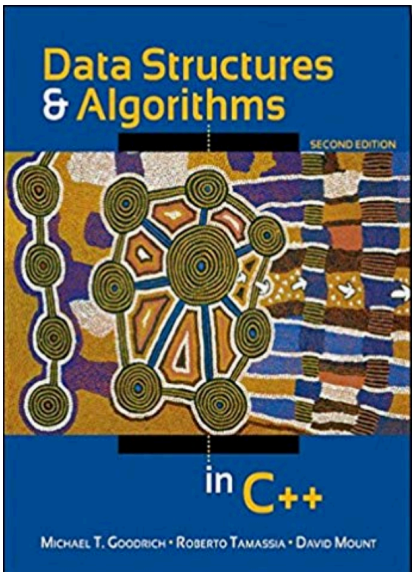
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS I – Aula 06 – 2º SEMESTRE/2018
PROF. LUCIANO SILVA/MARCIO PORTO FEITOSA

TEORIA: LISTAS LIGADAS (PARTE I)

Nossos **objetivos** nesta aula são:

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



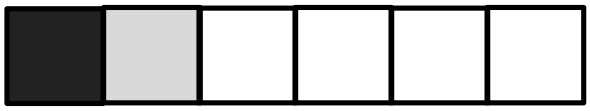
Para esta aula, usamos como referência a **Seção 3.2 (Single Linked Lists)** do nosso livro-texto:

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

Não deixem de ler esta seção depois desta aula!!

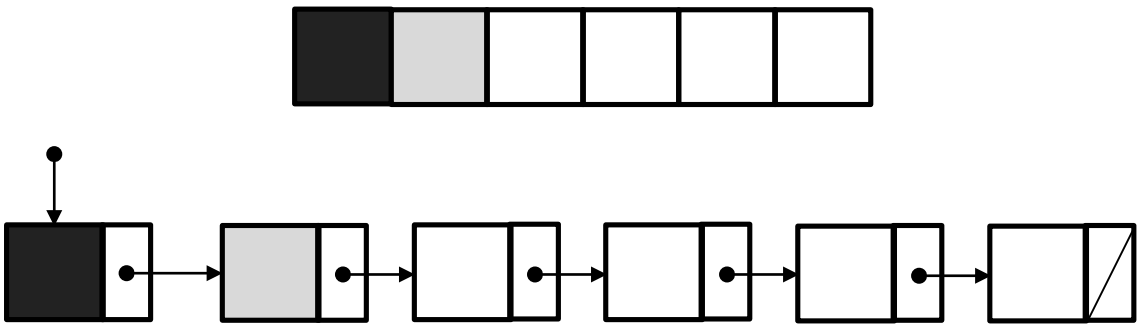
LISTAS LIGADAS COMO TIPO ABSTRATO DE DADO

- Na aula anterior, estudamos o TAD Vetor e sua realização como estrutura de dados vetor. Um dos principais defeitos desta estrutura é que sempre precisamos alocar todo o tamanho do vetor no início do programa, mesmo que no início só estejamos pensando em usar algumas posições iniciais.



- Porém, por outro lado, esta alocação contígua de memória tem uma vantagem, pois o **próximo elemento** do vetor pode ser acessado somando-se uma posição de memória em relação à posição anterior.

- Em termos de alocação de memória, poderíamos ter uma maneira alternativa de ir alocando as posições à medida que elas forem se tornando necessárias. Como as alocações de memória podem estar em posições distantes, vamos precisar usar um operador (**prox**) para saber onde está a próxima posição. Assim, um vetor transforma-se-ia numa lista ligada de posições da seguinte maneira:



cabeça elem prox

- Cada elemento da lista ligada é chamada de um nó. Alocações parciais sempre serão realizadas com base no tamanho de um nó. Um nó contém uma informação (elem) e um apontador para o próximo elemento (prox). O primeiro nó de uma lista é chamada **cabeça da lista**.

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

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

IMPLEMENTAÇÃO DA ESTRUTURA DE DADOS LISTA LIGADA EM C++

- A primeira abstração a ser implementada refere-se ao nó da lista:

```
#ifndef NO_H
#define NO_H

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

#endif
```

no.h

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

```
#ifndef LISTA_LIGADA_H
#define LISTA_LIGADA_H
#include “no.h”

template <typename E>
class ListaLigada{
   private:
     No<E> cabeca; // inicio da lista ligada (head)
   public:
     ListaLigada();
     ~ListaLigada();
     bool vazia() const;
     const E& inicio() const;
     void insereInicio(const E& e);
     void removeInicio();
};

#endif
```

lista_ligada.h

EXERCÍCIO TUTORIADO

Implemente o construtor e destruidor para a classe Lista 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 o método inicio(), que devolve o primeiro elemento da lista (cabeça).

EXERCÍCIO TUTORIADO

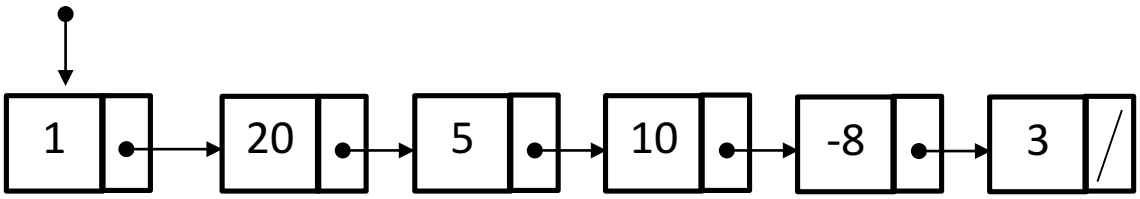
Implemente o método insereInicio(const E& e), que insere o elemento e na primeira posição da lista.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método removeInicio(), que remove o primeiro elemento da lista.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente uma função main() para construir a lista ligada abaixo, de números inteiros:



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:

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

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 ligada de monômios** para representar um polinômio, criando uma classe Polinomio que herda de ListaLigada:

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

(3) Incluir, na classe Polinomio, um método chamado mostra() que imprime todos os elementos (monômios) do polinômio.

(4) 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.

EXERCÍCIOS EXTRA-CLASSE

1. Refatore a classe ListaLigada para permitir a inserção de elementos no final da lista.

2. Refatore a classe ListaLigada para permitir remoção do último elemento da lista.

3. Refatore a classe ListaLigada para permitir a consulta do último elemento da lista.

4. Refatore a classe ListaLigada para inverter a posição dos seus elementos. O últimos elementos serão os primeiros e vice-versa.