UFSC-CTC-INE

INE5384 - Estruturas de Dados

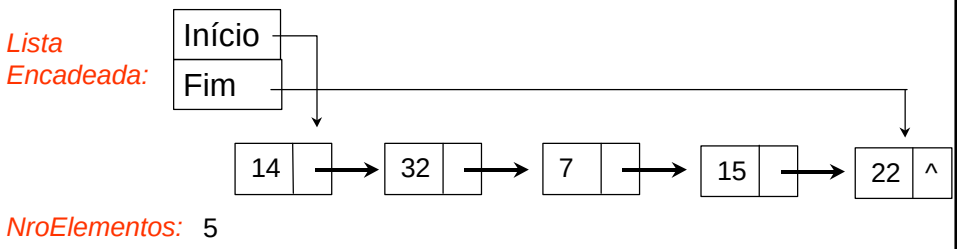
Listas Encadeadas (2)

Prof. Ronaldo S. Mello

2002/2

Lista Classificada Encadeada

- Supõe-se os seguintes atributos:
  - ListaEncadeada (referência a objeto da classe *ListaEncadeada*)
  - NroElementos
- Exemplo:



Implementação

```
Classe ListaClassificadaListaEncadeada
   Implementa ListaClassificada
início
   listaEncadeada ListaEncadeada;
   NroElementos inteiro;

   Construtor ListaClassificadaListaEncadeada ()
   início
        listaEncadeada ← NOVO ListaEncadeada();
        NroElementos ← 0;
   fim
   . . .
```

Inserção de Elemento

```
Classe ListaClassificadaListaEncadeada
   Implementa ListaClassificada
início
   listaEncadeada ListaEncadeada;
   NroElementos inteiro;

   Método Insere (objeto ObjetoComparável)
   início
        listaEncadeada.InsereNoFinal(objeto);
        NroElementos ← NroElementos + 1;
   fim
   . . .
```

Exclusão de Elemento

```
Classe ListaClassificadaListaEncadeada
   Implementa ListaClassificada
início
   listaEncadeada ListaEncadeada;
   NroElementos inteiro;

   Método Exclui (objeto ObjetoComparável)
   início
         se NroElementos = 0 então Exceção EstruturaVazia();
         listaEncadeada.Exclui(objeto);
         NroElementos ← NroElementos - 1;
   fim
   . . .
```

Pesquisa na Lista

- *retPosição (posição inteiro)* retorna ObjetoComparável; (método *get* do livro)
- *encontra (objeto ObjetoComparável)* retorna ObjetoComparável; (método *find* do livro)
- *ehMembro (objeto ObjetoComparável);* (método *isMember* do livro)
- *procurePosição (objeto ObjetoComparável)* retorna Cursor; (método *findPosition* do livro)

Retorna Posição do Elemento

```
Classe ListaClassificadaListaEncadeada
    Implementa ListaClassificada
início

    Método retPosição (posição inteiro) retorna ObjetoComparável;
    início
            ListaEncadeada.Elemento ptr;
            i inteiro;

            se NroElementos = 0 então Exceção EstruturaVazia();
            se posição < 0 ou posição >= NroElementos então
                    Exceção OperaçãoIlegal();
            ptr = listaEncadeada.ObtemInício();
            para i de 0 até (posição - 1) faça ptr = ptr.ObtemProx();
            retorna (ObjetoComparável) ptr.ObtemDado();
    fim
. . .
```

Procura Posição do Elemento

```
. . .
    Método procuraPosição (objeto ObjetoComparável) retorna Cursor;
    início
            ListaEncadeada.Elemento ptr;
            i inteiro;

            se NroElementos = 0 então Exceção EstruturaVazia();
            ptr = listaEncadeada.ObtemInício();
            enquanto ptr ≠ NULL faça
            início
                    se objeto.ehIg(ptr.ObtemDado()) = VERDADEIRO então
                            retorna NOVO MeuCursor (ptr);
                    ptr = ptr.ObtemProx();
            fim;
            retorna NULL;
    fim;
. . .
```

o cursor de uma lista encadeada mantém um elemento *(ao contrário do cursor do vetor que mantinha uma posição)*

Complexidade de Algoritmos

- Medida de complexidade de tempo de algoritmos: notação *O()*
- Avalia o tempo de execução do algoritmo (no pior caso) em função do número de dados (*n*) na estrutura de dados
- Exemplos:
  - O(1): executa sempre em tempo constante;
  - O(n): no pior caso, analisa os *n* dados da ED
  - $O(n^2)$: no pior caso, analisa *n.n* vezes os dados da ED
- $O(1) < O(n) < O(n^2)$

Vetor X Encadeamento

Operação	Vetor	Encadeamento
Inserção início	O(n)	**O(1)**
Inserção final	O(1)	O(1)
Inserção posição "x"	O(n)	O(n)
Pesquisa elemento "x"	O(n)	O(n)
Pesquisa posição "x"	**O(1)**	O(n)
Exclusão início	O(n)	**O(1)**
Exclusão final	**O(1)**	O(n)
Exclusão posição ou elemento "x"	O(n)	O(n)

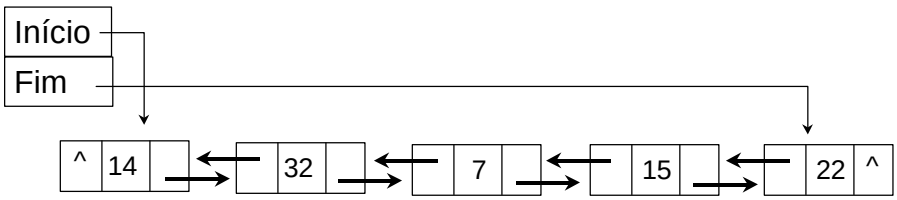
Vetor X Encadeamento

Operação	Vetor	Encadeamento
Inserção início	O(n)	O(1)
Inserção final	O(1)	O(1)
Inserção posição "x"	O(n)	O(n)
Pesquisa elemento "x"	O(n)	O(n)
Pesquisa posição "x"	O(1)	O(n)
Exclusão início	O(n)	O(1)
Exclusão final	O(1)	**O(n)**
Exclusão posição ou elemento "x"	O(n)	O(n)

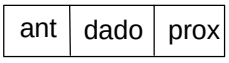
o algoritmo poderia ser melhorado se fosse usada uma **lista duplamente encadeada**

Lista Duplamente Encadeada

Lista Duplamente Encadeada:



Elemento:



Implementação

```
Classe ListaDuplamenteEncadeada
início
    inicio, fim ElementoDuplamenteEncadeado;

    Classe ElementoDuplamenteEncadeado
    início
        dado Object;
        ant, prox  ElementoDuplamenteEncadeado;

        Construtor Elemento (dado object, ant Elemento, prox Elemento)
        início
            this.dado ← dado;
            this.ant ← ant;
            this.prox ← prox;
        fim
    . . .
```

Implementação

```
    . . .
        Método ObtémProx() retorna ElementoDuplamenteEncadeado;
        ...
        Método ObtémAnt() retorna ElementoDuplamenteEncadeado;

    . . .
    fim;
    Construtor ListaDuplamenteEncadeada ();
    início
        início ← null;      fim ← null;
    fim;

    Método ObtemInício() retorna ElementoDuplamenteEncadeado;
    ...
    Método ObtemFim() retorna ElementoDuplamenteEncadeado;
    ...

fim;
```

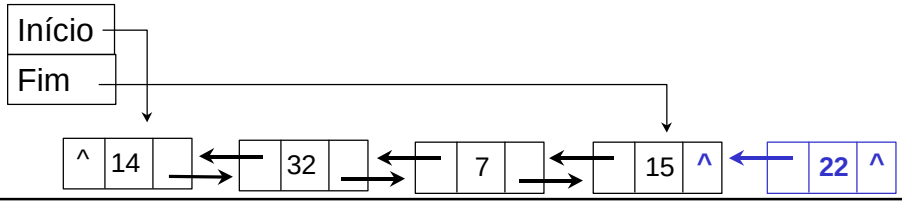
Exclusão no Final

- Exceções a tratar?
  - lista vazia
- Como proceder?

  fim ← fim.ant;

  (fim.prox.destroy;)

  fim.prox ← NULL;



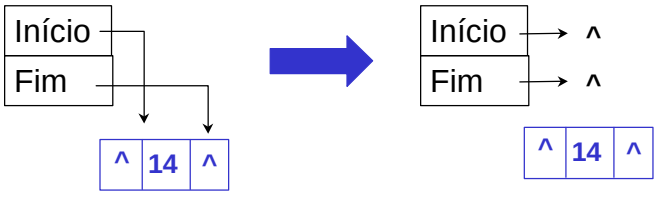
Exclusão no Final

- Exceções a tratar?
  - lista vazia
- Se existe apenas 1 elemento na lista?

  fim ← fim.ant;

  se fim = NULL então inicio ← NULL

  senão fim.prox ← NULL;



Implementação

```
Método excluiNoFinal ();
início

    se inicio = NULL então Exceção EstruturaVazia();
    fim ← fim.ant;
    se fim = NULL então inicio ← NULL
    senão fim.prox ← NULL;
fim;
```

Complexidade: *O(1)*

Exercícios

- Implementar na classe *ListaDuplamenteEncadeada*:
  - *excluiNoInício();*
  - *insereNoInício(objeto Objeto);*
  - *insereNoFinal(objeto Objeto);*
  - *exclui(objeto Objeto);*
  - *incluiPosição(objeto Objeto, posição inteiro);*
  - *imprimeLista();*
  - *imprimeListaOrdemInversa();*