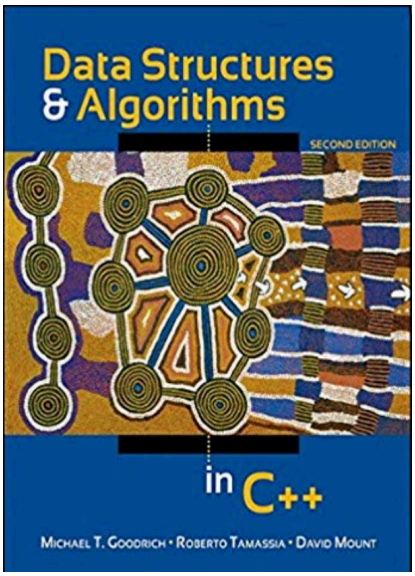
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS I – Aula 01 – 2º SEMESTRE/8
PROF. LUCIANO SILVA/MARCIO PORTO FEITOSA

TEORIA: TIPOS ABSTRATOS DE DADOS e ESTRUTURAS DE DADOS

Nossos **objetivos** nesta aula são:

- conhecer a noção de tipo abstrato de dado (TAD)
- conhecer a noção de estrutura de dados
- implementar estruturas de dados de tipos abstratos de dados básicos em C++



Para esta aula, usamos como referência o **Capítulo 1** do nosso livro-texto:

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

Não deixem de ler este capítulo depois desta aula!

TIPOS ABSTRATOS DE DADOS E ESTRUTURAS DE DADOS

- Em diversas linguagens de programação, encontramos os mais variados **tipos de dados**: int, float, double, char, dentre outros.

- Normalmente, podemos **estender os tipos básicos** suportados por uma linguagem. Uma das maneiras para se fazer isto é especificar formalmente (geralmente, de forma algébrica) qual é o **conjunto subjacente** ao tipo e quais são as **operações suportadas** pelo tipo.

- Por exemplo, vamos considerar um novo tipo de dados chamado NATURAL, para representar números do conjunto dos números naturais e sua principal operação(+):

$$\mathbb{N}=\{0,1,2,3,...\}$$

- Formalmente, podemos recorrer à Algebra para especificar este tipo de dado:

SPEC NATURAL
SORTS NATURAL
OPERATIONS: create: → NATURAL
suc: NATURAL → NATURAL
soma: NATURAL NATURAL → NATURAL
AXIOMS: soma (suc(X),Y)=suc(soma(X,Y))
END SPEC

Neste tipo, temos o conjunto subjacente criado (NATURAL), três operações (create, suc e +). Adicionalmente, temos uma única regra válida (axioma) envolvendo as funções de soma e sucessor.

- Observe que esta definição de tipo é **independente da linguagem de programação**, sendo chamada de **TIPO ABSTRATO DE DADO (TAD)**.

EXERCÍCIO TUTORIADO

Mostre que o tipo abstrato de dados NATURAL está formalmente correto.

- A **implementação** de um determinando **TAD** em uma **linguagem de programação** é chamada **realização de um TAD** ou **estrutura de dado**. Ou seja, uma **estrutura de dado depende uma linguagem de programação para existir.**

- Como TADs são especificados em termos de conjuntos (SORTS) e operações (OPERATIONS), linguagens orientadas a objetos como C++, Java, Python, C#, Smalltalk, dentre outras, são viáveis para implementação das estruturas de dados subjacentes aos tipos, pois **encapsulam** numa mesma estrutura os **dados (atributos)** e os **métodos** que manipulam estes dados.

- Vamos implementar, em C++, uma estrutura de dados para o TAD NATURAL:

SPEC NATURAL
SORTS NATURAL
OPERATIONS: create: → NATURAL
suc: NATURAL → NATURAL
soma: NATURAL NATURAL → NATURAL
AXIOMS: soma (suc(X),Y)=suc(soma(X,Y))
END SPEC
	class Natural{
	
	private:
	unsigned int value;
	unsigned int getValue();
	
	public:
	Natural(int v);
	~Natural();
	Natural suc();
	Natural soma(Natural n);
	};

E a implementação efetiva das operações ocorre abaixo:

```
Natural::Natural(int v){
   value=v;
}

Natural::~Natural(){}

unsigned int Natural::getValue(){
     return value;
}

Natural Natural::suc(){
   Natural n(value+1);
   return n;
}

Natural Natural::soma(Natural n){
   Natural s(value+n.getValue());
   return s;
}
```

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Compile e teste o programa C++ construído anteriormente.

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Como você poderia garantir que a sua implemente está correta ? Sugestão: teste o axioma do TAD Natural.

ATIVIDADE DE LABORATÓRIO

- Construa um TAD INTEIRO para representar o conjunto dos números inteiros

$$\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, 3, 4, \dots\}$$

  Identificando o conjunto do tipo, suas operações (soma e subtração) e axiomas.

- Implemente o TAD anterior na linguagem C++ e teste se a implementação está correta, utilizando os axiomas construídos na questão anterior.

EXERCÍCIOS EXTRA-CLASSE (ENTREGA NO MOODLE)

1. Aumente TAD NATURAL para suportar a operação de multiplicação e faça a implementação correspondente em C++.

2. Aumente TAD INTEIRO para suportar a operação de multiplicação e faça a implementação correspondente em C++.