



Entendendo a Lista Ligada

Serão apresentadas, a seguir, as operações de lista ligada para lista vazia, destruir lista ligada, contar nós, mostrar lista, elemento do início da lista ligada, elemento do final da lista ligada, inserir no início da lista ligada e inserir no final da lista ligada. Para as operações de lista ligada em pseudocódigo, utilizaremos as seguintes definições de registro:

// definição do tipo registro No com os campos abaixo

tipo No = registro

elemento \leftarrow 0 numérico_inteiro; // campo que armazena o elemento

prox \leftarrow nulo No; // campo que armazena o endereço do próximo nó

fimregistro;

```
class No
```



```
{
```

```
    int elemento;
```

```
    No prox;
```

```
    No (int elem)
```

```
    {
```

```
        elemento = elem;
```

```
        prox = null;
```

```
    }
```

```
}
```

Onde o campo elemento armazena o elemento e o campo prox é o ponteiro que armazena o endereço do próximo nó que contém o próximo elemento da lista ligada.

```
// definição do tipo registro ListaLigada com os campos abaixo
```

```
tipo ListaLigada = registro // o tipo registro chama-se ListaLigada
```

```
primeiro ← nulo No; // campo primeiro que armazena o primeiro elemento da  
lista
```

ultimo ← nulo No; // campo ultimo que armazena o último elemento da lista ligada



fimregistro;

```
class Listaligada
{
    No primeiro, ultimo;

    Listaligada ()
    {
        primeiro = null;
        ultimo = null;
    }
}
```

Onde o campo primeiro é um ponteiro que armazena o endereço do primeiro elemento da listaligada e o campo último é um ponteiro armazena o endereço do último elemento da lista ligada.

LISTA VAZIA E DESTRUIR DA LISTA LIGADA

ListaVazia é um módulo função sem parâmetros da operação lista vazia que retorna verdadeiro se a lista estiver vazia e retorna falso se a lista não estiver vazia.



lógico ListaVazia()

início_módulo

se (primeiro = nulo e ultimo = nulo)

então

retornar verdadeiro;

senão

retornar falso;

fimse;

fim_módulo;

```
public boolean ListaVazia( )
{
    if (primeiro == null && ultimo == null)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Destruir é um módulo procedimento sem parâmetros da operação destruir que simplesmente destrói uma lista ligada.



Destruir()

início_módulo

início <- nulo;

ultimo <- nulo;

fim_módulo;

```
public void Destruir( )  
{  
    primeiro = null;  
    ultimo = null;  
}
```

CONTAR NÓS E MOSTRAR LISTA LIGADA

ContarNos é um módulo função que é utilizado na operação inserir no meio que verifica e retorna quantos elementos possui a lista ligada.

numérico_inteiro ContarNos()

início_módulo

Declarar



tamanho \leftarrow 0 numérico_inteiro;

NoTemp \leftarrow primeiro No;

enquanto (NoTemp \neq nulo) faça

tamanho \leftarrow tamanho + 1;

NoTemp \leftarrow NoTemp.prox;

fimenquanto;

retornar tamanho;

fim_módulo;

```
public int ContarNos ( )
{
    int tamanho = 0;
    No NoTemp = primeiro;

    while (NoTemp != null)
    {
        tamanho = tamanho + 1;
        NoTemp = NoTemp.prox;
    }
    return tamanho;
}
```

MostraLista é um módulo procedimento da operação mostra lista ligada que mostra ao usuário todos os elementos da lista ligada.



MostrarLista()

início_módulo

Declarar

NoTemp \leftarrow primeiro No;

i \leftarrow 1 numérico_inteiro;

enquanto (NoTemp \neq nulo) faça

escrever ("Elemento ", NoTemp.elemento , " posição " , i);

NoTemp \leftarrow NoTemp.prox;

i \leftarrow i + 1;

fimenquanto;

fim_módulo;

```
public void MostrarLista( )
```

```
{
```

```
    int i = 1;
```

```
    No NoTemp = primeiro;
```

```
    while (NoTemp != null)
```

```
    {
```

```
        System.out.println("Elemento " + NoTemp.elemento  
                             + " posição " + i);
```

```
        NoTemp = NoTemp.prox;
```

```
        i = i + 1;
```

```
    }
```

```
}
```



ELEMENTO DO INÍCIO E DO FINAL DA LISTA LIGADA

ElementoInicio é um módulo procedimento da operação elemento do início que mostra ao usuário o primeiro elemento que está na lista ligada.

ElementoInicio()

início_módulo

se (não ListaVazia())

então

escrever ("O primeiro elemento da lista ligada é ", primeiro.elemento);

senão

escrever ("Lista Ligada vazia");

fimse;

fim_módulo;



```
public void ElementoInicio( )
{
    if (! ListaVazia())
    {
        System.out.println("O primeiro elemento é " +
                           primeiro.elemento);
    }
    else
    {
        System.out.println("Lista Ligada Vazia");
    }
}
```

ElementoFinal é um módulo procedimento da operação elemento do final que mostra ao usuário o último elemento que está na lista ligada.

ElementoFinal()

início_módulo

se (não ListaVazia())

então

escrever ("O último elemento da lista ligada é ", ultimo.elemento);

senão

escrever ("Lista Ligada vazia");

fimse;

fim_módulo;



```
public void ElementoFinal( )
{
    if (! ListaVazia())
    {
        System.out.println("O último elemento é " +
                           ultimo.elemento);
    }
    else
    {
        System.out.println("Lista Ligada Vazia");
    }
}
```

INSERIR NO INÍCIO E NO FINAL DA LISTA LIGADA

InserirInicio é um módulo procedimento da operação inserir no início que recebe como parâmetro um elemento do tipo No a ser inserido e insere este elemento no início da lista ligada.

InserirInicio (novoNo \uparrow No)

início_módulo

se (ListaVazia())

então

ultimo \leftarrow novoNo;

senão

novoNo.prox \leftarrow primeiro;



fimse;

primeiro \leftarrow novoNo;

fim_módulo;

```
public void InserirInicio (No novoNo)
{
    if (ListaVazia())
    {
        ultimo = novoNo;
    }
    else
    {
        novoNo.prox = primeiro;
    }
    primeiro = novoNo;
}
```

InserirFinal é um módulo procedimento da operação inserir no final que recebe como parâmetro um elemento do tipo No a ser inserido e insere este elemento no final da lista ligada.

InserirFinal (novoNo \uparrow No)

início_módulo

se (ListaVazia())

então

primeiro ← novoNo;



senão

ultimo.prox ← novoNo;

fimse;


ultimo ← novoNo;

fim_módulo;

```
public void InserirFinal (No novoNo)
{
    if (ListaVazia())
    {
        primeiro = novoNo;
    }
    else
    {
        ultimo.prox = novoNo;
    }
    ultimo = novoNo;
}
```

Atividade extra

Indicação de leitura:

Você pode utilizar o livro Estrutura de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++, da Ana Fernanda Gomes Ascencio e Graz Santos de Araújo, no capítulo 3 de análise da análise de complexidade das operações de Lista Ligada. 

Referência Bibliográfica

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos Teoria e Prática**. 3ª ed. Editora Cammpus, 2012.

GOODRICH, M. T.; TAMASSIA, R. **Estruturas de Dados & Algoritmos em Java**. 5ª ed. Editora Grupo A: Bookman, 2013.

Atividade Prática 11 – Entendendo a Lista Ligada

Título da Prática: Operações com Lista Ligada em Java

Objetivos: Entender como utilizar o netbeans para desenvolver programas em Java para manipular e desenvolver as operações com Lista Ligada

Materiais, Métodos e Ferramentas: Computador, netbeans, Java.



Atividade Prática

O Algoritmo com as operações de Lista Ligada para desenvolver um algoritmo pode ser escrito como segue.

Desenvolva o programa em Java deste algoritmo no NetBeans.

Algoritmo No

início_algoritmo

tipo No = registro

elemento \leftarrow 0 **numérico_real**;

prox \leftarrow nulo No;

fimregistro;

fimalgoritmo.

Algoritmo ListaLigada

início_algoritmo

// definição do tipo registro ListaLigada com os campos abaixo

tipo ListaLigada = **registro** // o tipo registro chama-se ListaLigada



primeiro \leftarrow nulo **No**;

ultimo \leftarrow nulo **No**;

fimregistro;

lógico ListaVazia()

início_módulo

se (primeiro = **nulo** e ultimo = **nulo**)

então

retornar verdadeiro;

senão

retornar falso;

fimse;

fim_módulo;

InserirInicio (novoNo \uparrow **No**)

início_módulo

se (ListaVazia())

então

ultimo ← novoNo;



senão

novoNo.prox ← primeiro;

fimse;

primeiro ← novoNo;

fim_módulo;

InserirFinal (novoNo ↑ **No**)

início_módulo

se (ListaVazia())

então

primeiro ← novoNo;

senão

ultimo.prox ← novoNo;

fimse;

ultimo ← novoNo;

fim_módulo;

numérico_inteiro ContarNos()

início_módulo



Declarar

tamanho \leftarrow 0 **numérico_inteiro**;

NoTemp \leftarrow primeiro **No**;

enquanto (NoTemp \neq nulo) **faça**

tamanho \leftarrow tamanho + 1;

NoTemp \leftarrow NoTemp.prox;

fimenquanto;

retornar tamanho;

fim_módulo;

InserirMeio(NovoNo \uparrow **No**, posicao **numérico_inteiro**)

início_módulo

Declarar

NoTemp \leftarrow primeiro **No**;

NroNos, posAux \leftarrow 1 **numérico_inteiro**;

NroNos \leftarrow ContarNos();

se (posicao \leq 1)



então

InserirInicio(NovoNo);

senão

se (posicao $>$ NroNos)

então

InserirFinal(NovoNo);

senão

enquanto (posAux $>$ (posicao -1))

NoTemp \leftarrow NoTemp.prox;

posAux \leftarrow posAux + 1;

fimenquanto;

NovoNo.prox \leftarrow NoTemp.prox;

NoTemp.prox \leftarrow NovoNo;

fimse;

fimse;

fim_módulo;

Remover (elemento **numérico_real**)



Declarar

NoTemp \leftarrow primeiro **No**;

NoAnt \leftarrow **nulo** No;

se (primeiro.elemento = elemento)

então

primeiro \leftarrow primeiro.prox;

senão

enquanto (NoTemp \neq **nulo** e NoTemp.elemento \neq elemento)

NoAnt \leftarrow NoTemp;

NoTemp \leftarrow NoTemp.prox;

fimenquanto;

se (NoTemp \neq **nulo**)

então

NoAnt.prox \leftarrow NoTemp.prox;

fimse;

se (NoTemp = ultimo)

então

ultimo ← NoAnt;



fimse;

fimse;

fim_módulo;

ElementoInicio()

início_módulo

se (não ListaVazia())

então

escrever ("O primeiro elemento da lista ligada é " , primeiro.elemento);

senão

escrever ("Lista Ligada vazia");

fimse;

fim_módulo;

ElementoFinal()

início_módulo

se (não ListaVazia())

então

escrever ("O último elemento da lista ligada é " , ultimo.elemento);



senão

escrever ("Lista Ligada vazia");

fimse;

fim_módulo;

↑ **No** BuscarNo (elemento **numérico_real**)

início_módulo

Declarar

$i \leftarrow 1$ **numérico_inteiro**;

NoTemp \leftarrow primeiro **No**;

enquanto (NoTemp \neq **nulo**) **faça**

se (NoTemp.elemento = elemento)

então

escrever ("No " , NoTemp.elemento , " posição " , i);

retornar NoTemp;

fimse;

$i \leftarrow i + 1$;

NoTemp \leftarrow NoTemp.prox;



fimenquanto;

retornar nulo;

fim_módulo;

MostrarLista()

início_módulo

Declarar

NoTemp \leftarrow primeiro **No**;

i \leftarrow 1 **numérico_inteiro**;

enquanto (NoTemp \neq nulo) **faça**

escrever ("Elemento " , NoTemp.elemento , " posição " , i);

NoTemp \leftarrow NoTemp.prox;

i \leftarrow **i** + 1;

fimenquanto;

fim_módulo;

fimalgoritmo.



Gabarito Atividade Prática

```
class No
```

```
{
```

```
    double elemento;
```

```
    No prox;
```

```
    No (double elem)
```

```
{
```

```
    elemento = elem;
```

```
    prox = null;
```

```
}
```

```
}
```

```
class ListaLigada
```

```
{
```

```
    No primeiro, ultimo;
```



ListaLigada ()

{

primeiro = **null**;

ultimo = **null**;

}

public boolean ListaVazia()

{

if (primeiro == **null** && ultimo == **null**)

{

return true;

}

else

{

return false;

}

}

public void InserirInicio (No novoNo)



```
{  
  
    if (ListaVazia())  
  
    {  
  
        ultimo = novoNo;  
  
    }  
  
    else  
  
    {  
  
        novoNo.prox = primeiro;  
  
    }  
  
    primeiro = novoNo;  
  
}
```

public void InserirFinal (No novoNo)

```
{  
  
    if (ListaVazia())  
  
    {  
  
        primeiro = novoNo;  
  
    }
```

else



{

ultimo.prox = novoNo;

}

ultimo = novoNo;

}

public int ContarNos ()

{

int tamanho = 0;

No NoTemp = primeiro;

while (NoTemp **!=** null)

{

tamanho = tamanho + 1;

NoTemp = NoTemp.prox;

}

return tamanho;

}



```
public void InserirMeio(No NovoNo, int posicao)
```

```
{
```

```
    No NoTemp = primeiro;
```

```
    int NroNos, posAux = 1;
```

```
    NroNos = ContarNos();
```

```
    if (posicao <= 1)
```

```
    {
```

```
        InserirInicio(NovoNo);
```

```
    }
```

```
    else
```

```
    {
```

```
        if (posicao > NroNos)
```

```
        {
```

```
            InserirFinal(NovoNo);
```

```
        }
```

```
    else
```

```
    {
```

```
while (posAux < (posicao - 1))
```



```
{
```

```
    NoTemp = NoTemp.prox;
```

```
    posAux = posAux + 1;
```

```
}
```

```
NovoNo.prox = NoTemp.prox;
```

```
NoTemp.prox = NovoNo;
```

```
}
```

```
}
```

```
}
```

```
public void Remover (double elemento)
```

```
{
```

```
    No NoTemp = primeiro;
```

```
    No NoAnt = null;
```

```
if (primeiro.elemento == elemento)
```

```
{
```

```
    primeiro = primeiro.prox;
```

```
}
```



```
else
```

```
{
```

```
while (NoTemp != null && NoTemp.elemento != elemento)
```

```
{
```

```
    NoAnt = NoTemp;
```

```
    NoTemp = NoTemp.prox;
```

```
}
```

```
if(NoTemp != null)
```

```
{
```

```
    NoAnt.prox = NoTemp.prox;
```

```
}
```

```
if (NoTemp == ultimo)
```

```
{
```

```
    ultimo = NoAnt;
```

```
}
```

```
}
```

```
}
```

public void ElementoInicio()



```
{  
  
    if (! ListaVazia())  
  
    {  
  
        System.out.println("O primeiro elemento é " +  
                           primeiro.elemento);  
  
    }  
  
    else  
  
    {  
  
        System.out.println("Lista Ligada Vazia");  
  
    }  
  
}
```

public void ElementoFinal()

```
{  
  
    if (! ListaVazia())  
  
    {  
  
        System.out.println("O último elemento é " +  
                           ultimo.elemento);  
  
    }  
  
}
```

else



{

System.out.println("Lista Ligada Vazia");

}

}

public No BuscarNo (**double** elemento)

{

int i = 1;

No NoTemp = primeiro;

while (NoTemp **!=** null)

{

if (NoTemp.elemento == elemento)

{

System.out.println("No " + NoTemp.elemento + " posição "
+ i);

return NoTemp;

}

```
i = i + 1;
```



```
NoTemp = NoTemp.prox;
```

```
}
```

```
return null;
```

```
}
```

```
public void MostrarLista( )
```

```
{
```

```
    int i = 1;
```

```
    No NoTemp = primeiro;
```

```
    while (NoTemp != null)
```

```
    {
```

```
        System.out.println("Elemento " + NoTemp.elemento + " posição  
            " + i);
```

```
        NoTemp = NoTemp.prox;
```

```
        i = i + 1;
```

```
    }
```

```
}
```




Ir para exercício