



# Estrutura de Dados

Prof. Dr. Bruno Aguilar da Cunha  
[bruno.cunha@facens.br](mailto:bruno.cunha@facens.br)




01

# LISTAS ENCADEADAS

# LISTAS ENCADEADAS



Arranjos (Arrays) são **sequências de elementos de um mesmo tipo**, armazenados em posições contíguas de memória, e que possuem tamanho fixo, sendo úteis nos mais variados cenários. Entretanto, há situações em que desejamos maior flexibilidade, e as propriedades de arranjos não se adequam às nossas necessidades. Por exemplo, arranjos não são a estrutura de dados ideal se desejamos alguma das características a seguir:




# LISTAS ENCADEADAS



**Capacidade de inserir elementos no meio da estrutura de dados.** Se usarmos um arranjo e precisarmos inserir um elemento no meio da sequência, precisamos mover todos os elementos do arranjo.

**Capacidade de inserir um número arbitrário de elementos.** Em outras palavras, deseja-se que a estrutura de dados possua um tamanho dinâmico, sem valor máximo pré-definido.



# LISTAS ENCADEADAS

Listas encadeadas são a forma mais simples de estrutura de dados dinâmica. Em uma lista, os elementos não são armazenados contiguamente, como acontece em arranjos. Nelas, cada elemento é armazenado separadamente, e possui valor (o dado que desejamos acessar) e um ponteiro para o próximo elemento.

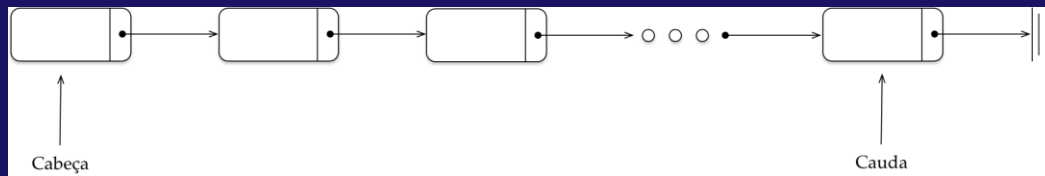
Um ponteiro (ou apontador) é uma variável que armazena o endereço de outra variável (eles referenciam outra variável).

# LISTAS ENCADEADAS

Listas encadeadas são a forma mais simples de estrutura de dados dinâmica. Em uma lista, os elementos não são armazenados contiguamente, como acontece em arranjos. Nelas, cada elemento é armazenado separadamente, e possui valor (o dado que desejamos acessar) e um ponteiro para o próximo elemento.

Um ponteiro (ou apontador) é uma variável que armazena o endereço de outra variável (eles referenciam outra variável).

# LISTAS ENCADEADAS



**Listas encadeadas são a forma mais simples de estrutura de dados dinâmica.** Em uma lista, os elementos não são armazenados contiguamente, como acontece em arranjos. Nelas, **cada elemento é armazenado separadamente,** e possui valor (o dado que desejamos acessar) e um ponteiro para o próximo elemento.

**Um ponteiro (ou apontador)** é uma variável que armazena o endereço de outra variável (eles referenciam outra variável).

# LISTAS ENCADEADAS

No caso de listas encadeadas, um ponteiro armazena o endereço do próximo elemento da lista. Para percorrermos a lista, basta seguirmos os ponteiros ao longo dela.

A flexibilidade de listas encadeadas não vem de graça. Listas encadeadas possuem duas desvantagens principais. Primeiro, precisamos de espaço adicional para armazenar os ponteiros. Segundo, se quisermos acessar um elemento em uma dada posição da lista, precisamos percorrer todos os elementos anteriores a ele na lista.



# LISTAS ENCADEADAS

A classe abaixo um nó de uma lista encadeada

```
public class IntNoSimples {  
    int valor;  
    IntNoSimples prox;  
    IntNoSimples(int ValorNo) {  
        valor = ValorNo;  
        prox = null;  
    }  
}
```

# LISTAS ENCADEADAS

Para facilitar a implementação de alguns métodos importantes, criaremos uma nova classe para armazenar um ponteiro para o início (primeiro) e fim (último) da lista. Essa classe será nossa lista encadeada propriamente dita.

```
public class ListaEncadeada {  
    IntNoSimples primeiro, ultimo;  
    int numero_nos=0;  
  
    ListaEncadeada() {  
        primeiro = ultimo = null;  
    }  
}
```

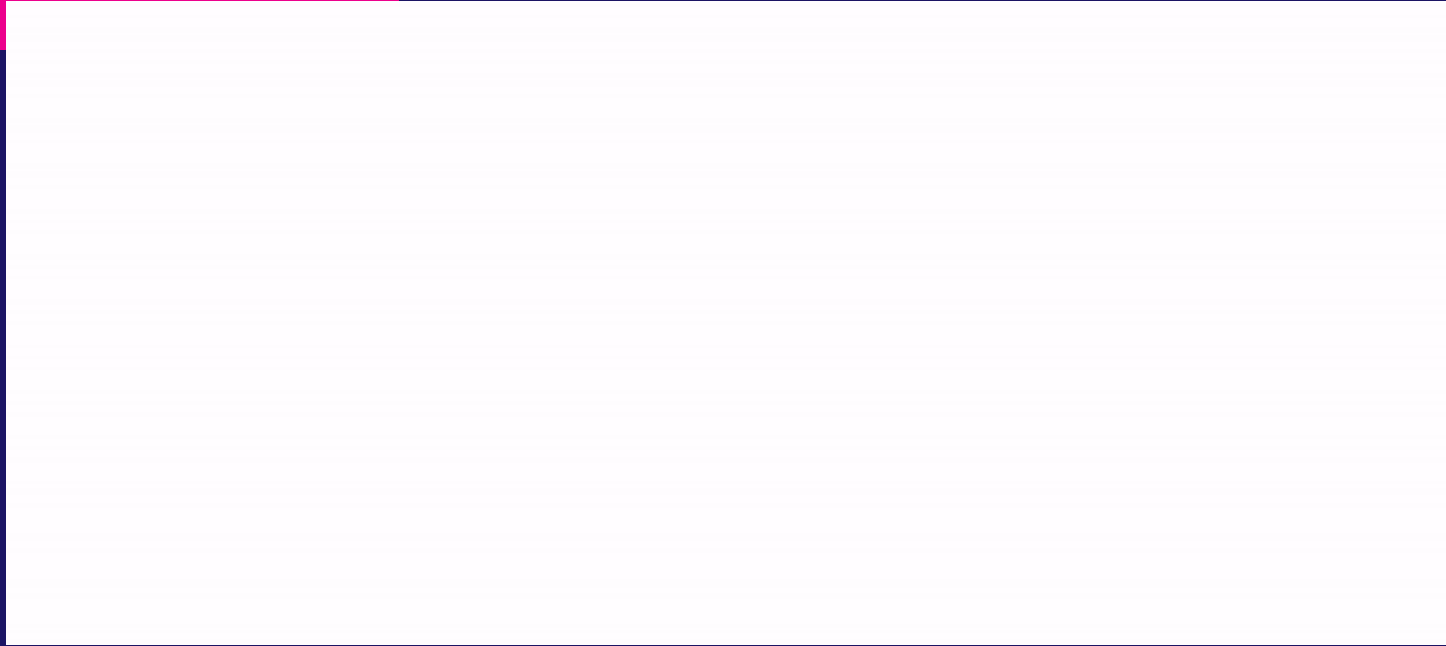
# INSERIR NA LISTA (INICIO)

A forma mais simples e mais rápida de se inserir um elemento em uma lista encadeada é inseri-lo no começo da lista. O código abaixo estende nossa classe Lista definida anteriormente para conter uma função `insere_no_inicio`.

```
void insereNo_inicio (IntNoSimples novoNo) {  
    novoNo.prox = primeiro;  
    if(primeiro == null && ultimo==null)  
    {  
        ultimo = novoNo;  
    }  
    primeiro = novoNo;  
    numero_nos++;  
}
```

# ILUSTRAÇÃO

## INSERE\_NO\_INICIO



# INSERIR NA LISTA (FIM)

```
void insereNo_fim (IntNoSimples novoNo) {  
    novoNo.prox = null;  
    if (primeiro == null)  
        primeiro = novoNo;  
    if (ultimo != null)  
        ultimo.prox = novoNo;  
    ultimo = novoNo;  
    numero_nos++;  
}
```

# INSERIR NA LISTA (POSIÇÃO)

```
void insereNo_posicao(IntNoSimples novoNo, int posicao){
    IntNoSimples temp_no = primeiro;
    int numero_nos = ContarNos();
    int pos_aux;
    if(posicao == 0)
    {
        novoNo.prox = primeiro;
        if(primeiro == ultimo)
        {
            ultimo = novoNo;
        }
        primeiro = novoNo;
    }
    else
    {
        if (posicao <= numero_nos)
        {
            pos_aux = 1;
            while(temp_no != null && posicao > pos_aux)
            {
                temp_no = temp_no.prox;
                pos_aux ++;
            }
            novoNo.prox = temp_no.prox;
            temp_no.prox = novoNo;
        }
        else
        {
            if(posicao > numero_nos)
            {
                ultimo.prox = novoNo;
                ultimo = novoNo;
            }
        }
    }
}
```

# BUSCA NA LISTA

```
IntNoSimples buscaNo (int buscaValor){  
    int i = 0;  
    IntNoSimples temp_no = primeiro;  
    while (temp_no != null)  
    {  
        if (temp_no.valor == buscaValor)  
        {  
            JOptionPane.showMessageDialog(null,  
                "No " + temp_no.valor + " posição " + i);  
            return temp_no;  
        }  
        i++;  
        temp_no = temp_no.prox;  
    }  
    return null;  
}
```

# EXCLUSÃO NA LISTA

```
void excluNo (int valor){
    IntNoSimples temp_no = primeiro;
    IntNoSimples anterior_no=null;
    while (temp_no != null && temp_no.valor != valor){
        anterior_no = temp_no;
        temp_no = temp_no.prox;
    }
    if (temp_no == primeiro){
        if (temp_no.prox !=null)
            primeiro = temp_no.prox;
        else
            primeiro = null;
    }
    else{
        anterior_no.prox =temp_no.prox;
    }
    if (ultimo == temp_no)
        ultimo = anterior_no;
}
```



# EXIBIÇÃO DA LISTA

```
void exhibeLista() {  
    IntNoSimples temp_no = primeiro;  
    int i = 0;  
    while (temp_no != null)  
    {  
        System.out.println(  
            "Valor" + temp_no.valor + " posição " + i);  
        temp_no = temp_no.prox;  
        i++;  
    }  
}
```

# EXEMPLO DE USO (PARTE 1)

```
public class ExemploLista {
    int i = 0;
    IntNoSimples temp_no;
    int valor;
    public static void main(String[] args) {
        ListaEncadeada l = new ListaEncadeada();
        int opcao = 1, valor, posicao;
        while (opcao != 7) {
            opcao = Integer.parseInt (JOptionPane.showInputDialog(null,
                "Escolha uma Opção \n" + "1-Inserir Nó no início \n" +
                "2-Inserir Nó no fim \n" + "3-Inserir Nó em uma posição\n" + "4-Localizar Nó \n" +
                "5-Excluir Nó \n" + "6-Exibir lista \n" + "7-Sair"));
            switch (opcao) {
                case 1 :
                    valor = Integer.parseInt (JOptionPane.showInputDialog(null, "Inserir um Nó no início da lista\n" +
                        "Digite um valor"));
                    l.inserirNo_inicio(new IntNoSimples(valor));
                    break;
                case 2 :
                    valor = Integer.parseInt (JOptionPane.
                        showInputDialog(null, "Inserir um Nó no final da lista \n" +
                        "Digite um valor"));
                    l.inserirNo_fim(new IntNoSimples(valor));
                    break;
```

# EXEMPLO DE USO (PARTE 2)

```
case 3 :
    valor = Integer.parseInt (JOptionPane.showInputDialog(null,
        "Inserir um Nó em uma posição \n" +
        "Digite um valor"));
    posicao = Integer.parseInt (JOptionPane.showInputDialog(null, "Digite a posição"));
    l.inserirNo_posicao(new IntNoSimples(valor),posicao);
    break;
case 4:
    valor = Integer.parseInt (JOptionPane.showInputDialog(null, "Localiza um valor \n" +
        "Digite um valor"));
    l.buscaNo(valor);
    break;
case 5:
    valor = Integer.parseInt (JOptionPane.showInputDialog(null,
        "Exclui um nó da lista \n" + "Digite um valor"));
    l.excluiNo(valor);
    break;
case 6:
    JOptionPane.showMessageDialog(null,"Exibe a lista");
    l.exibeLista();
    break;
default : JOptionPane.showMessageDialog(null,"Sair");
    }
}
```

# BUSCAS EM LISTAS



A pesquisa por um valor em uma lista é uma tarefa relativamente fácil, se comparada à tarefa de inserção de elementos.

O único detalhe a ser observado é o fato de que precisamos percorrer a lista desde o começo para pesquisar por qualquer um dos elementos.





02

## EXERCÍCIOS - LISTAS

# LISTA ENCADEADA - EXERCÍCIOS

1) Escreva um programa em JAVA que leia números inteiros e armazene em uma LISTA ENCADEADA. A entrada de dados deve ser interrompida quando o usuário informar o número zero ou esgotar a quantidade definida de elementos a serem armazenados na estrutura. Por último, imprima os elementos existentes na lista encadeada criada.

# LISTA ENCADEADA - EXERCÍCIOS

2) Escreva um programa em JAVA que leia números inteiros e só armazene aqueles que forem pares em uma LISTA ENCADEADA. A entrada de dados deve ser interrompida quando o usuário informar o número zero ou esgotar a quantidade definida de elementos a serem armazenados na estrutura. Por último, imprima os elementos existentes na lista encadeada criada.

# LISTA ENCADEADA - EXERCÍCIOS

3) Escreva um programa em JAVA que crie e preencha uma pilha e remova os elementos da pilha e coloque na lista encadeada e na sequência imprima os elementos existentes na lista.





03

LISTAS DUPLAMENTE  
ENCADEADAS

# LISTA DUPLAMENTE ENCADEADA

Quando se percorre uma lista de encadeamento simples é bastante difícil fazer o caminho inverso. Nas listas de encadeamento duplo esse problema não existe, pois cada nó possui uma referência para o próximo elemento da lista e outra para o anterior.

**A construção de uma lista duplamente encadeada é bastante similar à construção de listas simples, bastando acrescentar ao nó uma variável para fazer a referência ao elemento anterior, da mesma maneira que é feito com o próximo.**

# IMPLEMENTAÇÃO EM JAVA

A seguir será apresentada a implementação da lista duplamente encadeada em Java.

Observe que na classe **IntNoDuplo**, é criada a estrutura do nó, assim como fizemos para a lista simples, apenas acrescentamos o campo de referência **ant** que fará referência ao nó anterior.

# IMPLEMENTAÇÃO EM JAVA

A seguir será apresentada a implementação da lista duplamente encadeada em Java.

Observe que na classe **IntNoDuplo**, é criada a estrutura do nó, assim como fizemos para a lista simples, apenas acrescentamos o campo de referência **ant** que fará referência ao nó anterior.

# IntNoDuplo (criação do nó)

```
public class IntNoDuplo {  
    int valor;  
    IntNoDuplo prox;  
    IntNoDuplo ant;  
  
    IntNoDuplo (int ValorNo) {  
        valor = ValorNo;  
        prox = ant = null;  
    }  
}
```

# ListaDupla (implementação da lista)

```
public class ListaDupla {  
    IntNoDuplo primeiro, ultimo;  
    int numero_nos;  
  
    ListaDupla() {  
        primeiro = ultimo = null;  
        numero_nos = 0;  
    }  
}
```

# Inserção

```
void insereNo (IntNoDuplo novoNo) {  
    novoNo.prox = null;  
    novoNo.ant = ultimo;  
    if (primeiro == null)  
        primeiro = novoNo;  
    if (ultimo != null)  
        ultimo.prox = novoNo;  
    ultimo = novoNo;  
    numero_nos++;  
}
```

# Obter nó (de acordo com o índice)

```
IntNoDuplo pegarNo (int indice) {  
    IntNoDuplo temp_no = primeiro;  
    for (int i = 0; (i < indice) && (temp_no != null); i++)  
        temp_no = temp_no.prox;  
    return temp_no;  
}
```



## Incluir nó (informar índice)

```
void incluiNo (IntNoDuplo novoNo, int indice){
    IntNoDuplo temp_no = pegarNo (indice);
    novoNo.prox = temp_no;
    if (novoNo.prox != null){
        novoNo.ant = temp_no.ant;
        novoNo.prox.ant = novoNo;
    } else {
        novoNo.ant = ultimo;
        ultimo = novoNo;
    }
    if (indice == 0)
        primeiro = novoNo;
    else
        novoNo.ant.prox = novoNo;
    numero_nos++;
}
```

## Excluir nó (informar índice)

```
void excluNo (int indice){
    if (indice == 0){
        primeiro = primeiro.prox;
    }
    if (primeiro != null)
        primeiro.ant = null;
    }else{
        IntNoDuplo temp_no = pegarNo (indice);
        temp_no.ant.prox = temp_no.prox;
        if (temp_no != ultimo){
            temp_no.prox.ant = temp_no.ant;
        }else{
            ultimo = temp_no;
        }
    }
    numero_nos--;
}
```

## Testando a lista (Início)

```
public static void main(String[] args) {  
    ListaDupla Slist = new ListaDupla();  
    Slist.inserereNo (new IntNoDuplo (1));  
    Slist.inserereNo (new IntNoDuplo (3));  
    Slist.inserereNo (new IntNoDuplo (5));  
    Slist.inserereNo (new IntNoDuplo (7));  
    IntNoDuplo temp no = Slist.primeiro;
```

## Testando a lista (cont.)

```
IntNoDuplo temp_no = Slist.primeiro;
while (temp_no != null){
    System.out.println (temp_no.valor);
    temp_no = temp_no.prox;
}
Slist.incluiNo (new IntNoDuplo (2), 1);
System.out.println ("Apos incluir o no 2...");
```

## Testando a lista (cont.)

```
System.out.println ("Apos incluir o no 2...");  
temp_no = Slist.primeiro;  
while (temp_no != null) {  
    System.out.println (temp_no.valor);  
    temp_no = temp_no.prox;  
}
```

## Testando a lista (cont.)

```
Slist.excluiNo (2);  
System.out.println ("Apos excluir o no 3...");  
temp_no = Slist.primeiro;  
while (temp_no != null) {  
    System.out.println (temp_no.valor);  
    temp_no = temp_no.prox;  
}
```



04

EXERCÍCIOS

LISTAS DUPLAMENTE ENCADEADAS

# LISTA ENCADEADA - EXERCÍCIOS

1) Escreva um programa em JAVA que leia números inteiros e armazene em uma LISTA DUPLAMENTE ENCADEADA. A entrada de dados deve ser interrompida quando o usuário informar o número zero ou esgotar a quantidade definida de elementos a serem armazenados na estrutura. Por último, imprima os elementos existentes na lista duplamente encadeada criada.



# LISTA ENCADEADA - EXERCÍCIOS

2) Escreva um programa em JAVA que receba o nome e a idade de um conjunto de 10 pessoas em uma LISTA DUPLAMENTE ENCADEADA. Por último, imprima os elementos existentes na lista duplamente encadeada criada.

# LISTA ENCADEADA - EXERCÍCIOS

3) Escreva um programa em JAVA que crie e preencha uma fila e remova os elementos da fila e coloque na lista duplamente encadeada e na sequência imprima os elementos existentes na lista. Crie um método na classe ListaDupla que realize corretamente a exibição dos elementos da lista por meio de uma caixa de diálogo (JOptionPane).

# LISTA ENCADEADA - EXERCÍCIOS

4) Escreva um programa em JAVA que possibilite ao usuário incluir elementos em uma lista duplamente encadeada. Permita também ao usuário realizar uma busca por um valor inteiro na lista e informe se o valor está ou não na estrutura. Implemente e utilize um método que permita percorrer os elementos começando no último até chegar no primeiro (utilize o ant (anterior)) e mostre na tela os elementos visitados nessa ordem.

# Dúvidas?

Dúvidas fora do horário de aula:  
[bruno.cunha@facens.br](mailto:bruno.cunha@facens.br)