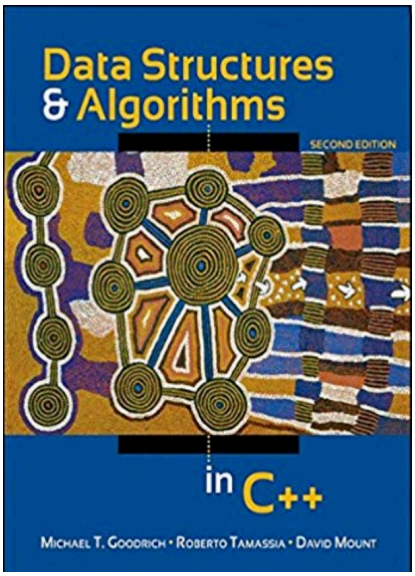
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS I – Aula 05 – 2º SEMESTRE/2018
PROF. LUCIANO SILVA/MARCIO PORTO FEITOSA

TEORIA: VETORES (ARRAYS)

Nossos **objetivos** nesta aula são:

- conhecer o tipo abstrato de dados Vetor
- conhecer e implementar a estrutura de dados vetro em C++



Para esta aula, usamos como referência a **Seção 3.1 Capítulo 3** 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!!

VETOR COMO TIPO ABSTRATO DE DADOS

- Um **vetor (ou array)** é uma coleção linear de um **número fixo** de componentes que, normalmente, são do mesmo tipo. Este tipo permite realizar as seguintes operações:
  - Inserir um elemento genérico no tipo numa determinada posição
  - Remover um elemento genérico de uma determinada posição
  - Verificar se há espaço para que possamos inserir novos elementos no vetor.
  - Verificar se um elemento genérico está no vetor
  - Verificar se o vetor não está vazio

As próximas seções irão detalhar a estrutura de dados Vetor.

VETOR COMO ESTRUTURA DE DADOS

- O primeiro passo é definir uma estrutura de classe para representar a estrutura de dados Vetor:

```
#ifndef VETOR_H
#define VETOR_H

template <typename T>
class Vetor{

   private:
     T* v;    // vetor de elementos do tipo T
     int tam;  // tamanho do vetor

   public:
     Vetor(int t);
     ~Vetor();
     int tamanho();
     T elemento (int i);
     bool procura(T elem);
     void insere (T d,int i);
     void remove (int i);
};

#endif
```

- A construção com template <typename T> permite construir **vetor de tipos genéricos** (polimorfismo paramétrico).

- As implementações dos primeiros métodos são mostradas abaixo:

```
#include <iostream>
#include “vetor.h”

template <typename T>
Vetor<T>::Vetor(int t)
{
  v=new T[t];  // aloca dinamicamente o vetor
  if (v==NULL)
    throw new std::string("memoria insuficiente");
  tam=t;
}

template <typename T>
Vetor<T>::~Vetor()
{
  delete v;    // remove a alocação dinâmica da memória
}


template <typename T>
int Vetor<T>::tamanho()
{
  return tam;
}
```

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método **T elemento (int i);** . Caso a posição esteja ocupada, devolve o elemento presente nela. Caso contrário, devolver NULL.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método **bool procura (T elem);** . Caso o elemento elem esteja no vetor, o resultado da chamada da função deve ser true e, false, caso contrário.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método **void insere (T elemento, int i);** . Se a posição for válida, substituir o elemento pelo parâmetro fornecido. Se o indice for inválido, lançar uma exceção.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implemente o método **void remove (int i);** . Se a posição a ser removida for válida, proceder à operação, deslocando todos os elementos do vetor para cobrir esta posição, caso necessário. Caso contrário, lançar uma exceção.

EXERCÍCIO DE LABORATÓRIO

Uma operação bastante importante em vetor é a junção, que permite agrupar dois vetores em um só. Acrescente, na classe Vetor, um método chamado **agrupamento**, que recebe como parâmetro um outro vetor e insere todos os seus elementos no final do vetor que invocou o método de agrupamento.

Implemente este método em C++.

EXERCÍCIOS EXTRA-CLASSE (PARA ENTREGA NO MOODLE)

1. Crie instâncias da classe Vetor com os tipos abstratos que vimos antes: Natural, Inteiro, Racional, Real e Complexo.

2. O método insere(...) implementado em aula poderia ser substituído pelo operador [ ] , através de sobrecarga. Refatorar este método para fazer esta substituição.