As linguagens de programação podem ser classificadas de várias formas, mas eu vou classificá-las segundo ao seu paradigma.

O paradigma de uma linguagem de programação é a sua identidade. Corresponde a um conjunto de características que, juntas, definem como ela opera e resolve os problemas. Algumas linguagens, inclusive, possuem mais de um paradigma, são as chamadas multi paradigmas.

Alguns dos principais paradigmas utilizados hoje no mercado:

* Funcional
* Lógico
* Declarativo
* Imperativo
* Orientado a objetos
* Orientado a eventos

FUNCIONAL: Em [ciência da computação](https://pt.wikipedia.org/wiki/Ci%C3%AAncia_da_computa%C3%A7%C3%A3o), programação funcional é um [paradigma de programação](https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o) que trata a computação como uma avaliação de [funções matemáticas](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_matem%C3%A1tica) e que evita estados ou dados mutáveis. Ela enfatiza a aplicação de [funções](https://pt.wikipedia.org/wiki/Sub-rotina), em contraste da [programação imperativa](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa), que enfatiza mudanças no estado do programa[[1]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional#cite_note-1).

Uma função, neste sentido, pode ter ou não ter [parâmetros](https://pt.wikipedia.org/wiki/Par%C3%A2metro_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)) e um simples valor de retorno. Os parâmetros são os valores de entrada da função, e o valor de retorno é o resultado da função. A definição de uma função descreve como a função será avaliada em termos de outras funções. Por exemplo, a função f(x)=x^{2}+2 é definida em termos de funções de [exponenciação](https://pt.wikipedia.org/wiki/Exponencia%C3%A7%C3%A3o) e [adição](https://pt.wikipedia.org/wiki/Adi%C3%A7%C3%A3o). Do mesmo modo, a linguagem deve oferecer funções básicas que não requerem definições adicionais.

[Linguagens de programação](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o) funcionais, especialmente as puramente funcionais, tem sido mais usadas academicamente que no [desenvolvimento comercial de software](https://pt.wikipedia.org/wiki/Desenvolvimento_de_software). Entretanto, algumas linguagens notáveis usadas na indústria e no comércio incluem [Erlang](https://pt.wikipedia.org/wiki/Erlang_(linguagem_de_programa%C3%A7%C3%A3o)) (aplicações concorrentes)[[2]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional#cite_note-2), [R](https://pt.wikipedia.org/wiki/R_(linguagem_de_programa%C3%A7%C3%A3o)) (estatística), [Mathematica](https://pt.wikipedia.org/wiki/Mathematica) (matemática simbólica)[[3]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional#cite_note-3) [J](https://pt.wikipedia.org/wiki/J_(linguagem_de_programa%C3%A7%C3%A3o)), [K](https://pt.wikipedia.org/w/index.php?title=K_(linguagem_de_programa%C3%A7%C3%A3o)&action=edit&redlink=1)(análise financeira) e [XSLT](https://pt.wikipedia.org/wiki/XSLT).[[4]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional#cite_note-4)[[5]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional#cite_note-5) Importantes influências na programação funcional foram o [cálculo lambda](https://pt.wikipedia.org/wiki/C%C3%A1lculo_lambda), as linguagens de programação [APL](https://pt.wikipedia.org/wiki/APL) e [Lisp](https://pt.wikipedia.org/wiki/Lisp), e mais recentemente [ML](https://pt.wikipedia.org/wiki/ML_(linguagem_de_programa%C3%A7%C3%A3o)), [Haskell](https://pt.wikipedia.org/wiki/Haskell_(linguagem_de_programa%C3%A7%C3%A3o)), [OCaml](https://pt.wikipedia.org/wiki/OCaml), [F#](https://pt.wikipedia.org/wiki/F_Sharp) e [Elixir](https://pt.wikipedia.org/wiki/Elixir_(linguagem_de_programa%C3%A7%C3%A3o)).

LÓGICO: Programação lógica é um [paradigma de programação](https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o) que faz uso da [lógica matemática](https://pt.wikipedia.org/wiki/L%C3%B3gica_matem%C3%A1tica). [John McCarthy](https://pt.wikipedia.org/wiki/John_McCarthy) [1958] foi o primeiro a publicar uma proposta de uso da lógica matemática para [programação](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o).

A primeira linguagem de programação lógica foi a [Planner](https://pt.wikipedia.org/wiki/Planner), a qual permitia a invocação orientada a padrões de planos procedimentais de asserções e de objetivos. Com a necessidade de adaptação aos sistemas de [memória](https://pt.wikipedia.org/wiki/Mem%C3%B3ria_(computador)) muito limitada, que eram disponíveis quando ela foi desenvolvida. A linguagem Planner usava [estruturas de controle](https://pt.wikipedia.org/wiki/Estrutura_de_controle) de [backtracking](https://pt.wikipedia.org/wiki/Backtracking), de tal forma que apenas um único caminho computacional tinha que ser armazenado por vez. Em seguida, o [Prolog](https://pt.wikipedia.org/wiki/Prolog) foi desenvolvido como uma simplificação do Planner que permitia a invocação orientada a padrões apenas a partir de objetivos (também baseado em backtracking).[[1]](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica#cite_note-1)

A partir do Planner, foram desenvolvidas as linguagens de programação QA-4, Popler, Conniver, e QLISP. As linguagens de programação [Mercury](https://pt.wikipedia.org/wiki/Mercury), Visual Prolog, [Oz](https://pt.wikipedia.org/wiki/Oz_(linguagem_de_programa%C3%A7%C3%A3o)) e Frill, foram desenvolvidas a partir do Prolog. Atualmente existem linguagens de programação lógica [concorrente](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_concorrente) (não baseadas em backtracking) derivadas do Planner (por exemplo, a Ether) e derivadas do Prolog (ver Shapiro [1989](https://pt.wikipedia.org/wiki/1989) para um apanhado geral).

DECLARATIVO: Programação declarativa é um [paradigma de programação](https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o) baseado em [programação funcional](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional), [programação lógica](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica) ou [programação restritiva](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_restritiva). Tal termo é utilizado para discernir tais linguagens em relação à [linguagens de programação imperativa](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa).

Outra definição determina que um programa é declarativo se descreve o que ele faz e não como seus procedimentos funcionam. Dessa forma, [linguagens de marcação](https://pt.wikipedia.org/wiki/Linguagem_de_marca%C3%A7%C3%A3o) são declarativas pois descrevem o que são suas estruturas e não como elas serão utilizadas. Tal definição foge do escopo pois não trata-se de [linguagens de programação](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o).

Em particular, a programação restritiva e a programação lógica descrevem propriedades da solução desejada, não especificando como o algoritmo em si deve agir. Apesar disso a maioria das linguagens desses paradigmas permite que os algoritmos sejam implementados com detalhes, então não constituem uma linguagem declarativa pura por definição.

Similarmente, é possível escrever programas de forma declarativa utilizando linguagens de programação imperativas. Isso pode ser obtido utilizando métodos como [encapsulamento](https://pt.wikipedia.org/wiki/Encapsulamento) para esconder detalhes de implementação fora de uma [biblioteca](https://pt.wikipedia.org/wiki/Biblioteca_de_rotinas).

[linguagens funcionais](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional) que admitem tal abordagem incluem [Erlang](https://pt.wikipedia.org/wiki/Erlang_(linguagem_de_programa%C3%A7%C3%A3o)), [Haskell](https://pt.wikipedia.org/wiki/Haskell_(linguagem_de_programa%C3%A7%C3%A3o)) e [Lisp](https://pt.wikipedia.org/wiki/Lisp). [Linguagens lógicas](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica) que admitem tal abordagem incluem [Prolog](https://pt.wikipedia.org/wiki/Prolog) e [Mercury](https://pt.wikipedia.org/w/index.php?title=Mercury_(linguagem_de_programa%C3%A7%C3%A3o)&action=edit&redlink=1). [Linguagens restritivas que](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_restritiva) admitem tal abordagem incluem [Oz](https://pt.wikipedia.org/wiki/Oz_(linguagem_de_programa%C3%A7%C3%A3o)).

IMPERATIVO: O paradigma imperativo de programação é o mais antigo de todos os paradigmas de programação. Este paradigma se baseia no modo de funcionamento do computador, ou seja, é influenciado pela arquitetura do computador. Isto é refletido na execução seqüencial baseada em comandos e no armazenamento de dados alterável, conceitos que são baseados na maneira pela qual computadores executam os programas a nível de linguagem de máquina. O termo “imperare” em Latim significa “comandar”. O paradigma imperativo foi predominante nas LP, pois tais linguagens são mais fáceis de traduzir para uma forma adequada para execução na máquina.

ORIENTADO A OBJETOS: Programação Orientada a Objetos (também conhecida pela sua sigla POO) é um modelo de [análise](https://pt.wikipedia.org/wiki/An%C3%A1lise_de_sistemas), [projeto](https://pt.wikipedia.org/wiki/Padr%C3%A3o_de_projeto_de_software) e [programação](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o) de [software](https://pt.wikipedia.org/wiki/Software) baseado na composição e interação entre diversas unidades chamadas de 'objetos'.[[1]](https://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos#cite_note-oop-1) A POO é um dos 4 principais paradigmas de programação (as outras são programação [imperativa](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa), [funcional](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional) e [lógica](https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica)). Os objetos são operados com o conceito de 'this' (isso) ou 'self' (si), de forma que seus métodos (muitas vezes) modifiquem os dados da própria instância. Os programas são arquitetados através de objetos que interagem entre si. Dentre as várias abordagens da POO, as baseadas em classes são as mais comuns: objetos são instâncias de classes, o que em geral também define o tipo do objeto. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.[[2]](https://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos#cite_note-Meyer2009-2) A alternativa mais usual ao uso de classes é o uso de protótipos. Neste caso, objetos são copias de outros objetos, não instâncias de classes. Javascript e [Lua](https://pt.wikipedia.org/wiki/Lua_(linguagem_de_programa%C3%A7%C3%A3o)) são exemplos de linguagens cuja POO é realizada por protótipos. A diferença prática mais evidente é que na POO baseada em protótipos apenas a herança simples é implementada pela cópia do objeto. Assim, na POO, implementa-se um conjunto de classes passíveis de serem instanciadas como objetos, e.g. [Python](https://pt.wikipedia.org/wiki/Python) e [C++](https://pt.wikipedia.org/wiki/C%2B%2B) (ou objetos protótipos que são copiados e alterados, e.g. [JavaScript](https://pt.wikipedia.org/wiki/JavaScript) e [VimL](https://pt.wikipedia.org/w/index.php?title=VimL&action=edit&redlink=1)).

Em alguns contextos, o termo [modelagem](https://pt.wikipedia.org/wiki/Modelagem_de_dados) orientada ao objeto ([MOO](https://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos#MOO)) é preferível ao termo POO. De fato, o paradigma "orientado ao objeto" tem origem nos estudos da [cognição](https://pt.wikipedia.org/wiki/Cogni%C3%A7%C3%A3o) e influenciou a [inteligência artificial](https://pt.wikipedia.org/wiki/Intelig%C3%AAncia_artificial) e a [linguística](https://pt.wikipedia.org/wiki/Lingu%C3%ADstica), dada a relevância para a abstração de conceitos do mundo real. A [MOO](https://pt.wikipedia.org/wiki/MOO) é considerada a melhor estratégia para diminuir o "gap semântico" (o hiato entre o mundo real e a representação dele), e facilita a comunicação das partes interessadas no [modelo](https://pt.wikipedia.org/wiki/Modelo) ou [software](https://pt.wikipedia.org/wiki/Software) (e.g. o [modelador](https://pt.wikipedia.org/wiki/Projetista) e o [usuário final](https://pt.wikipedia.org/wiki/Usu%C3%A1rio_final)) na medida em que conceitos, terminologia, símbolos, grafismo e estratégias, são, potencialmente, mais óbvios, intuitivos, naturais e exatos.

Muitas das linguagens de programação mais utilizadas atualmente (talvez a maioria) são multi-paradigma com suporte à POO. [C++](https://pt.wikipedia.org/wiki/C%2B%2B), [C#](https://pt.wikipedia.org/wiki/C_Sharp), [VB.NET](https://pt.wikipedia.org/wiki/VB.NET), [Java](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_Java), [Object Pascal](https://pt.wikipedia.org/wiki/Object_Pascal), [Objective-C](https://pt.wikipedia.org/wiki/Objective-C), [Python](https://pt.wikipedia.org/wiki/Python), [SuperCollider](https://pt.wikipedia.org/wiki/SuperCollider), [Ruby](https://pt.wikipedia.org/wiki/Ruby_(linguagem_de_programa%C3%A7%C3%A3o)) e [Smalltalk](https://pt.wikipedia.org/wiki/Smalltalk) são exemplos de linguagens de programação orientadas a objetos. [ActionScript](https://pt.wikipedia.org/wiki/ActionScript), [ColdFusion](https://pt.wikipedia.org/wiki/ColdFusion), [Javascript](https://pt.wikipedia.org/wiki/Javascript), [PHP](https://pt.wikipedia.org/wiki/PHP) (a partir da versão 4.0), [Perl](https://pt.wikipedia.org/wiki/Perl)(a partir da versão 5), [Visual Basic](https://pt.wikipedia.org/wiki/Visual_Basic) (a partir da versão 4), [VimL](https://pt.wikipedia.org/w/index.php?title=VimL&action=edit&redlink=1) (ou Vim script) são exemplos de linguagens de programação com suporte a orientação a objetos. Vivace é um exemplo de linguagem sem suporte à POO.

ORIENTADO A EVENTOS: Também conhecida como programação baseada em eventos, também é um estilo de programar no qual a execução do programa é feita por eventos, guiado por sinais externos ao programa. Nesse paradigma os programas são pequenos trechos de códigos que executam ações em respostas a determinados eventos, isto é ele trata eventos para produzir respostas, que realiza uma tarefa em resposta a um evento chamado handler do evento. Handler nada mais é que um método que está associado a um evento, no sentido de que seja realizada alguma ação quando o mesmo ocorre.

A maioria dos eventos é associada em desenvolvimento de formulários e componentes de tela, é amplamente utilizado em desenvolvimento de sistemas que possuem interface com o usuário, que o seu funcionamento ocorre em um loop de eventos que ao ser disparado reage de acordo com uma função vinculada.

Exemplo de um evento quando o usuário clica em um botão para salvar um cadastro, mostra uma mensagem para o usuário se ele deseja ou não salvar as informações digitadas.