

Estrutura de dados

Listas encadeadas circulares

Prof^a. Dr^a. Alana Moraes

Recapitulando - Estrutura de dados

Dados simples:

- padrão:
 - inteiro (int);
 - real (float);
 - caracter (str);
 - lógico (boolean).

Dados estruturados:

- Estáticos:
 - arrays;
 - registros;
 - arquivos;
 - conjuntos;
 - cadeias.
- Dinâmicos:
 - filas;
 - pilhas;
 - listas encadeadas;
 - árvores;
 - grafos.

Recapitulando - Estrutura de dados

Dados simples:

- padrão:
 - inteiro (int);
 - real (float);
 - caracter (str);
 - lógico (boolean).

Dados estruturados:

- Estáticos:

- arrays;
- registros;
- arquivos;
- conjuntos;
- cadeias.



- Dinâmicos:

- filas;
- pilhas;
- listas encadeadas;
- árvores;
- grafos.



Recapitulando - Estrutura de dados

Dados simples:

- padrão:
 - inteiro (int);
 - real (float);
 - caracter (str);
 - lógico (boolean).

Dados estruturados:

- Estáticos:

- arrays;
- registros;
- arquivos;
- conjuntos;
- cadeias.



- Dinâmicos:

- filas;
- pilhas;
- listas encadeadas;
- árvores;
- grafos.



Precisaremos novamente da classe No

node.py

```
class No(object):  
    def __init__(self):  
        self.valor = None  
        self.proximo = None  
    def setValor(self, valorNovo):  
        self.valor = valorNovo  
    def setProximo(self, proximoNovo):  
        self.proximo = proximoNovo  
    def getValor(self):  
        return self.valor  
    def getProximo(self):  
        return self.proximo
```

Lista Encadeada

- Também conhecida como **lista ligada**;
- O próximo do último elemento é None;



valor

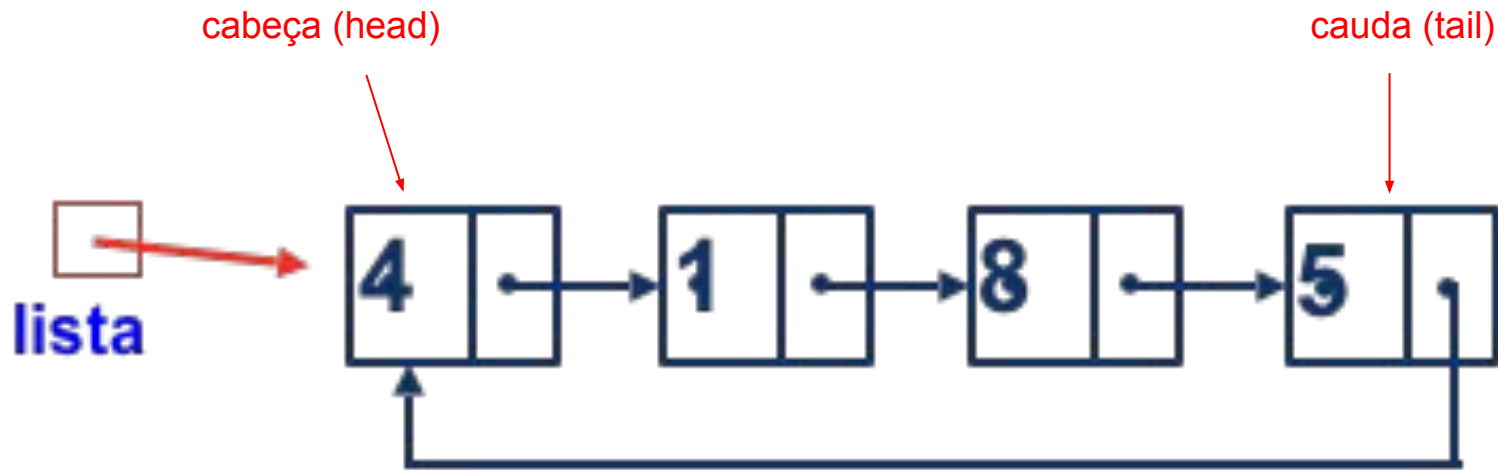
próximo

cabeça (head)

cauda (tail)

Lista encadeada Circular

- É um tipo de **lista encadeada**;
- O próximo do último elemento é a cabeça;



Operações com lista encadeada circular

- Criação;
- Inserção de elementos na lista encadeada circular;
- Mostrar elementos da lista encadeada circular;
- Remoção de elementos na lista encadeada circular;

Operações com lista encadeada circular

- **Criação;**
- Inserção de elementos na lista encadeada circular;
- Mostrar elementos da lista encadeada circular;
- Remoção de elementos na lista encadeada circular;

Criação de uma lista encadeada circular

listaencadeadacircular.py

```
import node

class ListaEncadeadaCircular(object):

    def __init__(self):
        self.cabeca = None
        self.cauda = None
        self.tam = 0
```

main.py

```
from listaencadeadacircular import ListaEncadeadaCircular

def main():
    lista = ListaEncadeadaCircular()

main()
```

Exercício A

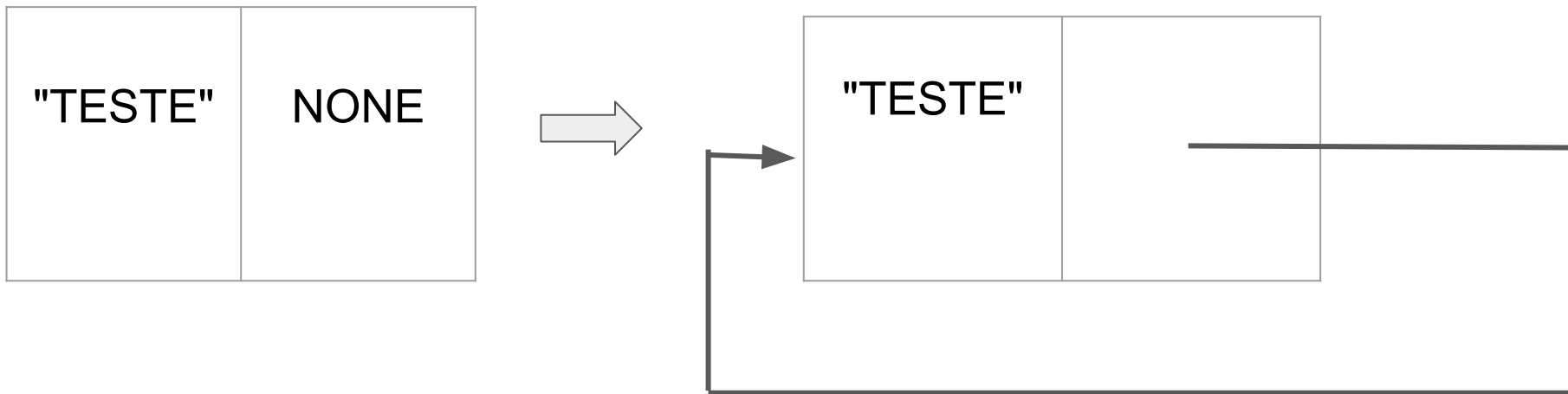
Implemente o método de criação de duas listas encadeadas circulares, utilizando uma classe chamada `ListaEncadeadaCircular`.

Operações com lista encadeada circular

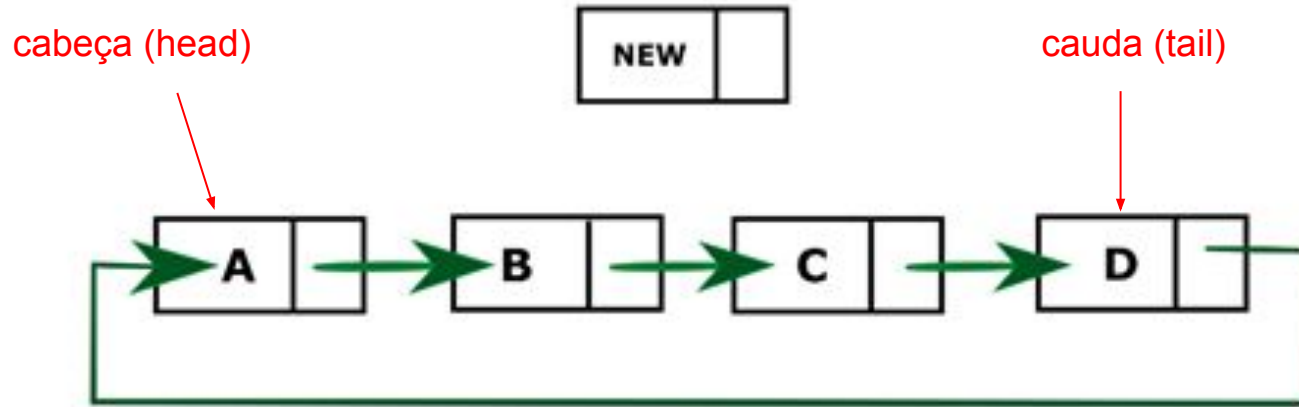
- Criação;
- **Inserção de elementos na lista encadeada circular;**
- Mostrar elementos da lista encadeada circular;
- Remoção de elementos na lista encadeada circular;

Lista encadeada Circular

- O primeiro elemento adicionado aponta para si próprio.

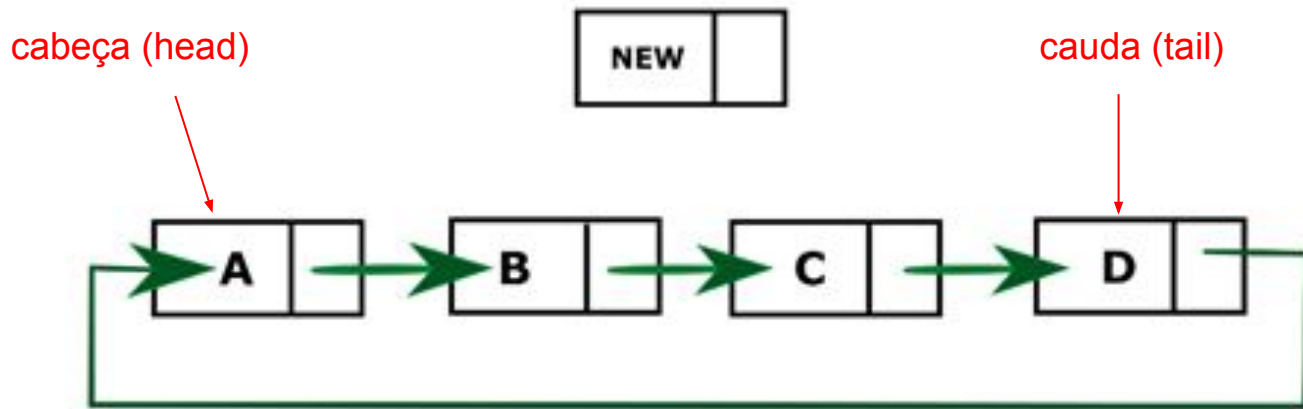


Como Inserir um novo elemento?



Como Inserir um novo elemento?

- Duas maneiras de inserir



Inserção de valores na cabeça (insert)

listaencadeadacircular.py

```
from node import No

class ListaEncadeadaCircular(object):
    ...
    def inserirNoInicio(self, valor):
        new_node = No()
        new_node.setValor(valor)
        new_node.setProximo(self.cabeca)
        self.cabeca = new_node

        if self.tam == 0:
            self.cauda = new_node
        self.cauda.setProximo(self.cabeca)
        self.tam += 1
```

main.py

```
from listaencadeadacircular
    import ListaEncadeadaCircular

def main():
    lista = ListaEncadeadaCircular()
    lista.inserirNoInicio(10)

main()
```


Inserção de valores na cauda (append)

listaencadeadacircular.py

```
import node

class ListaEncadeadaCircular(object):
    ...
    def inserirNoFinal(self, valor):
        new_node = node.No()
        new_node.setValor(valor)
        if self.tam == 0:
            self.cabeca = new_node
        else:
            self.cauda.setProximo(new_node)
        self.cauda = new_node
        self.cauda.setProximo(self.cabeca)
        self.tam += 1
```

main.py

```
from listaencadeadacircular
    import ListaEncadeadaCircular

def main():
    lista = ListaEncadeadaCircular()
    lista.inserirNoInicio(10)
    lista.inserirNoFinal(12)

main()
```

Operações com lista encadeada

- Criação;
- Inserção de elementos na lista encadeada circular;
- **Mostrar elementos da lista encadeada circular;**
- Remoção de elementos na lista encadeada circular;

Mostrar elementos da fila circular

listaencadeadacircular.py

```
import node

class ListaEncadeadaCircular(object):
    ...
    def __iter__(self):
        node = self.cabeca
        stop = 1
        while (node!=None and stop==1):
            yield node
            if node == self.cauda:
                stop = 0
            node = node.getProximo()
```

Mostrar elementos da fila circular

listaencadeadacircular.py

```
import node

class ListaEncadeadaCircular(object):
    ...
    def mostrar(self):
        items = ''
        for item in self.__iter__():
            items += str(item.getValor()) + ' '
            #print(item.getValor())
        return items
```

main.py

```
from listaencadeadacircular
    import ListaEncadeadaCircular

def main():
    lista = ListaEncadeadaCircular()
    lista.inserirNoInicio(10)
    lista.inserirNo(12)

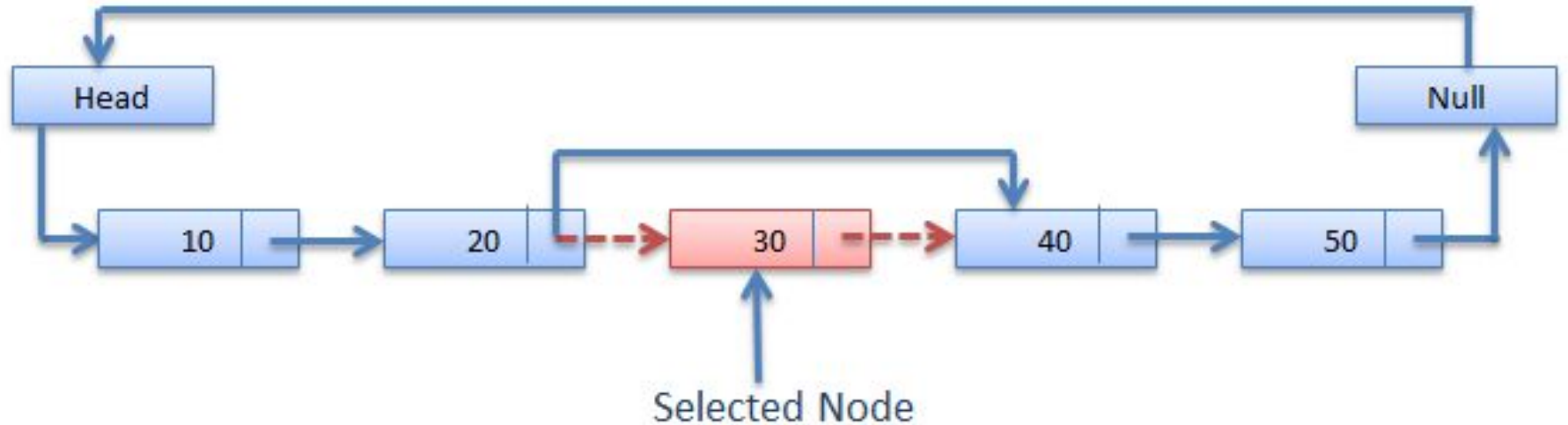
    print(lista.mostrar())

main()
```

Operações com lista encadeada

- Criação;
- Inserção de elementos na lista encadeada circular;
- Mostrar elementos da lista encadeada circular;
- **Remoção de elementos na lista encadeada circular;**

Remoção de elementos

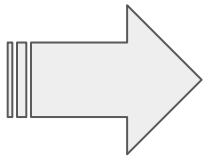


Remoção

listaencadeadacircular.py

```
import node
```

```
class ListaEncadeadaCircular(object):  
...  
    def remove(self, valor):  
        ...
```



```
def remove(self, valor):  
    prev_node = None  
    found = False  
    for curr_node in self.__iter__():  
        if valor == curr_node.getValor():  
            found = True  
            if prev_node:  
                prev_node.setProximo(curr_node.getProximo())  
                if curr_node == self.cauda:  
                    self.cauda = prev_node  
            else:  
                if self.tam == 1:  
                    self.__init__()  
                    return None  
                else:  
                    self.cabeca = curr_node.getProximo()  
                    self.cauda.setProximo(self.cabeca)  
  
            self.tam = self.tam - 1  
            prev_node = curr_node  
    if not found:  
        print("Valor "+str(valor)+" não encontrado na lista  
encadeada circular.")
```

Mostrar elementos da fila circular

listaencadeadacircular.py

```
import node

class ListaEncadeadaCircular(object):
    ...
    def remover(self, valor):
        ...
```

main.py

```
from listaencadeadacircular
    import ListaEncadeadaCircular

def main():
    lista = ListaEncadeadaCircular()
    lista.inserirNoInicio(10)
    lista.inserirNo(12)
    print(listaC.mostrar())

    lista.remover(10)
    lista.remover(12)
    print(listaC.mostrar())

main()
```


Exercício A

Implemente uma função que identifica se um elemento existe ou não na lista encadeada circular.

Exercício B

Insira elementos de forma ordenada em uma lista encadeada circular.

Dica: Você precisa criar uma nova função na classe `ListaEncadeadaCircular` chamada `inserirOrdenado(self, valor)`

Dúvidas?



alanamm.prof@gmail.com