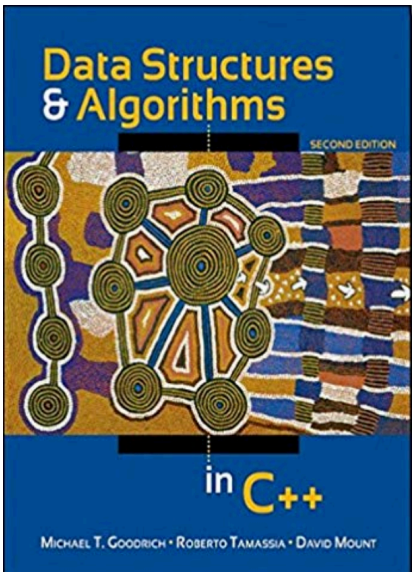
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS I – Aula 07 – 2º SEMESTRE/2018
PROF. LUCIANO SILVA/MARCIO PORTO FEITOSA

TEORIA: LISTAS LIGADAS (PARTE II)

Nossos **objetivos** nesta aula são:

- conhecer o tipo abstrato de dados lista duplamente ligada
- conhecer e implementar a estrutura de dados lista duplamente ligada



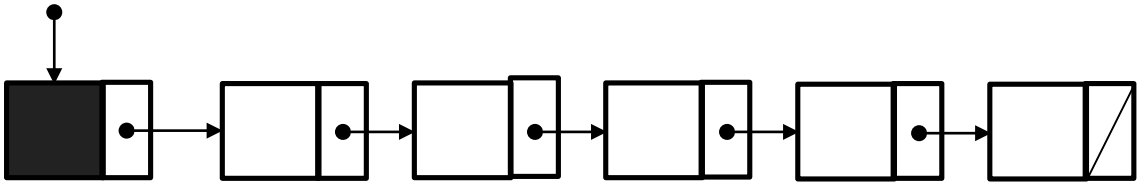
Para esta aula, usamos como referência a **Seção 3.3 (Doubly Linked Lists)** do nosso livro-texto:

GOODRICH,M., **Data Structures and Algorithms**. 2.ed. New York: Wiley, 2011.

Não deixem de ler esta seção depois desta aula!!

LISTAS DUPLAMENTE LIGADAS COMO TIPO ABSTRATO DE DADO

- Na aula anterior, estudamos as listas (**simplemente**) ligadas, formadas através de elementos chamados **nós**. Cada nó possui uma informação (**elem**) e um apontador (**prox**) para o próximo nó. O início da lista era identificado através de um nó especial chamado **cabeça**.



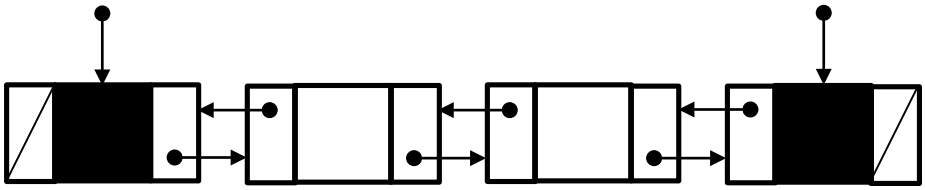
cabeça elem prox

- Embora a lista ligada nos permita gerenciar melhor a memória do que um vetor, a busca por elementos na lista pode não eficiente. Por exemplo, se sempre tivermos que

procurar por elementos que estejam próximos do final da lista, teríamos que passar por quase todos os elementos da lista a partir de seu início.

- Neste contexto, podemos gastar um pouco mais de memória e criar uma estrutura que nos permita percorrer a lista nos dois sentidos possíveis: **da cabeça para o final e do final para a cabeça**. Este tipo abstrato chama-se **lista duplamente ligada** (ou **duplamente encadeada**).

- Estruturalmente, uma lista duplamente ligada faz o uso de um novo apontador (**prev**) para indicar o elemento anterior (ou prévio), além de um apontador especial para indicar o final da lista (**final**).



cabeça prev elem prox final

- No TAD Lista Duplamente Ligada clássico, poderemos realizar as seguintes operações:

  - consultar se a lista está vazia
  - obter o primeiro elemento da lista (cabeça)
  - **obter o último elemento da lista (final)**
  - adicionar elemento no início da lista
  - **adicionar elemento no final da lista**
  - remover um elemento do início da lista
  - **remover um elemento do final da lista**

IMPLEMENTAÇÃO DA ESTRUTURA DE DADOS LISTA DUPLAMENTE LIGADA EM C++

- A primeira abstração a ser implementada refere-se ao nó da lista, acrescentando o atributo para o nó anterior:

```
#ifndef DNO_H
#define DNO_H

template <typename E>
class DNo{
   private:
      E elem; // informação do no
      DNo<E>* prox; // próximo elemento
      DNo<E>* prev; // elemento anterior
      friend class ListaLigada<E>;
};

#endif
```

dno.h

- Em seguida, vamos preparar a estrutura da lista ligada:

```
#ifndef LISTA_DUPLAMENTE_LIGADA_H
#define LISTA_DUPLAMENTE_LIGADA_H
#include "dno.h"

template <typename E>
class ListaDuplamenteLigada{

   private:
     DNo<E> * cabeca; // inicio da lista ligada (head)
     DNo<E> * fim;    // final da lista ligada (trailer)

   public:
     ListaDuplamenteLigada();
     ~ListaDuplamenteLigada();
     bool vazia() const;
     const E& inicio() const;
     const E& final() const;
     void insereInicio(const E& e);
     void insereFinal(const E& e);
     void removeInicio();
     void removeFinal();
};

#endif
```

lista_duplamente_ligada.h

EXERCÍCIO TUTORIADO

Implemente o construtor e destruidor para a classe Lista Duplamente Ligada.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método vazia(), que verifica se a lista contém elementos ou não.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente os método inicio() e final(), que devolve o primeiro elemento da lista (cabeça) e o último elemento da lista, respectivamente.

EXERCÍCIO TUTORIADO

Faça um esquema gráfico de como inserir um novo elemento no início da lista. Implemente, com base neste esquema, o método insereInicio(const E& e), que insere o elemento e na primeira posição da lista.

EXERCÍCIO TUTORIADO

Faça um esquema gráfico de como inserir um novo elemento no final da lista. Implemente, com base neste esquema, o método insereFinal(const E& e), que insere o elemento e na primeira posição da lista.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

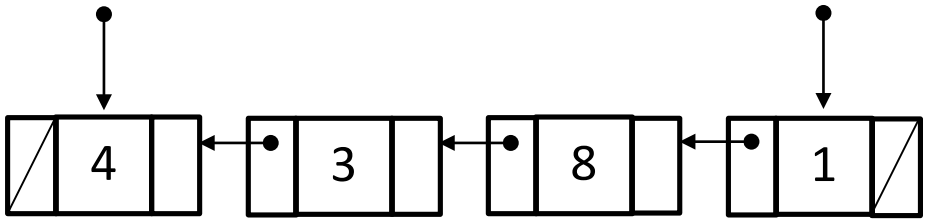
Faça um esquema gráfico de como remover o elemento do início da lista. Implemente o método removeInicio(), que remove o primeiro elemento da lista.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Faça um esquema gráfico de como remover o elemento do final da lista. Implemente o método removeFinal(), que remove o último elemento da lista.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente uma função main() para construir a lista duplamente ligada abaixo contendo os números inteiros abaixo. Faça diversos testes com consulta, inserção e remoção no início e final da lista.



ATIVIDADE DE LABORATÓRIO

Um polinômio é uma estrutura algébrica bastante importante em vários ramos da Computação e, em particular, em Computação Simbólica, que utiliza diversos recursos de Computação Algébrica para calcular derivadas, integrais e vários objetos em Matemática discreta. Nesta atividade, vamos implementar polinômios em listas ligadas.

Genericamente, um polinômio é uma soma formal como abaixo:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 x^0$$

Um exemplo prático seria $p(x) = 3x^2 + 2x \quad - 4$. O valor de n indica o grau do polinômio. Um polinômio é formado pela soma de vários monômios, que possuem um coeficiente a_i e um expoente i. Assim, seu desafio será:

(1) Criar um tipo **Monomio** para representar cada monômio de um polinômio

(2) Construir uma **lista duplamente ligada de monômios** para representar um polinômio, criando uma classe Polinomio que herda de ListaLigada:

```
class Polinomio: public ListaDuplamenteLigada{
    ....
}
```

(3) Incluir, na classe Polinomio, um método chamado calcula(double k) que recebe o valor k e devolve o valor de p(k), onde seja, o resultado da soma dos monômios com x=k. O seu método calcula deve percorrer simultaneamente da cabeça para o final e do final para a cabeça. Quando os dois apontadores se encontrarem, o cálculo deverá ser finalizado. **Isto mostrará que conseguimos calcular o valor de um polinômio na metade do tempo utilizado por uma lista ligada simples.**

EXERCÍCIOS EXTRA-CLASSE

1. Refatore a classe ListaDuplamenteLigada para permitir a troca da cabeça e final da lista duplamente.

2. Refatore a classe ListaDuplamenteLigada para que, dado um apontador para um determinado nó da lista ligada, remova este nó da lista. Observe que este nó pode estar no início, meio ou final da lista.

3. Refatore a classe ListaDuplamenteLigada para que, dado um apontador para um determinado nó da lista ligada, insira um novo nó imediatamente após este apontador.

4. Refatore a classe ListaDuplamenteLigada para que, dado um apontador para um determinado nó da lista ligada, insira um novo nó imediatamente antes deste apontador.