

---

# Introdução a Orientação a Objetos<sup>12</sup>

## Parte I: Encapsulamento – String

---

A orientação a objetos (OO) é um paradigma de programação que se baseia em 03 (três) pilares. Neste roteiro, você estudará de forma prática sobre o primeiro pilar – o encapsulamento. Para isto, utilizaremos a biblioteca padrão do C++, a STL (*Standand Template Library*). Especificamente, neste roteiro utilizaremos a classe `string`, através de exemplos, exercícios de implementação e da referência fornecida pelo site `cplusplus.com`.

## 1 Introdução conceitual

O encapsulamento é a propriedade oferecida por classes para reunir, em uma só estrutura, atributos (dados) e métodos (procedimentos). Além disso, o encapsulamento permite configurar a visibilidade (permissão de acesso) dos diferentes atributos e métodos de uma classe. Em termos de conceitos de engenharia de software, o encapsulamento aprimora dois aspectos fundamentais:

**Coesão** é um conceito da engenharia de software que mede a dependência entre os diferentes componentes de uma aplicação. Mais precisamente, um sistema onde diferentes componentes co-relacionados estejam espalhados em diversos módulos ou dependam de estruturas externas (variáveis globais ou parâmetros) é considerado de elevada dependência. Para diminuir o nível de dependência em uma aplicação, tenta-se aumentar ao máximo a coesão dos módulos, reunindo procedimentos co-relacionados em um mesmo módulo e reduzindo a dependência de variáveis globais.

A orientação a objetos permite aumentar consideravelmente a coesão em uma aplicação ao juntar variáveis e procedimentos em uma só entidade (uma classe). Assim, todas as propriedades (*atributos*) e operações (*métodos*) relacionados a uma classe se mantêm coesos, implementados não só em um mesmo módulo, mas em uma só entidade.

**Visibilidade** é um conceito da engenharia de software que reflete as permissões de acesso a componentes de uma aplicação. Uma variável global, por exemplo, pode ser utilizada (e/ou modificada) por qualquer procedimento da aplicação. Da mesma forma, as variáveis dentro de uma `struct C` podem ser utilizadas por quaisquer procedimentos. No contexto de reusabilidade de código (disponibilização de APIs e bibliotecas em geral), isto se torna um obstáculo à segurança dos sistemas, uma vez que informações confidenciais poderiam se tornar públicas ou serem alteradas por terceiros.

Linguagens que seguem o paradigma da orientação a objetos permitem configurar a visibilidade de atributos e métodos de cada classe individualmente. Em C++, os labels `private` e `public` são usados para distinguir os atributos/métodos que podem ou não ser acessados por outros componentes da aplicação. Via de regra, os atributos de uma classe devem ser configurados como privados, enquanto os métodos devem ser configurados como públicos ou privados em função do seu propósito.

---

<sup>1</sup>Roteiro de estudo fornecido na disciplina de Linguagem de Programação 1 (LP1) da Universidade Federal do Rio Grande do Norte (UFRN). Disponível em <https://leobezerra.github.io/LP1-2017-1-T05>.

<sup>2</sup>Autor: Leonardo Bezerra ([leobezerra@imd.ufrn.br](mailto:leobezerra@imd.ufrn.br)).

## 2 Prática STL: string

A classe `string` pode ser utilizada com a inclusão da header `<string>`. Em códigos C++, objetos `string` substituem os arrays de caracteres usados no C (referidos na literatura C++ como *strings C*). Para aprender a utilizar a classe `string`, acesse a referência disponibilizada no GitHub Pages da disciplina e faça os exercícios sugeridos abaixo. Note que você deverá criar um novo projeto no GitLab para estes exercícios (lab06).

### 2.1 Exercício 1: atribuição, capacidade e acesso

Reescreva o procedimento `palindromo` usando a classe `string`. Não se preocupe com o código da aplicação por enquanto – apenas com a implementação do procedimento e do teste de unidade correspondente.

**Seções úteis** – dentro de Member functions (métodos), consulte as seções Capacity e Element access.

### 2.2 Exercício 2: modificadores e sobrecargas

Reescreva a aplicação do exercício *lab01* para utilizar a nova versão do procedimento `palindromo`. Mais precisamente, o usuário digitará até 10 palavras para formar uma frase palíndromo. Para verificar se uma frase é um palíndromo, desconsidere espaços entre as palavras. No entanto, caso a frase formada seja um palíndromo, esta deverá ser impressa com espaçamentos simples separando cada palavra digitada pelo usuário.

**Seções úteis** – dentro de Member functions (métodos), consulte a seção Modifiers. Consulte também a seção Non-member function overloads.

**Aceitação** – os testes de aceitação para este exercício se encontram no arquivo `aceitacao-frase.zip`.

### 2.3 Exercício 3: operações e constantes

Amplie a aplicação implementada no exercício acima. Especificamente, além de testar se a frase é palíndroma, você deverá considerar a seguinte funcionalidade:

- Palavra contida – se uma palavra estiver contida em outra, apenas a palavra que a contém deverá ser impressa. No entanto, a parte que corresponde à palavra contida deverá ser impressa em letras maiúsculas.

**Seções úteis** – dentro de Member functions (métodos), consulte a seção String operations. Consulte também a seção Member constants.

**Aceitação** – os testes de aceitação para este exercício se encontram no arquivo `aceitacao-ampliada.zip`.