Introdução a Orientação a Objetos[1][2]

Parte III: Herança – I/O

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 terceiro pilar – a herança. Para isto, utilizaremos a biblioteca padrão do `C++`, a `STL` (*Standand Template Library*). Especificamente, neste roteiro utilizaremos a classe `vector`, através de exemplos, exercícios de implementação e da referência fornecida pelo site `cplusplus.com`.

1 Introdução conceitual

A herança é uma propriedade de linguagens de programação que permite que o código de uma classe seja compartilhado com outra. Em `C++`, é possível ter diferentes tipos de herança, categorizadas de acordo com a cardinalidade e a visibilidade:

Cardinalidade – o tipo mais trivial de herança é a herança simples, onde uma classe derivada herda os atributos e métodos de uma única classe base. No entanto, atributos e métodos privados não são compartilhados com a classe derivada. Diferentemente de outras linguagens, `C++` permite também o uso de herança múltipla, quando uma classe derivada herda atributos e métodos de múltiplas classes base. Este último caso deve ser usado com cautela, uma vez que pode gerar conflitos de escopo e problemas de ciclo.

Visibilidade – assim como mencionado anteriormente, atributos e métodos privados de uma classe base não são compartilhados com a classe derivada. No que concerne aos demais atributos e métodos, `C++` permite distinguir entre a visibilidade protegida (`protected`) e pública (`public`). Ainda que tanto atributos e métodos protegidos como públicos sejam compartilhados com a classe derivada, apenas os atributos públicos são visíveis para o mundo externo à classe. Além disso, é possível reclassificar a visibilidade dos atributos e métodos herdados na classe derivada, utilizando pra isso a noção de visibilidade de herança:

1. *Herança privada*, onde todos os atributos e métodos herdados se tornam privados;
2. *Herança protegida*, onde todos os atributos e métodos herdados se tornam protegidos;
3. *Herança pública*, onde a visibilidade original dos atributos e métodos herdados é mantida.

2 Prática STL: I/O

As classes de entrada/saída (input/output, ou I/O) do `C++` formam uma estrutura ampla e flexível, baseada predominantemente em herança. Para começar a compreender estas classes, neste roteiro você irá estudar mais a fundo os objetos de entrada e saída padrão do `C++`. Para isto, acesse a referência disponibilizada no GitHub Pages da disciplina e faça o exercício sugeridos abaixo. Note que você deverá criar um novo projeto no GitLab para este exercício (`lab04`).

[1] 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/LP1-2017-2-T04`.

[2] Autor: Leonardo Bezerra (`leobezerra@imd.ufrn.br`).

(*lab04*) Matrizes – no roteiro anterior, você implementou uma aplicação para trabalhar com matrizes. Neste roteiro, você deverá criar uma classe Matriz que forneça as principais operações básicas que uma matriz deve disponibilizar:

- *inicialização*, recebendo como parâmetros a capacidade máxima de linhas e colunas da matriz e uma *flag* indicando se a matriz deve ser inicializada ou apenas alocada.
- *cópia e atribuição*, que permitem que uma matriz seja copiada de outra.
- *adição, subtração e multiplicação*, usando os operadores aritméticos padrões do `C++` (`operator+`, `operator-` e `operator*`).
- *entrada e saída formatada* usando os objetos padrões de I/O do `C++` (`cin`, `cout` e `cerr`). Para isso, você necessitará fazer sobrecarga nos operadores de inserção e extração de fluxo (`<<` e `>>`), como exemplificado no código disponível neste link: `http://cpp.sh/3er5h`.