Paradigmas de Programação

Paradigmas

Introdução

Este material contém notas de aula sobre *Paradigmas*. O objetivo é apresentar conceitos e classificações para que esta base permita o avanço no estudo de alguns paradigmas de programação.

Exercícios de Revisão

Antes de iniciar, teste seus conhecimentos resolvendo as questões a seguir. Elas têm relação com os assuntos já abordados na disciplina.

1) (POSCOMP 2019) Um procedimento recursivo é aquele que contém em sua descrição:
    a) Uma prova de indução matemática.
    b) Duas ou mais chamadas a procedimentos externos.
    c) Uma ou mais chamadas a si mesmo.
    d) Somente chamadas externas.
    e) Uma ou mais chamadas a procedimentos internos.

2) Códigos de programas podem ser compilados ou interpretados. Referente à compilação e à interpretação de código, assinale a alternativa correta.
    a) Desenvolvedores de linguagens de alto nível utilizam somente linguagens compiladas.
    b) A interpretação do código se caracteriza principalmente pela geração do arquivo

executável.

c) Com uma linguagem interpretada, é necessário ter o código-fonte acessível para que o programa seja executado.
d) O interpretador converte o programa inteiro, de uma única vez, para linguagem de máquina.
e) Caso não seja desejável um código-objeto como saída, após o procedimento de tradução ou interpretação da linguagem de alto nível, uma linguagem compilada seria indispensável.

3) (POSCOMP 2019) De acordo com a Teoria de Sistema de Tipos, classifique a função a seguir:

```
int soma(int x, int y) {
 return x+y;
}
```

a) Função Somadora.
b) Função Polimórfica.
c) Função Monomórfica.
d) Função Sobrecarregada.
e) Função Abstrata.

4) (2019/COMPERVE/UFRN/Técnico de TI) Python é uma linguagem de programação de alto nível, interpretada, orientada a objetos, funcional, de tipagem dinâmica e forte. Levando isso em conta, analise o código em Python abaixo.

```
def e(b):
      a = b*b
      return a
a = 10
e(a)
e(a)
print(e(a))
```

a) 0
b) 10
c) 100
d) 1000
e) 10000

Paradigma

Um paradigma é um modelo, uma forma ou um ponto de vista. É a classificação dada às linguagens de programação com base em recursos e na forma com que eles são oferecidos. Também pode ser visto como uma forma de pensar ou um estilo de programar (LIERET, S.d.).
O conceito de paradigma é amplamente usado (e também criticado).

Uma linguagem pode pertencer a vários paradigmas. A linguagem que pertence a vários paradigmas é dita *multiparadigma*. Python, Scala e Swift são exemplos de linguagens que suportam mais de um paradigma.

Em geral, costuma-se dizer que existem dois grandes grupos de paradigmas: **declarativo** e **imperativo**. Uma forma de classificar os paradigmas está apresentada na Figura 1. Nela, linguagens de programação são inicialmente classificadas nos grupos citados. Dentro do paradigma imperativo, a figura coloca as linguagens de programação estruturadas, procedurais, orientadas a objetos, genéricas e concorrentes. Dentro do paradigma declarativo, a figura de Kowalczyk coloca as linguagens lógicas, funcionais e SQL, XML. Embora a figura seja interessante e válida, XML não se enquadra em linguagem de programação (é uma linguagem de marcação).

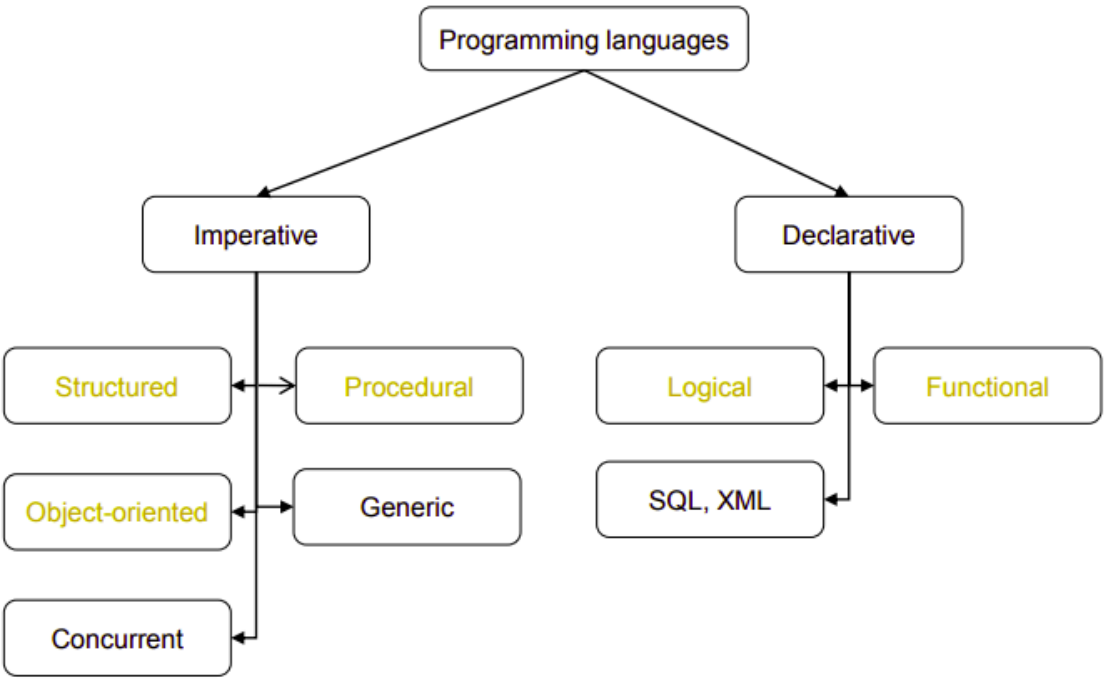

Figura 1. Classificação de linguagens de programação. Fonte: KOWALCZYK, S.d.

A Figura 2 traz outra forma de classificar os paradigmas de programação. Nela há também a divisão principal em paradigmas imperativo e declarativo; depois, entre os imperativos, coloca os paradigmas procedural, orientado a objetos e a abordagem de processamento paralelo. Já entre os declarativos, estão os paradigmas lógico, funcional e a abordagem de processamento de banco de dados.

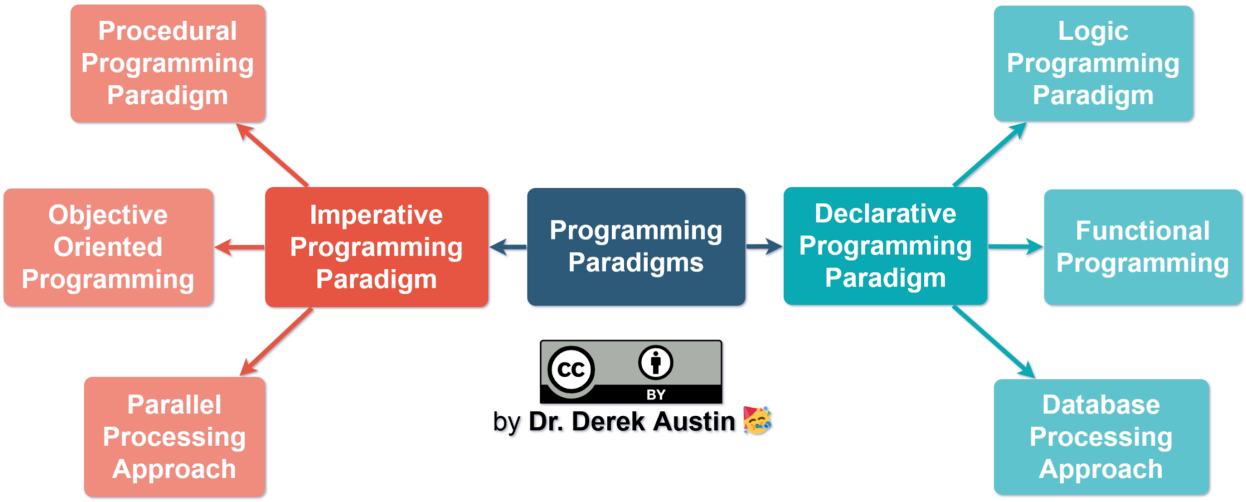

Figura 2. Paradigmas de Programação. Fonte: Austin, 2019.

Paradigma Imperativo

O termo *imperativo* é derivado do latim: *imperare* - imperar, reinar, governar, dominar, mandar, ordenar. Neste modelo, a programação é feita com comandos que mudam o estado do programa. A programação descreve o fluxo de controle, baseada na arquitetura de von Neumann.

Características da programação imperativa:

- Dados e programas são armazenados na mesma memória.
- CPU executa instruções.

O paradigma orientado a objetos é um exemplo de subcategoria do paradigma imperativo (LIERET, S.d.; SEBESTA, 2012).

Exemplos de linguagens imperativas:

- C
- C++
- C#
- Java
- Python

Linguagens imperativas se caracterizam por três construções:

- **Variável**: espaço de armazenamento nomeado que pode conter um valor. É usada para representar dados que podem variar ao longo da execução do programa
- **Atribuição**: operação que associa um valor a uma variável. Em linguagens imperativas, a atribuição é geralmente realizada usando o operador de atribuição. Quando uma variável é atribuída com um novo valor, o valor anterior (se houver) é substituído pelo novo valor.
- **Sequência**: execução de instruções em uma ordem específica. Em linguagens imperativas, essas instruções são executadas em sequência, uma após a outra, seguindo a ordem em que foram escritas no código fonte. A sequência permite controlar o fluxo de execução do programa, determinando a ordem em que as operações devem ser realizadas.

O paradigma imperativo pode ser dividido em **estruturado** e **não estruturado**.

Paradigma Estruturado

O paradigma estruturado surgiu na década de 50, com a linguagem ALGOL. Este paradigma é reduzido a três estruturas: sequência, decisão e repetição. Essas estruturas especificam os passos necessários para que o programa atinja o resultado esperado.

O paradigma estruturado representa a forma "natural" de pensar na solução de problemas.

Exemplos de linguagens estruturadas:

- ALGOL
- C
- Java
- Python

Paradigma Não Estruturado

O paradigma não estruturado possui instruções de salto (também chamadas de desvios, pulos, *gotos* ou *jumps*) em vez de estruturas de repetição e decisão. É mais difícil de manter e de

encontrar erros. Linguagens não estruturadas geralmente têm legibilidade comprometida conforme a quantidade de saltos aumenta no programa. Esse paradigma é pejorativamente conhecido como *código espaguete*.

Códigos que abusam destes desvios, mesmo em linguagens que possuem estruturas que as qualificam como estruturadas, também são denominados "espaguete".

Exemplos de linguagens não estruturadas:

- Linguagens de montagem (*assembly*)
- Versões mais antigas de linguagens, como COBOL e Fortran, e Scripts Batch.

Paradigma Procedural

No paradigma procedural, o código contém as três estruturas que definem um paradigma estruturado (quais são?) e tem como unidade básica a *função*, uma pequena porção coesa de código, permitindo assim decompor um problema maior.

Em linguagens procedurais, o código é organizável, reusável e modular. Exemplos de linguagens procedurais incluem Basic, C, Fortran, Perl, PHP e Python.

Este paradigma será abordado em detalhes na próxima aula.

Paradigma Declarativo

Em linguagens de programação declarativas, a programação não é feita definindo a ordem das operações ou o fluxo de controle, mas descrevendo a lógica (LIERET, S.d.), fornecendo e declarando as operações disponíveis e as regras às quais estas operações se sujeitam.

Este paradigma tem correspondência com a lógica matemática, devido à ênfase na descrição da lógica e das relações entre os elementos, em vez de se concentrar em detalhes de implementação e controle de fluxo, como ocorre no paradigma imperativo. Na matemática, muitas vezes os problemas e as soluções são descritos em termos de relações lógicas, equações, funções e axiomas.

Os paradigmas lógico e funcional são subcategorias do paradigma declarativo. Assim, são exemplos de linguagens declarativas:

- CLIPS
- Cloujure
- Haskell
- LISP
- Miranda
- Prolog
- SQL

Linguagens de marcação e estilização, embora não sejam de programação, também são declarativas. Como já citado, XML é uma linguagem de marcação e é uma linguagem declarativa. As folhas de estilo em cascata (*Cascading Style Sheets* – CSS) também são declarativas. O quadro abaixo exibe um exemplo de declaração CSS:

```
h1 {
      color: blue;
}
```

A regra acima declara (descreve) como deve ser a estilização de elementos h1.

Conclusão

O estudo dos paradigmas de programação fornece uma compreensão das diferentes abordagens e estilos de codificação. Foram apresentadas classificações e características dos paradigmas, classificados principalmente em dois grupos: imperativo e declarativo. O paradigma imperativo enfatiza o controle do fluxo por meio de comandos que alteram o estado do programa. O paradigma declarativo se concentra na descrição da lógica e das operações disponíveis.

A análise dos diferentes paradigmas permite classificar as linguagens de programação com base em suas características e suporte a esses paradigmas. Linguagens como CLU, Python,

Scala e Swift são exemplos de linguagens multiparadigma, pois oferecem suporte a mais de um paradigma de programação. Isso proporciona aos desenvolvedores uma flexibilidade significativa ao escolher o estilo de programação mais adequado para resolver problemas específicos.

Além disso, foram exploradas subdivisões mais detalhadas dos paradigmas, como as abordagens estruturada e não estruturada. Cada paradigma tem suas próprias características e vantagens, e a escolha do paradigma certo depende das necessidades e requisitos específicos do projeto.

Referências

AUSTIN, Derek. What Are JavaScript Programming Paradigms? Aug 26, 2019. Disponível em: https://javascript.plainenglish.io/what-are-javascript-programming-paradigms-3ef0f576dfdb. Acesso em: 26 fev. 2024.

KOWALCZYK, Robert. Programming Paradigms and Languages. Introduction. [S.d.]. Disponível em: https://slideplayer.com/slide/17199613/. Acesso em 26 fev. 2024.

LIERET, Kilian. Programming Paradigms. [S.d.]. Disponível em: https://indico.cern.ch/event/853710/contributions/3708306/attachments/1985126/3307454/programming_paradigms_280920_handout.pdf. Acesso em 26 fev. 2024.

SEBESTA, Robert W. Concepts of programming languages. Pearson. 10th ed. 2012.