4 Introdução aos paradigmas das linguagens de programação

Da mesma forma que o hardware proposto para o computador sofreu diversas mudanças ao longo dos anos, também a forma de programar sofreu alterações, de acordo com a visão de cada linguagem projetada, criando assim vários paradigmas da programação.

O termo paradigma significa modelo, isto é, a definição de um padrão a ser seguido. Esse conceito, que nasceu em 1900 e relacionava-se inicialmente somente à linguagem e comunicação humana, passou mais tarde a ser adotado também na Computação.

Um paradigma de programação, assim, determina forma como o programador deve estruturar as instruções em uma dada linguagem a fim de gerar um programa capaz de executar uma determinada tarefa.

Dois são os principais paradigmas da programação atualmente, a programação imperativa e a programação declarativa, cada qual sendo subdividida em novos paradigmas, como veremos a seguir.

4.1 Programação imperativa

No paradigma imperativo, há uma grande preocupação quanto ao detalhamento do funcionamento do programa.

O programa é visto como um conjunto de rotinas e subrotinas (daí ser também conhecida como procedural ou procedimental) e enfatizam-se as mudanças de estado do programa (registrada por meio de suas variáveis globais e locais).

Nesta forma de programação, há um foco maior em “como um programa deve executar uma tarefa”, em vez de focar em “o que o programa deve fazer”.

Neste paradigma, podemos encontrar alguns dos subparadigmas mais empregados atualmente: a programação estruturada, a programação orientada a objetos e a programação concorrente.

4.1.1 Programação estruturada

Na programação estruturada, o programador deve descrever todo o seu programa por meio de instruções, estruturas de dados e subrotinas e qualquer programa pode ser reduzido a três tipos de estruturas: sequência, decisão e interação.

Algumas das linguagens de programação consideradas estruturadas são: Pascal, C, COBOL e Fortran.

Vantagens: geralmente são linguagens mais fáceis de aprender, uma vez que envolve um menor número de conceitos distintos do que aqueles apresentados por outros paradigmas, como a programação orientada a objetos; permitem a modularização, isto é, a criação de módulos independentes lógica e fisicamente, facilitando assim o desenvolvimento e manutenção de sistemas; e são úteis na resolução de problemas mais simples.

Desvantagens: apresentam maiores dificuldades na representação de soluções para certos "problemas do mundo real", que geralmente são mais complexos; além disso, apresentam baixa manutenibilidade quando empregadas em problemas mais complexos, geralmente problemas onde conceitos como herança e polimorfismo poderiam ajudar a abstrair certas partes dos mesmos.

4.1.2 Programação orientada a objetos

Na programação orientada a objetos, compreende-se a necessidade de compreender o problema por meio da relação existente entre as diversas entidades que participam do mesmo, sendo assim, a solução para o problema deve ser implementada levando-se em consideração os objetos e suas interações que sejam pertinentes ao universo do problema.

Alguns exemplos de linguagens orientadas a objetos são C++, Object Pascal e Java.

Vantagens: graças aos novos conceitos introduzidos por meio da orientação a objetos, conseguiu-se uma melhor organização do código, geralmente separado em classes e bibliotecas; outra vantagem é um melhor reaproveitamento do código, por meio de conceitos como herança.

Desvantagem: a programação orientada a objetos apresenta um maior número de conceitos e abstrações necessárias, quando comparada à programação estruturada, o que pode tornar seu aprendizado um pouco mais complexo.

4.1.3 Programação concorrente

Com o desenvolvimento da tecnologia, surgiu um novo conceito: o paralelismo, isto é, a possibilidade de executar-se mais de uma tarefa paralelamente, seja por meio de vários núcleos processadores, seja por meio de um só, executando-se assim um paralelismo emulado.

A programação concorrente visa estudar e propor como melhor desenvolver programas visando esse cenário, onde processos executam simultaneamente e concorrem por recursos, como a memória e o próprio processamento.

Várias linguagens oferecem suporte à programação concorrente, como Java, C#, C++, C e Object Pascal. Como se pode perceber, uma mesma linguagem pode oferecer suporte a mais de um subparadigma.

Vantagens: aumento do desempenho, na medida em que melhor se utiliza de todos os recursos computacionais disponíveis; possibilidade de melhor modelagem dos programas, pois determinados problemas computacionais são concorrentes por natureza.

Desvantagens: programação mais complexa, com possíveis erros decorrentes da própria concorrência; difícil depuração devido ao aspecto não-determinístico que a solução pode tomar.

4.2 Programação declarativa

A programação declarativa trata-se de uma abordagem focada em descrever o que o programa deve fazer e menos em como seus procedimentos devem funcionar. Sendo assim, apresenta-se como um paradigma oposto ao paradigma imperativo, uma vez que se preocupa com “o que deve fazer”, em vez de preocupar-se com “como deve ser feito”.

Na programação declarativa, não há a ideia de estado do programa, como ocorre na programação imperativa.

Dentro do paradigma declarativo, podemos identificar a programação funcional, a programação lógica e a programação restritiva.

4.2.1 Programação funcional

A programação funcional busca descrever o programa em torno de funções.

O código-fonte de um programa escrito em linguagem funcional pode ser interpretado como um conjunto de funções que recebem zero ou mais argumentos como entrada e retornam sempre um valor em sua saída. Cada função, então, deve realizar um conjunto de operações sobre os parâmetros de entrada a fim de obter a saída.

Algumas linguagens que se utilizam dessa abordagem são APL, Lisp, ML, Haskell, OCaml e F#.

Vantagem: há uma maior transparência referencial, uma vez que todo o programa é composto por funções, tornando-se mais fácil efetuar uma inspeção do código.

Desvantagem: não oferece alocação explícita de memória ou declaração explícita de variáveis, necessários para a resolução de muitos problemas reais.

4.2.2 Programação lógica

Esta é a abordagem da programação que faz forte uso da lógica matemática para a resolução de problemas, geralmente por meio de valores ou objetos e relações entre os mesmos, formulando-se assim proposições e regras.

Duas linguagens de programação foram fundamentais para este paradigma, criando-se a partir delas duas famílias de linguagens lógicas: Planner e Prolog.

Dentre as linguagens de programação derivadas de Planner, podemos destacar QA-4, Popler, Conniver e QLisp.

Já dentre as linguagens de programação derivadas de Prolog, podemos apontar Mercury, Visual Prolog, Oz e Frill.

Vantagem: muito úteis em problemas que possam ser resolvidos por meio de dedução ou inferência.

Desvantagem: pode haver explosão combinatorial da base de conhecimento levando a grande desperdício de recursos computacionais.

4.2.3 Programação restritiva

A programação restritiva nasceu da programação lógica (motivo pelo qual muitos autores apontam-na como parte da programação lógica). Ela se utiliza de restrições na definição de relações entre as variáveis. Em inglês, denomina-se Constraint Programming.

Seu uso pode ser interessante em problemas cuja solução possa, de certa forma, ser restrita a um número de valores. Tal tipo de problema é conhecido como CSP (Constraint Satisfaction Problem), no qual são dados:

- Um conjunto finito de variáveis;
- Uma função que mapeia cada variável a um domínio finito;
- Um conjunto finito de restrições.

4.3 Linguagens de programação multiparadigma

Em meio a todos esses paradigmas, nasceram algumas linguagens com o desafio de abranger paradigmas bem diferentes em sua constituição, são as linguagens de programação multiparadigma.

Na verdade, muitas linguagens permitem mais de um subparadigma dentro de um mesmo paradigma, por exemplo, a linguagem C++ permite tanto a programação estruturada, quanto à orientada a objetos e a programação concorrente. Entretanto, há um número bem mais reduzido de linguagens que permitem paradigmas realmente distintos, como a linguagem Oz.

A principal vantagem de tais linguagens é que o programador pode utilizar o paradigma que preferir em sua implementação. Entretanto, tem como desvantagem o fato de que algoritmos que utilizem mais de um paradigma em sua implementação podem ter sua compreensibilidade afetada.