



# 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.

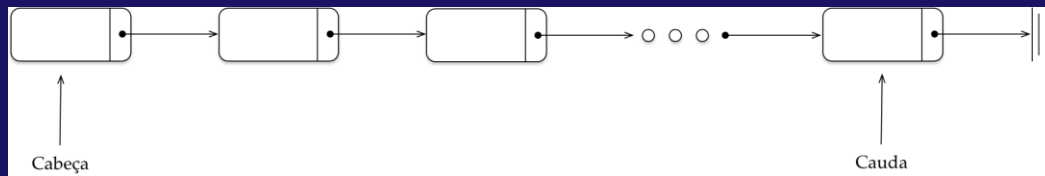
Um ponteiro (ou apontador) é uma variável que armazena o endereço de outra variável (eles referenciam outra variável).

# 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;  
    }  
}
```

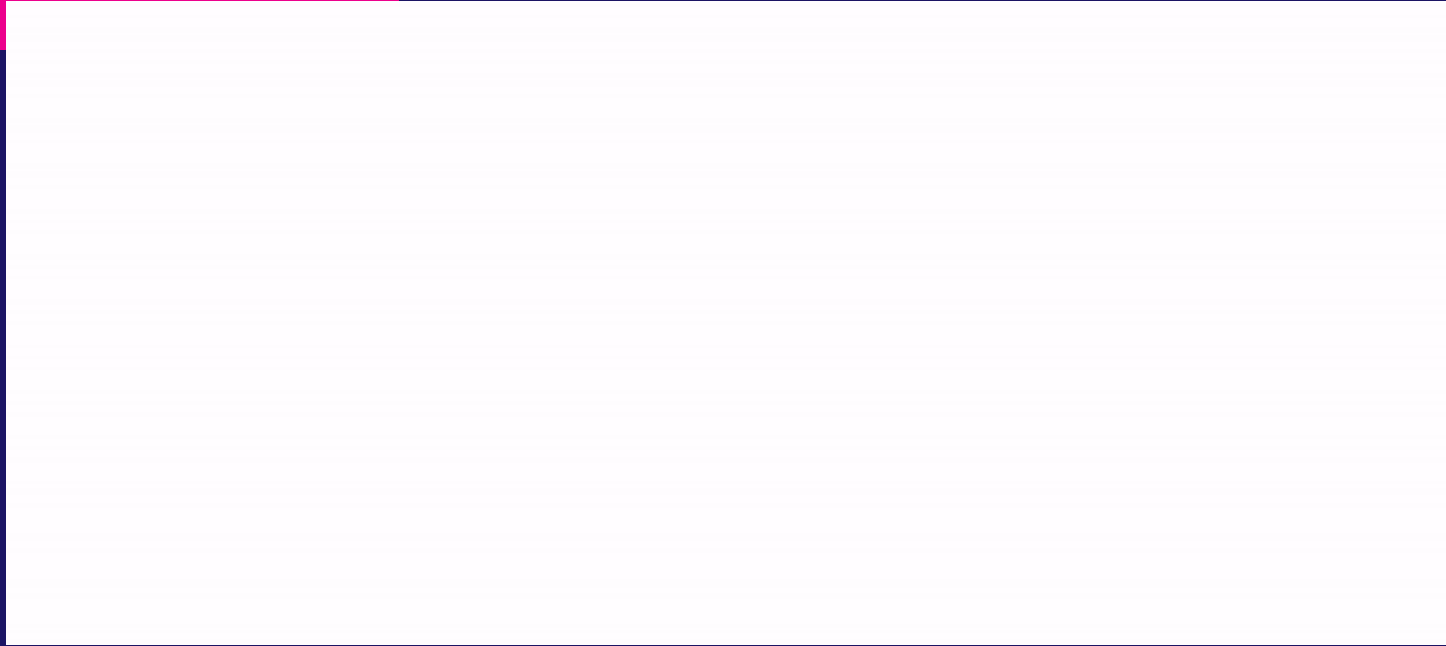
# INSERIR NA LISTA (INICIO)

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.



# LISTA ENCADEADA - EXERCÍCIOS

4) Escreva um programa em JAVA que receba valores inteiros e insira corretamente nas estruturas de dados conforme critérios a seguir:

- Números pares devem ser incluídos na pilha
- Números ímpares devem ser incluídos na fila
- Números negativos devem ser incluídos na lista encadeada

Após a entrada de dados, possibilite ao usuário exibir os itens existentes dentro de cada estrutura de dados, a quantidade de valores existentes e a média dos valores. Possibilite também ao usuário remover elementos e fazer uma busca em cada estrutura.



03

CORREÇÃO AC1 – PARTE 1

# QUESTÃO 1

Resposta:  
5, 10, 12, 2

Simule a execução do código a seguir e represente quais são os itens da fila (ordem: do fim para o início) após todos os comandos realizados:

\*\*\*Atenção: Considere que este código esteja dentro de uma classe com o método main e que a classe Fila está inserida dentro do mesmo projeto. Considere também que o código não possui erros.

```
Fila F = new Fila(5);  
F.enfileirar(12);  
F.enfileirar(10);  
F.enfileirar(3);  
F.enfileirar(2);  
F.enfileirar(Integer.parseInt(F.desenfileirar()));  
F.enfileirar(Integer.parseInt(F.desenfileirar()));  
F.enfileirar(5);  
F.enfileirar(8);  
System.out.println(F.desenfileirar());
```

# QUESTÃO 2

Resposta:

5, 7, 10, 6

Simule a execução do código a seguir e represente quais são os itens da pilha (ordem: base para o topo) após todos os comandos realizados:

\*\*\*Atenção: Considere que este código esteja dentro de uma classe com o método main e que a classe Pilha está inserida dentro do mesmo projeto. Considere também que o código não possui erros.

```
Pilha P = new Pilha(5);  
P.empilhar(5);  
P.empilhar(7);  
P.empilhar(10);  
P.empilhar(14);  
P.empilhar(P.desempilhar());  
P.empilhar(P.desempilhar());  
System.out.println(P.desempilhar());  
P.empilhar(6);  
P.empilhar(9);  
P.empilhar(20);  
System.out.println(P.desempilhar());
```

# QUESTÃO 3

Resposta:

PILHA: 5, 8, 9, 10, 11, 12, 55

FILA: 25, 36, 15

(Valor da questão: 2 pontos) Simule a execução do código a seguir e represente quais são os itens da pilha (ordem: base para o topo) e da fila (ordem: do fim para o início) após todos os comandos realizados:

\*\*\*Atenção: Considere que este código esteja dentro de uma classe com o método main e que a classe Fila e a classe Pilha estão inseridas dentro do mesmo projeto. Considere também que o código não possui erros.

```
Pilha P = new Pilha(10);  
Fila F = new Fila(10);  
P.empilhar(5);  
F.enfileirar(10);  
P.empilhar(8);  
P.empilhar(9);  
F.enfileirar(12);  
P.empilhar(F.desenfileirar());  
P.empilhar(11);  
P.empilhar(15);  
F.enfileirar((int) P.desempilhar());  
F.enfileirar(36);  
P.empilhar(25);  
F.enfileirar((int) P.desempilhar());  
P.empilhar(F.desenfileirar());  
P.empilhar(55);
```

# QUESTÃO 4

Com relação a estrutura de dados conhecida como FILA, julgue as afirmações a seguir:

I – A inserção de novos elementos é feita pelo fim da estrutura.

II – A fila implementa o critério de acesso denominado LIFO.

III – A retirada de elementos é feita pelo início da estrutura.

IV - A principal propriedade de uma fila é sua capacidade inverter a ordem de uma sequência.

É correto apenas o que se afirma em:

I, IV

I, II, III

I, III, IV

III, IV

I, III

# QUESTÃO 5

Considerando a estrutura de dados que utiliza o método LIFO, a operação de inserção é chamada de \_\_\_\_\_, enquanto a de exclusão é chamada de \_\_\_\_\_. Assinale a alternativa que preenche corretamente as lacunas na ordem em que aparecem:

- empilhamento e desempilhamento
- enfileiramento e desempilhamento
- enfileiramento e desenfileiramento
- empilhamento e desenfileiramento
- desempilhamento e enfileiramento

# QUESTÃO 6

A pilha é uma estrutura de dados que permite a inserção e remoção dos dados usando sempre a mesma política de acesso. Considere que as operações PUSH e POP são utilizadas para inserir e remover um dado da pilha, respectivamente. Para uma pilha denominada P que foi inicializada considere que as seguintes operações foram feitas: PUSH(P,2), PUSH(P,3), POP(P), PUSH(P,5), PUSH(P,10), PUSH(P,4), PUSH(P,2), POP(P), PUSH(P,1), PUSH(P,20) e POP(P). Qual o valor total da multiplicação dos elementos que permanecem dentro da pilha e qual o valor que está no topo da estrutura, respectivamente?

400, 1

2400, 20

9600, 2

800, 20

480, 3



# QUESTÃO 7

Considere que os valores a seguir foram inseridos em uma fila na seguinte ordem: 2, 3, 4, -1, 5, 6, -3, 4, -5. Considere também que os valores 6, 9, 10, 12, 14, 15, 17, 20, 25, 32 foram adicionados nessa respectiva ordem, em uma pilha vazia. Para cada valor removido da fila, caso seja positivo, será adicionado na pilha e caso o valor seja negativo, o seu valor absoluto representará a quantidade de elementos a serem removidos da pilha. Após a remoção de todos os elementos da fila, assinale a alternativa que representa os valores que se encontram na pilha (considere a ordem da base para o topo):

6, 9, 10, 12, 14, 15, 17

6, 5, 4, 3, 32, 25, 20, 17

17, 15, 14, 12, 10, 9, 6

6, 4, 3, 32, 15, 14, 12, 10

17, 20, 25, 32, 3, 4, 5, 6

# QUESTÃO 8

```
for (i=0;i<5;i++)
{
    for (j=0;j<4;j++)
    {
        if (vetor[j] < vetor[j+1])
        {
            aux = vetor[j];
            vetor[j] = vetor[j+1];
            vetor[j+1] = aux;
        }
    }
}
```

“Um array em Java é uma coleção ordenada que ocupa uma porção fixa e sequencial da memória. Além disso, é definido como uma estrutura homogênea, pois armazena um determinado tipo de dado. Esse, por sua vez, faz referências para objetos, valores de um tipo primitivo ou para outros arrays.” Considere que o usuário digitou os valores: 14, 40, 16, 22 e 60 para o array. Assinale a alternativa que contém o valor que será exibido quando executado o código Java a seguir.

14, 16, 22, 40 e 60.

28, 32, 44, 80 e 120.

60, 40, 22, 16 e 14.

120, 80, 44, 32 e 28.

60, 22, 16, 40, 14

# QUESTÃO 9

Um programador Java deseja utilizar uma *array* bidimensional do tipo *String* para armazenar os nomes (em **negrito**) mostrados na representação a seguir (os índices estão em **vermelho**):

Array bidimensional (matriz)

	0	1
0	<b>Ana</b>	<b>Pedro</b>
1	<b>Marcos</b>	<b>Paulo</b>

Para declarar a *array* e já armazenar os nomes em um ambiente de programação Java ideal, o programador deve escrever a instrução

```
String [ ] [ ] nomes = {"Marcos","Paulo"},{"Ana","Pedro"};
String [2] [2] nomes = {"Ana","Pedro"},{"Marcos","Paulo"};
String [ ] [ ] nomes = {"Ana","Pedro"},{"Marcos","Paulo"};
String [ ] [ ] nomes = {"Ana","Pedro"},{"Marcos","Paulo"};
String [2] [2] nomes = {"Ana","Pedro"},{"Marcos","Paulo"};
```

# Dúvidas?

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