lista
```


Lista Encadeada: Pesquisar

```
22  def pesquisa(self, valor):
23    if self.primeiro == None:
24      print('A lista esta vazia')
25      return None
26
27    atual = self.primeiro
28    while atual.valor != valor:
29      if atual.proximo == None:
30        return None
31      else:
32        atual = atual.proximo
33    return atual
```


Lista Encadeada: Excluir do início

```
35    def excluir_inicio(self):
36        if self.primeiro == None:
37            print('A lista esta vazia')
38            return None
39
40        temp = self.primeiro
41        self.primeiro = self.primeiro.proximo
42        return temp
```


Lista Encadeada: Excluir da posição

```
44   def excluir_posicao(self, valor):
45     if self.primeiro == None:
46       print('A lista esta vazia')
47       return None
48
49     atual = self.primeiro
50     anterior = self.primeiro
51     while atual.valor != valor:
52       if atual.proximo == None:
53         return None
54       else:
55         anterior = atual
56         atual = atual.proximo
```


Lista Encadeada: Excluir da posição (continuação)

```
58      if atual == self.primeiro:
59        self.primeiro = self.primeiro.proximo
60      else:
61        anterior.proximo = atual.proximo
62
63      return atual
```


Exercício

- Crie uma nova classe chamada PilhaListaEncadeada com os seguintes métodos:
  - empilhar
  - desempilhar
  - verificar se a pilha está vazia
  - mostrar topo

Pilha com Lista Encadeada

```
1 class No:
2   def __init__(self, valor):
3     self.valor = valor
4     self.proximo = None
5
6   def mostra_no(self):
7     print(self.valor)
```


Pilha com Lista Encadeada

```
9  class ListaEncadeada:
10
11   def __init__(self):
12     self.primeiro = None
13
14   def lista_vazia(self):
15     return self.primeiro == None
16
17   def insere_inicio(self, valor):
18     novo = No(valor)
19     novo.proximo = self.primeiro
20     self.primeiro = novo
```


Pilha com Lista Encadeada

```
22    def excluir_inicio(self):
23        if self.lista_vazia():
24            print('A lista esta vazia')
25            return None
26
27        temp = self.primeiro
28        self.primeiro = self.primeiro.proximo
29        return temp
```


Pilha com Lista Encadeada

```
31 class PilhaListaEncadeada:
32   def __init__(self):
33     self.lista = ListaEncadeada()
34
35   def empilhar(self, valor):
36     self.lista.insere_inicio(valor)
37
38   def desempilhar(self):
39     return self.lista.excluir_inicio()
40
41   def pilha_vazia(self):
42     return self.lista.lista_vazia()
43
44   def ver_topo(self):
45     if self.lista.primeiro == None:
46       return -1
47     return self.lista.primeiro.valor
```