# Bibliotecas procedurais e orientadas a objetos

**Álvaro Alvin Oesterreich Santos**

Instituto Federal Catarinense – campus Blumenau (IFC - Blumenau)
Blumenau – SC – Brasil

alvaro.alvinn@gmail.com

***Abstract.*** *This paper briefly talks about how the procedural and object-oriented computations paradigms works, also about how are its libraries and what should be take into consideration when using each paradigm to create libraries and softwares.*

***Resumo.*** *Este artigo fala de forma breve como funcionam os paradigmas de programação procedural e orientado a objetos, também como são suas bibliotecas e o que deve ser levado em consideração ao utilizar cada paradigma para a criação de bibliotecas e softwares.*

## 1. Introdução

Paradigmas de programação são a maneira como uma linguagem funciona, a forma como os programas são escritos e consequentemente como se comportam. Uma parte muito importante no processo de desenvolvimento de *softwares* é o uso de bibliotecas, pacotes de código que são de certa forma adições funcionalidades extras que incrementam e facilitam o processo de desenvolvimento. Cada linguagem pode "receber" bibliotecas especifica para si, por isso é importante observar junto ao paradigma principal da linguagem como funcionam e são desenvolvidas.

## 2. Paradigma Procedural e suas Bibliotecas

O paradigma de programação procedural é um dos mais básico e antigos, um código escrito com o paradigma procedural é lido de cima para baixo para resolver um problema em questão. A parte principal de uma linguagem do paradigma procedural é são as funções, que originam seu nome "Procedural" é derivado de procedimento, e esse paradigma consiste na elaboração de diversos procedimentos que são projetados para resolver problemas.

Esses procedimentos podem ser escalados para juntos resolverem problemas maiores e maiores. No código, cada um desses procedimentos é chamado de função e da mesma forma que uma função funciona na matemática também funciona na computação, em uma função $f$ cujo a variável é $x$ dependendo do valor de x teremos um $f(x)$ correspondente ou, também podemos chamar de $y$. Na computação procedural as funções recebem argumentos e como na matemática, para cada conjunto de argumentos elas retorna uma resposta.

Como a base do paradigma procedural são as funções, suas bibliotecas são constituídas por nada menos que conjuntos de funções. Uma particularidade das bibliotecas procedurais é que costumam ser grandes quantidades de funções independentes, não relacionadas a nenhum outro *software*, e por isso fica por conta de do utilizador da biblioteca adaptá-las para seu uso. Wegner (1989)

Funções não possuem memória, o que limita o que pode ser feito com elas, já que dependem exclusivamente dos argumentos que recebem. Por isso muitas vezes acabam utilizando dados que estão fora do seu controle o que pode levar a complicações no desenvolvimento e trazer mais risco em especial com a utilização de ponteiros e variáveis globais que exigem uma atenção extra do utilizador da biblioteca.

## 3. Paradigma Orientado a Objetos e suas Bibliotecas

O paradigma de programação orientada a objetos é relativamente recente surgindo por volta de 1960, e como seu nome diz sua linguagens desse paradigma tem como principal elemento os objetos, em Python por exemplo, uma linguagem orientada a objetos, literalmente todo o elemento da linguagem é um objeto. Esses objetos são instancias de classes, um exemplo prático para entender esse relacionamento é de que a classe é como se fosse uma forma, e o objeto é o bolo, a forma não é manipulável ou comestível mas ela é responsável pela forma do bolo .

Os principais elementos das classes são as variáveis e os métodos, os métodos são como funções da própria classe, e possuem acesso a todas as variáveis da classe que são a "memória" que as funções no paradigma procedural não possuem. A orientação a objetos abriu um grande leque de oportunidades, pois graças a ela se tornou muito mais fácil a criação e manutenção de grandes projetos e principalmente a reutilização de código já que cada classe pode ser vista como um módulo que tem grande capacidade de manipular a si mesmo e pode também tem bom relação com outras classes e costumam depois de testadas serem estáveis.

A melhor forma de aproveitar a reusabilidade de código que a orientação a objetos proporciona é por meio de bibliotecas, que basicamente são conjuntos de classes. Bibliotecas orientadas a objetos costumam ser muito interligadas, reutilizando código até de si mesmo de certa forma.

Existem vários "tipos" de bibliotecas com propósitos de reusabilidade diferentes. Wegner (1989)

**Reusabilidade entre aplicações**: Bibliotecas utilizadas em sistemas operacionais por exemplo, onde o sistema precisa funcionar com diversas aplicações diferentes, essas bibliotecas costumam ser muito grandes e complexas e por isso não são interessantes para projetos pequenos.

**Reusabilidade de desenvolvimento**: Bibliotecas com propósito mais especifico, que são escolhidas como base para a aplicações e costumam evoluir com a mesma, parte a parte, por acabarem pro final ficando muito especificas não são boas opções para protos mais genéricos.

**Reusabilidade de programa**: Bibliotecas que costumam uma facilidade maior de utilização e que são utilizadas em um mesmo programa para cumprir tarefas diversas e genéricas, por isso não são ideais para programas mais específicos.

**Reusabilidade de código**: Bibliotecas que mal podem ser chamadas de bibliotecas, costumam ser são pedaços de códigos feitos especificamente para algum programa pequeno e possuem pouca reusabilidade fora do mesmo.

A criação de uma biblioteca orientada a objetos precisa levar com consideração diversos pontos como, o que estará na biblioteca, o que ou quem a utilizará, como será as estruturas da biblioteca e a relação entre seus módulos. Bibliotecas para que sejam uteis precisam se encaixar onde serão utilizadas, e isso nem sempre acontece e esse é um ponto negativo, muitas vezes é difícil fazer uma biblioteca funcionar em conjunto com outras já existentes no projeto por terem estruturas e funcionamento muito diferentes.

## 4. Principais Diferenças

Como visto nos tópicos 2 e 3 as bibliotecas de paradigmas diferentes são compostas por diferentes estrutura. Mas além disso, existem grandes diferenças na hora de construir essas bibliotecas, e mesmo *softwares* em cada um desses paradigmas.

Na orientação a objetos, um dos principais focos do *softwares* ou da biblioteca que é desenvolvida é a sua reusabilidade, então é necessário fazer um planejamento de como cada classe ira funcionar, como ela se relacionará com outras classes do mesmo projeto

e até mesmo com outras classes genéricas de outras bibliotecas ou *softwares*. Por isso se pensa primeiramente em quais classes o *softwares* ira utilizar antes de sair escrevendo códigos.

Já no desenvolvimento de *softwares* e bibliotecas procedurais também é necessário planejamento, mas não tanto já que o intuito principal não é a reutilização de código. Entretanto é preciso pensar mais profundamente em como o *software* ira funcionar e as consequências disso nas funções que serão criadas. Já que não é possível ter um nível de modulação como acontece na orientação a objetos, com projetos que utilizam diversos módulos com classes mais especializadas.

Por isso hoje o paradigma orientado a objetos é muito mais utilizado, devido a possibilidade de reutilização de código e a modularização de sistemas. Mas para que essas vantagens existam é preciso haver muito planejamento para que um tempo economizado no inicio não resulte em um tempo maior gasto posteriormente.

## Referencias


Wegner, P. (1989) “Consepts and Paradigms of Objected-Oriented Programming em Brown University, Expansion of Oct 4 OOPSLA-89 Keynote Talk, paginas 13 – 18.

Eliason, K. (2013) “Difference Between Object-Oriented Programming and Procedural Programming Languages”, https://neonbrand.com/websites/development/procedural-programming-vs-object-oriented-programming-a-review, acesso em 22/06/2021.