Comparação entre implementações Orientada a Objetos e Estruturada de uma Biblioteca em C++

Gabriel Eduardo Lima[1], **Hylson Vescovi Netto**[1]

[1]Instituto Federal Catarinense (IFC) - Campus Blumenau - Blumenau, SC - Brasil

limaedugabriel@gmail.com, hylson.vescovi@ifc.edu.br

1. Introdução

O presente trabalho tem como propósito realizar uma comparação entre bibliotecas implementadas nos paradigmas estruturado e orientado a objetos. Uma vez observado o crescente interesse no uso de elementos reusáveis pela indústria de software, compreender as principais características desses recursos faz-se importante.

Pretendendo uma abordagem simples, após a apresentação de conceitos gerais sobre o reuso de software e dos paradigmas supracitados, alguns trechos de códigos desenvolvidos [1] em C++ fundamentam a discussão do tema principal do artigo.

2. Reusabilidade, Bibliotecas e Paradigmas

A reusabilidade é um recurso que além de facilitar a redução de custos no desenvolvimento de sistemas, aumenta a produtividade e qualidade do produto [Ferreira, 2011]. Bibliotecas são um bom exemplo de aplicação do reúso. A junção de códigos relacionados em um único módulo permite que diferentes situações sejam abordadas a partir de uma origem comum.

Todavia deve-se ater ao fato que códigos podem ser escritos usando diferentes paradigmas. Cada modelo possui suas próprias características que serão induzidas ao componente reusável. Sendo assim, bibliotecas de mesmo proposito implementadas diferentemente podem acarretar em características distintas na hora da aplicação.

Em programação estruturada observa-se a definição do programa em sequência de etapas. Funções e procedimentos são os componentes fundamentais para o reúso, uma vez que permitem um mesmo processamento seja realizado mais de uma vez com apenas uma única definição.

Já na programação Orientada a Objetos um programa é definido através da generalização, abstração e o relacionamento dos componentes criados. Além das classes e objetos, herança, polimorfismo e composição são outros elementos aplicáveis no contexto de reusabilidade em bibliotecas.

3. Definição do Caso de Estudo

A discussão do artigo baseia-se na implementação de uma biblioteca para o cálculo e representação numérica do movimento uniforme de um corpo físico em um espaço bidimensional.

[1]Códigos disponíveis em https://github.com/Lima001/BCC-POO-II/tree/main/Bibliotecas

Os requisitos e funcionalidades básicas da biblioteca são: (I) Representar Grandezas vetoriais; (II) Dadas a posição, velocidade e tempo decorrido, calcular a posição de um Corpo no plano; (III) Verificar se dois Corpos de formato circular colidiram;

A biblioteca está implementada de forma estruturada e orientada a objetos (OO), onde um código de testes foi produzido para que pudessem ser utilizadas. Por meio desses artefatos é possível analisar e comparar os dois tipo de implementação do módulo.

4. Comparação entre as Implementações

Inicialmente é possível observar uma diferença no que tange a quantidade de código em cada biblioteca. Enquanto na versão estruturada poucos comandos são necessários para definir um vetor, a versão OO necessita de mais detalhes para o mesmo propósito (Figura 1).

Figura 1. Implementação de um vetor pelas Bibliotecas

```
// Biblioteca Estruturada
float* criar_vetor(float xf, float yf, float xi=0, float yi=0){
    float* vetor = new float[2];
    *(vetor) = xf-xi;
    *(vetor+1) = yf-yi;

    return vetor;
}

// Biblioteca Orientada a Objetos
class Vetor {

    public:

        float x;
        float y;

        Vetor():
            x(0), y(0){
        }

        Vetor(float xf, float yf, float xi=0, float yi=0):
            x(xf-xi), y(yf-yi){
        }
```

Esse aspecto está relacionado a estrutura básica de um programa OO. Diferentemente do programa estruturado, uma classe foi criada. Essa estrutura necessita da definição de atributos e métodos construtores para operar, sendo consequentemente maior do que uma simples função. Todavia isso não implica necessariamente na complexidade de manipulação no programa.

O vetor estruturado é representado através de um *array*, algo que pode se tornar muito mais complexo de gerenciar ao longo do programa e utilizar em conjunto de outros elementos de código. Já o vetor OO é facilmente manipulado por um objeto, seus atributos e métodos (Figura 2).

Para finalizar, outra diferença notável tange a organização dos dados de um programa. Considere a necessidade da biblioteca em representar um corpo físico, elemento que possui velocidade, posição massa e um raio de circunferência. Em OO pode-se facilmente definir uma classe que implementa esses dados como atributos e utiliza métodos para manipulá-los.

Porém na implementação estruturada, uma função não consegue agrupar e representar os dados como uma entidade só. Nesse caso é necessário utilizar alguma estrutura de dados da linguagem (como os *arrays* para representar vetores) e agrupar logicamente variáveis que representam um mesmo elemento.

Enquanto na biblioteca OO os dados são facilmente agrupados para representar um elemento através de atributos e composição de objetos, na biblioteca estruturada é necessário que o programador crie uma logica própria de agrupamento – usar nomes padronizados, escrita agrupado etc. (Figura 3).

Figura 2. Manipulação de dados usando recursos das Bibliotecas

```
// Biblioteca Estruturada
// Array com os dados vetoriais de cada corpo - (Posição, Velocidade)
float* lista_vetores_corpos[2][3] = {
    {vetor_posicao_1, vetor_velocidade_1},
    {vetor_posicao_2, vetor_velocidade_2},
};

// Exibindo dados dos Corpos criados com a função exibir_vetor()
for (int i=0; i<2; i++){
    std::cout << "Objeto " << i << " - ";
    exibir_vetor(lista_vetores_corpos[i][0]);
    std::cout << " / ";
    exibir_vetor(lista_vetores_corpos[i][1]);
    std::cout << std::endl;
}

// Biblioteca Orientada a Objetos
// Array com objetos da classe Corpo
Corpo array_corpos[2] = {c1, c2};

// Exibindo os objetos usando sobrescrita do operador <<
for (int i=0; i<2; i++){
    std::cout << "Objeto: " << i << std::endl;
    std::cout << array_corpos[i] << std::endl;
}
```

Figura 3. Agrupamento de Dados de um programa que usa as Bibliotecas

```
// Biblioteca Estruturada - Dados do Corpo 1
float* vetor_velocidade_1 = criar_vetor(10, 0);
float* vetor_posicao_1 = criar_vetor(-60, 0);
float raio_1 = 2;
float massa_1 = 50;

// Biblioteca Orientada a Objetos - Dados do Corpo 1
Corpo c1 = Corpo(Vetor(-60,0), Vetor(10,0), 50, 2);
```

5. Conclusão

Nota-se que no contexto desse artigo a implementação OO é mais fácil de ser aplicada em um programa. Já a implementação estruturada possui uma maior dificuldade para abstrair elementos do problema.

Entretanto, não é correto afirmar que a biblioteca estruturada é pior. Em casos onde deseja-se reutilizar funcionalidade que independam de uma estrutura bem definida de dados, ela pode se sair melhor.

Imagine uma biblioteca de funções matemáticas (media, seno, raiz, etc.). Nesse caso é muito mais prático usar uma função do que definir uma classe. Deve-se possuir um conhecimento do contexto na qual os componentes serão utilizados e qual tipo de recurso o desenvolvedor deseja priorizar.

Por fim ressalta-se que visando simplicidade, muitos aspectos dos paradigmas abordados não foram trabalhados. Dessa forma é certo que mais diferenças podem ser trabalhadas em detalhe para futuros trabalhos.

Referências

Ferreira, H. N. M. and Naves, T. F. (2011). Reuso de sofwtare: Suas vantagens, técnicas e praticas. *IX EnAComp*, 4.

Sebesta, R. W. (2011). *Conceitos de Linguagens de Programação*. Bookman, 9ª edition.

Wegner, P. (1990). Concepts and paradigms of object-oriented programming. *ACM Sigplan Oops Messenger*, 1.