SEMINÁRIO DE PESQUISA E INICIAÇÃO CIENTÍFICA – UBM
CENTRO UNIVERSITÁRIO DE BARRA MANSA – 2023

OS PRINCÍPIOS DAS LINGUAGENS DE PROGRAMAÇÃO

THE PRINCIPLES OF PROGRAMMING LANGUAGES

Vinícius Fernandes dos Santos[1]

Vinicius Viana Oliveira Soares[2]

Orientadores: Pedro Luis Millen Penedo[3] **e Ricardo Alves Said**[4]

RESUMO

Com a evolução constante das linguagens de programação, é de suma importância que os programadores compreendam os princípios de tais linguagens para que possam estar preparados para essas evoluções e se adaptem às novas necessidades do mercado. Dessa forma, este artigo tem como finalidade demonstrar a importância dos princípios das linguagens de programação para a escrita de códigos de programação e desenvolvimento de software através de pesquisas bibliográficas. Para melhor compreensão foi abordado: a evolução histórica das linguagens de programação; princípios fundamentais das linguagens de programação como sintaxe e semântica, tipos de dados sendo tipos de dados em linguagens fortemente tipados e fracamente tipadas, modularidade, biblioteca e portabilidade.

Palavras-Chave: Linguagens de programação; Desenvolvimento de software; Escrita de códigos de programação.

ABSTRACT

With the constant evolution of programming languages, it is of paramount importance that programmers understand the fundamental principles of such languages so that they can be prepared for these evolutions and adapt to new market needs. Thus, this article aims to demonstrate the importance of the principles of programming languages for writing programming codes and software development through bibliographic research. For a better understanding, the following were addressed: the historical evolution of programming languages; fundamental principles of programming languages such as syntax and semantics, data types, modularity, library and portability.

Keywords: Programming languages; Software development; Writing programming code.

1 INTRODUÇÃO

Os princípios das linguagens de programação se referem aos fundamentos básicos, teóricos e práticos que orientam sobre a utilização das linguagens de programação. São extremamente importantes pois fornecem uma base para a criação de códigos e orientam as decisões que os programadores tomam durante o processo de desenvolvimento de software. Sobre a escrita de códigos de programação, segundo MARTIN:

> “[...] Você deve adquirir o conhecimento dos princípios, padrões, práticas e heurísticas que um profissional habilidoso sabe, e também esmiuçar esse conhecimento com seus dedos, olhos e corpo por meio do trabalho árduo e da prática [...]” (2009)

Todos os princípios são indispensáveis, porém alguns possuem maior destaque e são de fácil compreensão, como por exemplo: Sintaxe, Semântica, Tipo de dados, Modularidade, Bibliotecas e Portabilidade.

Portanto este artigo tem como objetivo demonstrar através desses princípios a importância do tema escolhido para a escrita de códigos de programação e desenvolvimento de software.

Justifica-se a realização do artigo devido a crescente demanda de mercado por desenvolvedores aptos a lidar com as mais diversas linguagens de programação, bem como a necessidade de profissionais capacitados em entender os princípios que regem essas linguagens.

Para atingir o objetivo do estudo será realizado pesquisas bibliográficas através de livros, artigos e estudos de casos publicados.

Por fim, é importante destacar que as linguagens de programação evoluem constantemente, e novas linguagens surgem a todo momento. Compreender os princípios das linguagens de programação permite que o programador esteja preparado para se adaptar às mudanças e acompanhar a evolução do mercado.

[1] Vinícius Fernandes dos Santos - Sistemas de Informação -Centro Universitário de Barra Mansa (UBM), RJ. E-mail: viniciusfdossantos@yahoo.com

[2] Vinicius Viana Oliveira Soares – Sistemas de Informação – Centro Universitário de Barra Mansa (UBM), RJ. E-mail: viana.vinicius32@gmail.com

[3] Pedro Luis Millen Penedo – Sistemas de Informação – Centro Universitário de Barra Mansa (UBM), RJ. Email: pedro.penedo@ubm.br

[4] Ricardo Alves Said - Sistemas de Informação – Centro Universitário de Barra Mansa (UBM), RJ. E-mail: ricardo.said@ubm.br

2 EVOLUÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação são uma forma de instruir computadores a realizar determinadas ações, como imprimir um documento, por exemplo. No entanto, os computadores não possuem capacidade de compreender e se expressar em linguagens humanas.

Os computadores e demais dispositivos eletrônicos entendem apenas sinais elétricos, que, no caso dos processadores, representam a presença ou ausência de tensão elétrica. Essa representação é composta pelos numerais inteiros ZERO e UM, que formam o sistema binário de números. Essa linguagem é conhecida como linguagem de máquina. Segundo AHO et al:

> "[...] Os primeiros computadores eletrônicos apareceram na década de 1940 e eram programados em linguagem de máquina por sequências de 0s e 1s que diziam explicitamente ao computador quais operações deveriam ser executadas e em que ordem. As operações em si eram de muito baixo nível: mover dados de um local para outro, somar o conteúdo de dois registradores, comparar valores e assim por diante. [...]" (2008, p.8)

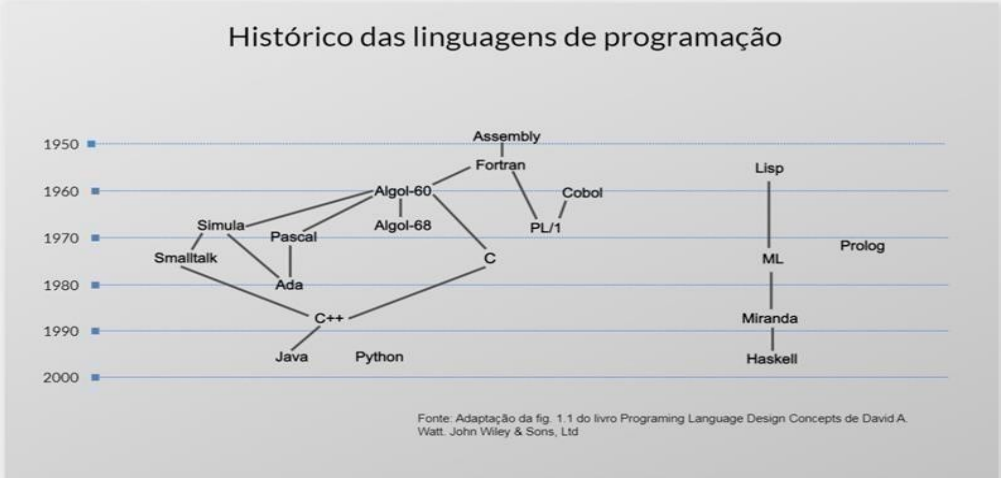
Devido o desenvolvimento de tais operações ser cansativo, passível de erro e difícil de estender ou modificar, em 1950 foi dado o primeiro passo para tornar as linguagens de programação mais compreensíveis através da linguagem simbólica, também conhecida como *Assembly*. No início utilizavam códigos mnemônicos (ADD, SUB, etc), pois são mais simples para aprender e memorizar. Logo após, foi-se acrescentado instruções de *macro* a essas linguagens a fim de que o programador definisse abreviaturas parametrizadas para sequência de instruções de máquinas usadas com frequência.

Na segunda metade da década de 1950 foi desenvolvido o Fortran para computação científica e em seguida o *Cobol* para o processamento de dados comerciais e o *Lisp* para a computação simbólica. Para AHO et al (2008) "A filosofia por trás dessas linguagens era criar construções de alto nível a partir das quais os programadores

poderiam escrever com mais facilidade cálculos numéricos, aplicações comerciais e programas simbólicos.”

Conforme podemos observar na Figura 1, com o passar dos anos e o avanço da tecnologia foram projetadas novas linguagens com diversos novos recursos para tornar a programação mais fácil, eficaz e grandiosa.

Figura 1 - Histórico das linguagens de programação



Fonte: Adaptado pelo autor fig 1.1. WATT, David A. Programming Language Design Concepts. 2023

3 PRINCÍPIOS FUNDAMENTAIS DAS LINGUAGENS DE PROGRAMAÇÃO

3.1 SINTAXE E SEMÂNTICA

A sintaxe é a forma como os comandos da linguagem são escritos e organizados, desde a estrutura e organização do código, a ordem dos comandos, a utilização correta de parênteses, colchetes e chaves, a definição de variáveis e funções, entre outros aspectos. Quando um programador escreve um código com erros de sintaxe, o compilador ou interpretador da linguagem não consegue interpretá-lo e retorna mensagens de erro.

A semântica se refere ao significado do código, ou seja, o que ele faz e como se comporta quando executado. Isso envolve questões como a lógica e a estruturação do código, a utilização correta de variáveis e funções, o tratamento de erros, entre outros aspectos. Uma linguagem de programação pode ter uma sintaxe perfeitamente correta, mas se o código escrito não tiver uma semântica adequada, ele pode produzir resultados indesejados ou não funcionar corretamente.

Segundo MELO E SILVA:

> "A semântica determina a interpretação pretendida para cada elemento sintático da linguagem. Para as linguagens de programação, em geral a semântica é caracterizada sob três aspectos[...]:
>
> 1. Semântica Axiomática: descrita tipicamente através de um conjunto de axiomas equacionais que relacionam diferentes expressões sintáticas na linguagem. Assim da mesma forma como ocorre com um dicionário para termos em português, o significado de uma expressão é determinado através de "sinônimos".
> 2. Semântica Operacional: descrita pelas operações efetuadas pelas expressões e seus resultados. A semântica operacional caracteriza o significado de cada expressão em um programa pelo que aquela expressão faz. O significado de um programa é o comportamento deste quando executado em uma máquina; e
> 3. Semântica Denotacional: descrita pelos conjuntos de dados associados a cada expressão. Considerando o programa como uma máquina de transformação de dados, o significado de cada expressão pode ser caracterizado em termos dos dados que ela transforma. O significado de um programa pode ser visto como uma função matemática que mapeia as entradas do programa para suas respectivas saídas." (2014, p.9)

Em um caso em que o código possui uma sintaxe correta, mas com uma semântica inadequada, pode fazer com que um programa que deveria realizar uma tarefa específica produza resultados diferentes do esperado. Por outro lado, um código com uma semântica adequada, mas com erros de sintaxe não funcionará. Portanto, ambas são necessárias para um bom desenvolvimento de software e para garantir que os programas funcionem corretamente e de acordo com as expectativas dos usuários.

3.2 TIPOS DE DADOS

Tipos de dados são uma forma de classificar os diferentes tipos de valores que podem ser manipulados por um programa de computador. Eles são usados para

determinar as operações que podem ser aplicadas a esses valores e como eles devem ser armazenados na memória do computador.

Existem vários tipos de dados comuns em linguagens de programação, dentre os principais temos:

- Inteiros: representam números inteiros, como -3, 0 ou 42. Eles podem ser armazenados em diferentes tamanhos, dependendo da linguagem e do sistema de computador, como 8, 16, 32 ou 64 bits.
- Ponto flutuante: representam números reais com ponto flutuante, como 3.14 ou 0.01. Eles também podem ser armazenados em diferentes tamanhos, como 32 ou 64 bits.
- Booleanos: representam valores lógicos verdadeiros ou falsos. Eles geralmente são armazenados como um único bit.
- Strings: representam uma sequência de caracteres, como "Hello, world!" ou "12345". Eles podem ser armazenados em diferentes tamanhos, dependendo da linguagem e do sistema de computador.

3.2.1 Tipos de dados em linguagens fortemente tipadas

Uma forma comum para garantir a integridade dos dados em uma linguagem fortemente tipada é atribuir um tipo de dado específico a uma variável que armazenará um valor determinado. Por exemplo, se uma variável chamada "Valor" for definida como tipo inteiro, e tentarmos atribuir a ela um valor do tipo string, o compilador ou interpretador da linguagem irá gerar um erro. Isso se deve ao fato de que essas linguagens de programação possuem regras rígidas para o tipo de dados que podem ser armazenados em variáveis, a fim de evitar problemas como a perda de dados ou resultados imprecisos. E em contrapartida gera menos flexibilidade na hora de programar, por isso, é importante que o programador esteja atento aos tipos de dados que estão sendo utilizados em seu código e que as variáveis sejam declaradas com o tipo adequado antes de receberem valores.

3.2.2 Tipos de dados em linguagens fracamente tipadas

Nesse tipo de linguagem não há necessidade em atribuir um tipo de dado a uma variável permitindo assim que ela receba diferentes tipos de dados ao longo da escrita ou execução do código, a fim de proporcionar ao programador mais liberdade e flexibilidade durante a criação do código. Entretanto podem surgir problemas de compatibilidade que dificultam a detecção de erros durante a programação, por isso, é importante que o programador tenha plena consciência do que está fazendo ao manipular uma variável que possua diferentes tipos de dados e sempre realize testes para garantir a integridade do código.

3.3 MODULARIDADE

A Modularidade está diretamente ligada às boas práticas de programação e tem como objetivo dividir o código em partes menores e independentes chamadas de módulos, facilitando assim o gerenciamento do mesmo. Além disso, a modularidade permite que mudanças em um módulo específico sejam feitas sem afetar os outros módulos. Módulos independentes podem ser facilmente incorporados em diferentes projetos e contextos reduzindo a necessidade de reescrever código, aumentando a eficiência durante o desenvolvimento. Não podemos deixar de citar que este princípio é essencial para grandes projetos pois possibilita um modelo de programação em equipe mais eficiente onde cada membro da equipe poderá trabalhar em um módulo distinto sem atrapalhar o trabalho de outros membros da equipe além de beneficiar a legibilidade e manutenção do código, tornando as futuras manutenções mais fáceis e seguras de serem realizadas.

3.4 BIBLIOTECAS

As bibliotecas são um conjunto de funções e rotinas pré-programadas que podem ou não serem implementadas no código durante o desenvolvimento. Em suma, as

bibliotecas, de acordo com a necessidade do programador, acrescentam funcionalidades na qual a linguagem por si só não é capaz de fornecer. Isso ocorre pois é inviável definir todas as funcionalidades existentes em uma determinada linguagem como funcionalidades nativas. Segundo STROUSTRUP:

> "[...] Nenhum programa significativo é escrito apenas com uma linguagem de programação simples. Primeiro um conjunto de bibliotecas é desenvolvido. E estes então formam a base para o trabalho posterior [...]" (2013, tradução nossa)

Cada linguagem possui suas próprias bibliotecas e dentre as mais utilizadas, salvo algumas exceções são as para manipulação de arquivos, conexão com banco de dados, criação de interfaces gráficas, desenvolvimento de APIs, manipulação de data e hora dentre outras. É necessário que o desenvolvedor tenha ciência de que as bibliotecas são essenciais para qualquer tipo de projeto e estarão presentes para facilitar o processo de desenvolvimento, tornar possível a modularidade e portabilidade e assim gerar códigos menores reduzindo a chance de erros e retrabalhos. Para isso, o desenvolvedor deve buscar dominar o conhecimento sobre as principais bibliotecas da linguagem escolhida.

3.5 PORTABILIDADE

A portabilidade se refere à capacidade de um programa ser executado em diferentes sistemas operacionais ou plataformas de hardware, sem a necessidade de modificações significativas no código, ela é uma característica altamente desejável em um mundo cada vez mais conectado e tecnologicamente diverso. A capacidade de um programa ser executado em diferentes plataformas é particularmente importante em áreas como o desenvolvimento de aplicativos móveis e web, onde há uma ampla gama de dispositivos com diferentes sistemas operacionais e capacidades de hardware. Isso é importante porque permite que os desenvolvedores escrevam um código que pode ser executado em diferentes ambientes, o que aumenta a compatibilidade e a flexibilidade do software. As linguagens de programação que priorizam a portabilidade geralmente utilizam abstrações de

hardware e sistemas operacionais para permitir que o código seja compilado e executado em diferentes ambientes sem maiores problemas.

4 CONSIDERAÇÕES FINAIS

Este artigo buscou demonstrar a importância dos princípios das linguagens de programação para o desenvolvimento de software. Através de pesquisas bibliográficas, foi possível identificar que dentre os principais princípios estão a sintaxe, a semântica, o tipo de dados, a modularidade, as bibliotecas e a portabilidade. Cada um desses princípios possui uma importância fundamental para o desenvolvimento de software, seja para garantir a qualidade do código, facilitar a manutenção, reutilizar códigos já desenvolvidos, permitir a execução em diferentes plataformas ou melhorar a eficiência do desenvolvimento.

Por fim, é importante ressaltar que as linguagens de programação estão em constante evolução, e novas linguagens surgem a todo momento. Por isso, é essencial que os desenvolvedores estejam sempre atualizados sobre as novas tecnologias e tendências do mercado, a fim de produzir códigos cada vez mais eficientes e eficazes. O conhecimento dos princípios fundamentais das linguagens de programação é essencial para isso, pois eles fornecem uma base sólida e orientação para a tomada de decisões no processo de desenvolvimento de software.

5 REFERÊNCIAS

AHO, Alfred V. et al. **Compiladores: princípios, técnicas e ferramentas**. 2 ed. São Paulo: Pearson, 2008.

BIGONHA, Roberto S. PROGRAMAÇÃO MODULAR: Programação Orientada por **Objetos - Java**. 1 ed. Belo Horizonte: Câmara Brasileira do Livro, 2021.

HOST MÍDIA BLOG. **Aprenda tudo sobre o que é uma linguagem de programação e para que ela serve**. São Paulo. Disponível em: https://www.hostmidia.com.br/blog/linguagem-de-programacao/ Acessado em: 16/08/2023

MARTIN, Robert C. **CÓDIGO LIMPO: Habilidades Práticas do Agile Software.** 1 ed. Rio de Janeiro: Alta Books, 2009.

MELO, Ana Cristina Vieira de; SILVA, Flávio Soares Corrêa da. **Princípios de Linguagens de Programação**. 3 ed. São Paulo: Blucher, 2014.

OLIVEIRA, Lanna Mayra. Caracterização do Conceito de Modularidade no **Desenvolvimento de Linguagens de Programação.** Minas Gerais: UFOP, 2017. Disponível em: https://www.monografias.ufop.br/bitstream/35400000/367/1/MONOGRAFIA_Caracteri_za%C3%A7%C3%A3oConceitoModularidade.pdf Acessado em: 19/06/2023

SEBESTA, Robert W. **Conceitos de Linguagens de Programação**. 11 ed. Rio de Janeiro: Alta Books, 2018.

SILVA, Gizele. **O que é biblioteca**. Belo Horizonte: COODESH. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-biblioteca/ Acessado em: 19/06/2023

STROUSTRUP, Bjarne. **The C++ Programing Language**. 4 ed. New Jersey: Pearson, 2013.

VALENTE, Marco Tulio. **Princípios e Práticas para Desenvolvimento de Software com Produtividade**. 1 ed. Minas Gerais: Independente, 2020