

# Capítulo 1

## **Introdução à programação**

Objetivo: Conhecer o que é programação, como ela funciona e quais são as suas aplicações.

- O que é programação?
- O que é um algoritmo?
- O que é uma linguagem de programação e quais são os tipos?
- Como escolher uma linguagem de programação?

# O que é programação?

Programação é o processo de criar programas de computador, que são conjuntos de instruções que dizem ao computador como realizar uma determinada tarefa. Para isso, é preciso usar uma linguagem de programação, que é uma forma de comunicação entre humanos e máquinas, baseada em regras, símbolos e palavras-chave.

Existem muitas linguagens de programação diferentes, cada uma com suas características, vantagens e desvantagens. Algumas das linguagens mais conhecidas são C, Java, Ruby e Python. Essas linguagens podem ser comparadas em vários aspectos, como:

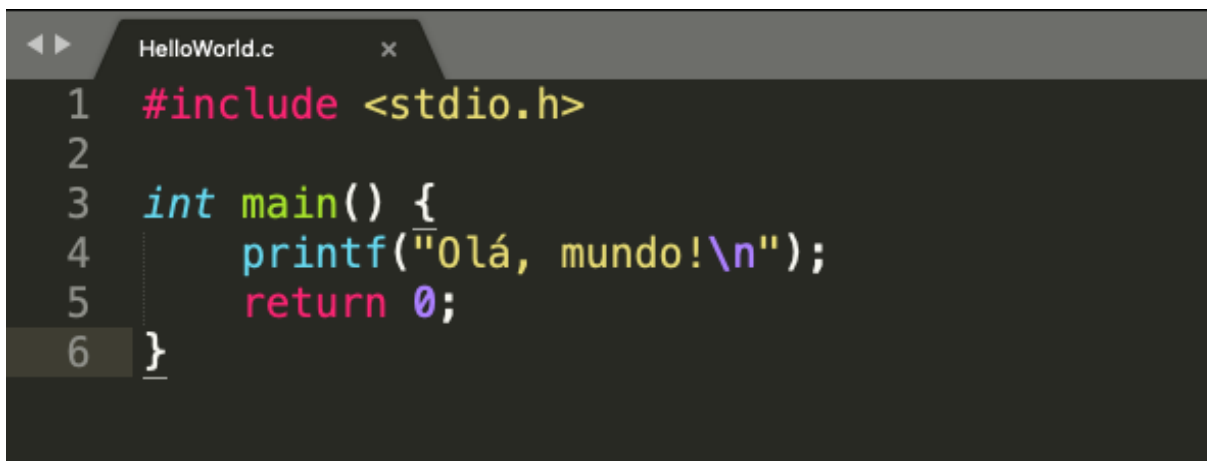
- **Paradigma:** O paradigma de uma linguagem de programação é a forma como ela organiza e estrutura o código, definindo os conceitos e as regras que o programador deve seguir. Existem vários paradigmas de programação, como imperativo, declarativo, funcional, orientado a objetos, entre outros. Cada paradigma tem suas vantagens e desvantagens, dependendo do tipo de problema que se quer resolver.
  - C é uma linguagem de programação **imperativa**, ou seja, ela foca em como o programa deve executar as instruções passo a passo, usando variáveis, estruturas de controle e funções. C também é uma linguagem de **baixo nível**, ou seja, ela permite um controle mais direto sobre o hardware do computador, como a memória e os dispositivos de entrada e saída. Isso torna C uma linguagem muito rápida e eficiente, mas também mais complexa e propensa a erros.
  - Java é uma linguagem de programação **orientada a objetos**, ou seja, ela foca em como o programa deve representar os dados e as operações usando objetos, que são entidades que possuem atributos (características) e métodos (ações). Java também é uma linguagem de **alto nível**, ou seja, ela abstrai os detalhes do hardware do computador, usando uma máquina virtual que permite que o mesmo código rode em diferentes plataformas e dispositivos. Isso torna Java uma linguagem muito portátil e versátil, mas também mais lenta e pesada.
  - Ruby é uma linguagem de programação **multiparadigma**, ou seja, ela suporta vários paradigmas de programação,

como imperativo, funcional e orientado a objetos. Ruby também é uma linguagem de **alto nível**, ou seja, ela oferece muitas facilidades e recursos para o programador, como sintaxe simples e expressiva, tipagem dinâmica (que não exige a declaração dos tipos das variáveis), coleta automática de lixo (que libera a memória ocupada por objetos não usados) e metaprogramação (que permite modificar o comportamento da linguagem em tempo de execução). Isso torna Ruby uma linguagem muito produtiva e flexível, mas também mais lenta e menos previsível.

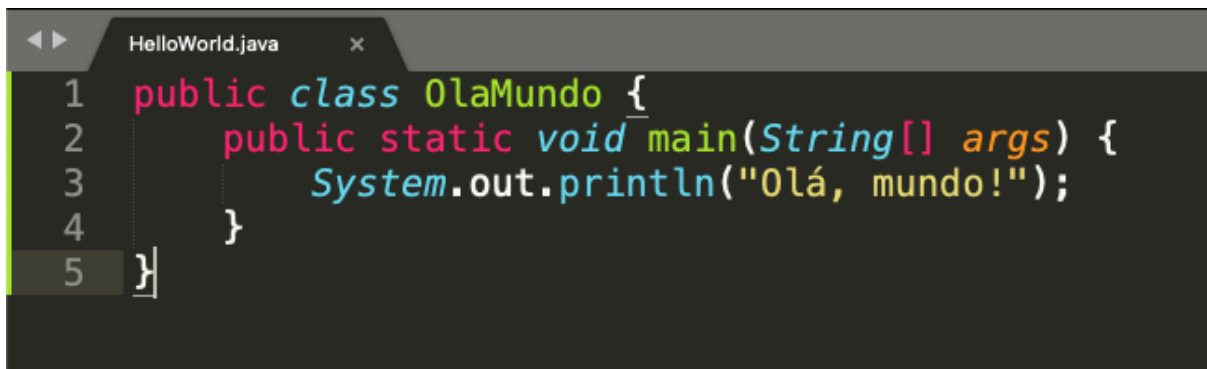
- Python é uma linguagem de programação **multiparadigma**, ou seja, ela suporta vários paradigmas de programação, como imperativo, funcional e orientado a objetos. Python também é uma linguagem de **alto nível**, ou seja, ela oferece muitas facilidades e recursos para o programador, como sintaxe clara e consistente, tipagem dinâmica (que não exige a declaração dos tipos das variáveis), coleta automática de lixo (que libera a memória ocupada por objetos não usados) e um grande número de bibliotecas (que fornecem funções prontas para diversas áreas e aplicações). Isso torna Python uma linguagem muito popular e versátil, mas também mais lenta e menos otimizada.
- Sintaxe: A sintaxe de uma linguagem de programação é o conjunto de regras que definem como o código deve ser escrito e formatado. A sintaxe pode variar muito entre as linguagens de programação, afetando a legibilidade e a manutenção do código.
  - C tem uma sintaxe baseada em símbolos especiais, como chaves (`{ }`), ponto-e-vírgula (`;`), asterisco (`*`) e outros. C também exige que o programador declare os tipos das variáveis antes de usá-las. Além disso, C usa a notação prefixa para as operações aritméticas (por exemplo: `*p = x + y;` significa que o valor de `x + y` deve ser atribuído ao endereço de memória apontado por `p`). Essas características tornam a sintaxe de C mais compacta e precisa, mas também mais difícil de ler e entender.
  - Java tem uma sintaxe baseada em palavras-chave, como `class`, `public`, `static`, `void` e outras. Java também exige que o programador declare os tipos das variáveis antes de usá-las. Além disso, Java usa a notação infixa para as operações aritméticas (por exemplo: `a = b + c;`

significa que o valor de `b + c` deve ser atribuído à variável `a`). Essas características tornam a sintaxe de Java mais explícita e padronizada, mas também mais verbosa e repetitiva.

- Ruby tem uma sintaxe baseada em palavras-chave, como `def`, `end`, `class`, `module` e outras. Ruby não exige que o programador declare os tipos das variáveis antes de usá-las. Além disso, Ruby usa a notação infixa para as operações aritméticas (por exemplo: `a = b + c`; significa que o valor de `b + c` deve ser atribuído à variável `a`). Essas características tornam a sintaxe de Ruby mais simples e elegante, mas também mais ambígua e inconsistente.
- Python tem uma sintaxe baseada em palavras-chave, como `def`, `return`, `class`, `import` e outras. Python não exige que o programador declare os tipos das variáveis antes de usá-las. Além disso, Python usa a notação infixa para as operações aritméticas (por exemplo: `a = b + c`; significa que o valor de `b + c` deve ser atribuído à variável `a`). Essas características tornam a sintaxe de Python mais clara e consistente, mas também mais sensível e restritiva.
- Exemplos: Um exemplo de um programa simples que imprime na tela a mensagem “Olá, mundo!” em cada uma das linguagens é:
  - C

A screenshot of a code editor window titled "HelloWorld.c". The code is written in C and consists of six lines. Line 1: `#include <stdio.h>`. Line 2: (empty). Line 3: `int main() {`. Line 4: `printf("Olá, mundo!\n");`. Line 5: `return 0;`. Line 6: `}`. The code is color-coded: `#include` is pink, `<stdio.h>` is yellow, `int` is blue, `main()` is green, `printf` is blue, the string is yellow, `\n` is purple, `return` is pink, and `0` is purple. The line numbers 1 through 6 are on the left side of the editor.

- JAVA



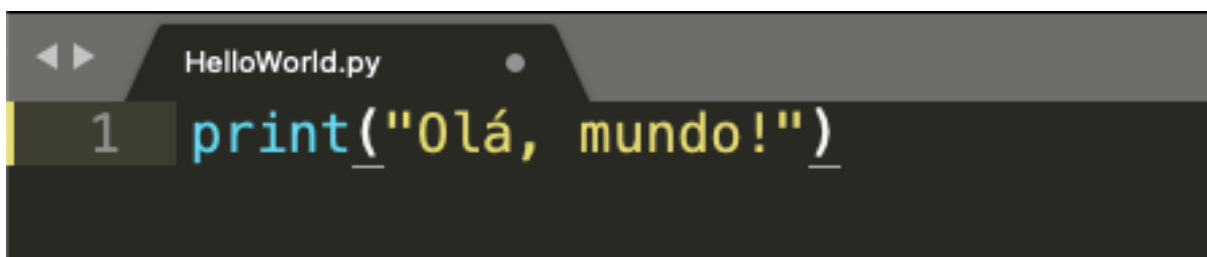
```
1 public class OlaMundo {  
2     public static void main(String[] args) {  
3         System.out.println("Olá, mundo!");  
4     }  
5 }
```

- RUBY



```
1 puts "Olá, mundo!"
```

- PYTHON



```
1 print("Olá, mundo!")
```

# O que é um algoritmo?

Um algoritmo é uma sequência de instruções ou comandos realizados de maneira sistemática com o objetivo de resolver um problema ou executar uma tarefa. A palavra “algoritmo” faz referência ao matemático árabe Al Khwarizmi, que viveu no século IX, e descreveu regras para equações matemáticas.

Os algoritmos são como uma receita de bolo: uma sequência de ações que devem ser executadas para que o objetivo final — o bolo pronto — seja atingido. Aplicam-se os algoritmos nas tarefas simples do dia a dia e também nos programas computacionais complexos que identificam o comportamento do consumidor na internet. Todas as funções dos computadores, smartphones e tablets, por exemplo, resultam de algoritmos. Essas máquinas conseguem realizar bilhões de comandos em poucos segundos.

Para escrever um algoritmo, é preciso usar uma linguagem de programação, que é uma forma de comunicação entre humanos e máquinas, baseada em regras, símbolos e palavras-chave. Existem muitas linguagens de programação diferentes, cada uma com suas características, vantagens e desvantagens. Algumas das linguagens mais conhecidas são HTML, Python, Java e JavaScript. Cada uma delas tem suas aplicações e finalidades.

Uma das linguagens de programação usadas para aprender os conceitos básicos de algoritmos é o portugol, também conhecido como português estruturado. O portugol é uma pseudolinguagem que permite ao programador pensar no problema em si e não no equipamento que irá executar o algoritmo. O portugol tem uma sintaxe simples e baseada na língua portuguesa, facilitando o entendimento do código.

O portugol pode ser usado em diversos editores ou compiladores, como o VisuAlg, o Portugol Studio e o Mapler. Essas ferramentas permitem escrever, executar e depurar o código em portugol, além de oferecer recursos como exemplos, ajuda, temas e suporte a jogos.

Um exemplo de código em português que calcula a área de um círculo é:

```
AreaCirculo.alg x
1 // Este é um comentário que explica o objetivo do programa
2 algoritmo "area-circulo"
3 // Este é o início do bloco principal do algoritmo
4
5 // Declaração das variáveis usadas no programa
6 var
7     raio: real // O raio do círculo
8     area: real // A área do círculo
9     pi: real // O valor aproximado de pi
10 inicio
11     // Atribuição dos valores às variáveis
12     raio <- 10 // O raio recebe 10 unidades
13     pi <- 3.14 // Pi recebe 3.14
14
15     // Cálculo da área do círculo usando a fórmula  $A = \pi r^2$ 
16     area <- pi * (raio ^ 2)
17
18     // Exibição do resultado na tela
19     escreva("A área do círculo é: ", area)
20
21 fimalgoritmo // Este é o fim do bloco principal do algoritmo
```

# O que é uma linguagem de programação?

Uma linguagem de programação é um conjunto de símbolos e regras que permitem expressar instruções para um computador realizar uma determinada tarefa. Uma linguagem de programação é uma forma de comunicação entre humanos e máquinas, baseada em uma sintaxe (a forma como o código é escrito) e uma semântica (o significado do código).

Existem muitos tipos de linguagens de programação, cada uma com suas características, vantagens e desvantagens. Alguns dos tipos mais comuns são:

- **Linguagens de Programação de Alto Nível:** São linguagens que se aproximam da linguagem natural humana, facilitando o entendimento e a escrita do código. Elas são mais abstratas e independentes do hardware do computador, ou seja, não dependem dos detalhes da máquina para funcionar. Elas são traduzidas para a linguagem de máquina por meio de compiladores ou interpretadores, que são programas que convertem o código de alto nível em código de baixo nível. Alguns exemplos de linguagens de alto nível são Python, Java, C#, Ruby e PHP.
- **Linguagens de Programação de Baixo Nível:** São linguagens que se aproximam da linguagem da máquina, ou seja, do conjunto de instruções binárias que o computador entende. Elas são mais rápidas e eficientes, mas também mais complexas e difíceis de escrever e ler. Elas dependem do hardware do computador, ou seja, cada máquina pode ter uma linguagem de baixo nível diferente. Elas não precisam ser traduzidas por compiladores ou interpretadores, pois já estão na forma que o computador pode executar. Alguns exemplos de linguagens de baixo nível são Assembly e C.
- **Linguagens de Scripting:** São linguagens que são usadas para automatizar tarefas ou controlar outros programas. Elas são geralmente interpretadas em tempo real, ou seja, o código é executado à medida que é lido pelo interpretador. Elas são usadas para criar scripts, que são pequenos programas que realizam funções específicas. Elas podem ser incorporadas em outras



linguagens ou plataformas, como páginas web ou sistemas operacionais. Alguns exemplos de linguagens de scripting são JavaScript, Python, Perl e Bash.

- **Linguagens Orientadas a Objetos:** São linguagens que se baseiam no conceito de objetos, que são entidades que possuem atributos (dados) e métodos (funções). Os objetos podem se comunicar entre si por meio de mensagens, que são chamadas aos métodos. Os objetos podem ser organizados em classes, que são modelos que definem as características e comportamentos dos objetos. As classes podem ser relacionadas por meio de herança (quando uma classe herda os atributos e métodos de outra classe) ou composição (quando uma classe contém objetos de outra classe). As linguagens orientadas a objetos permitem o encapsulamento (ocultar os detalhes internos dos objetos), a abstração (representar os objetos por meio de conceitos gerais), o polimorfismo (permitir que objetos de diferentes classes se comportem de forma diferente diante da mesma mensagem) e a reutilização (aproveitar o código já existente para criar novos objetos). Alguns exemplos de linguagens orientadas a objetos são Java, C#, Python, Ruby e PHP.
- **Linguagens Funcionais:** São linguagens que se baseiam no conceito de funções matemáticas, que recebem valores como entrada e retornam valores como saída. As funções podem ser compostas por outras funções, criando funções mais complexas. As funções não alteram o estado dos valores recebidos nem produzem efeitos colaterais (como alterar variáveis globais ou imprimir na tela). As funções podem ser tratadas como valores, podendo ser armazenadas em variáveis, passadas como parâmetros ou retornadas como resultados. As linguagens funcionais permitem o uso de recursão (quando uma função chama a si mesma), listas (estruturas que armazenam vários valores), padrões (formas de verificar a estrutura dos valores) e expressões lambda (funções anônimas criadas em tempo de execução). Alguns exemplos de linguagens funcionais são Haskell, Lisp, Scheme e Clojure.
- **Linguagens de Programação Web:** São linguagens que são usadas para criar páginas ou aplicações web, que são programas que rodam em um navegador de internet. Elas podem ser divididas em linguagens de programação front-end, que são responsáveis pela interface gráfica e interação com o usuário, e linguagens de programação back-end, que são responsáveis pela

lógica de negócio e comunicação com o banco de dados. As linguagens de programação web podem ser estáticas, ou seja, geram páginas que não mudam de acordo com o usuário ou o tempo, ou dinâmicas, ou seja, geram páginas que se adaptam ao contexto e às ações do usuário. Alguns exemplos de linguagens de programação web são HTML, CSS, JavaScript, PHP, Ruby on Rails e Django.

- **Linguagens de Programação de Banco de Dados:** São linguagens que são usadas para manipular dados armazenados em um banco de dados, que é um sistema que organiza e gerencia informações. Elas permitem realizar operações como criar, consultar, atualizar e deletar dados, além de definir a estrutura e as relações entre os dados. Elas podem ser divididas em linguagens de definição de dados (DDL), que são usadas para criar e modificar a estrutura do banco de dados, e linguagens de manipulação de dados (DML), que são usadas para inserir, alterar, consultar e remover os dados. Alguns exemplos de linguagens de programação de banco de dados são SQL, MongoDB e Oracle.
- **Linguagens de Programação de Domínio Específico (DSLs):** São linguagens que são projetadas para resolver problemas específicos de um domínio ou área de aplicação. Elas são mais simples e expressivas do que as linguagens de propósito geral, pois possuem uma sintaxe e uma semântica adaptadas ao contexto. Elas podem ser classificadas em internas ou externas, dependendo se elas são implementadas dentro ou fora de uma linguagem hospedeira. Alguns exemplos de DSLs são LaTeX (para produção de documentos científicos), R (para análise estatística), MATLAB (para computação numérica) e CSS (para estilização de páginas web).
- **Linguagens de Programação de Baixa Latência e Tempo Real:** São linguagens que são usadas para criar sistemas que exigem uma resposta rápida e previsível do computador, sem atrasos ou variações significativas no tempo de execução. Elas são usadas para aplicações críticas que envolvem controle, monitoramento ou comunicação com dispositivos externos, como robôs, aviões ou sensores. Elas requerem um gerenciamento eficiente dos recursos do computador, como memória, processador e rede. Elas podem ser classificadas em linguagens duras ou suaves, dependendo se elas garantem ou não o cumprimento dos prazos estabelecidos. Alguns exemplos de linguagens de baixa latência e tempo real são C, Ada, Rust e Elixir.

- Linguagens de Programação para Inteligência Artificial e Aprendizado de Máquina: São linguagens que são usadas para criar sistemas que simulam a capacidade humana de aprender, raciocinar e resolver problemas complexos. Elas permitem o uso de técnicas como redes neurais artificiais, algoritmos genéticos, sistemas especialistas e processamento natural da linguagem. Elas facilitam a manipulação e análise de grandes volumes de dados, a geração e teste de hipóteses e a adaptação a novos cenários. Alguns exemplos de linguagens para inteligência artificial e aprendizado de máquina são Python, R, Prolog e Lisp.

# Como escolher uma linguagem de programação?

Escolher uma linguagem de programação para um projeto pode ser uma decisão difícil, pois existem muitas opções disponíveis, cada uma com suas vantagens e desvantagens. Para fazer uma escolha adequada, é preciso considerar vários fatores, como:

- O objetivo e o escopo do projeto: Qual é o tipo de aplicação que você quer desenvolver? É um site, um aplicativo móvel, um jogo, um sistema embarcado, uma inteligência artificial ou algo diferente? Cada tipo de aplicação pode exigir uma linguagem de programação diferente, dependendo das funcionalidades, da interface, da interação e da performance que você deseja alcançar.
- O mercado e a demanda: Qual é o público-alvo do seu projeto? Quais são as expectativas e as necessidades dos seus usuários? Quais são as tendências e as oportunidades do mercado para o seu produto? Você quer trabalhar em uma empresa ou ser um freelancer? Cada mercado pode ter uma preferência por uma linguagem de programação diferente, dependendo da popularidade, da disponibilidade, da compatibilidade e da segurança que ela oferece.
- O tempo e o custo: Quanto tempo e dinheiro você tem para investir no seu projeto? Você quer aprender uma linguagem de programação nova ou usar uma que você já conhece? Você tem acesso a recursos e ferramentas que facilitam o seu trabalho? Cada linguagem de programação pode ter um impacto diferente no tempo e no custo do seu projeto, dependendo da facilidade de aprendizado, da produtividade, da qualidade e da manutenção que ela proporciona.

Para ilustrar como esses fatores podem influenciar na escolha de uma linguagem de programação, vamos comparar algumas linguagens que você mencionou:

- Assembly: É uma linguagem de baixo nível, que se aproxima da linguagem da máquina. Ela é muito rápida e eficiente, mas também muito complexa e difícil de escrever e ler. Ela é usada para criar sistemas que exigem um controle direto sobre o

hardware do computador, como robôs, aviões ou sensores. Ela é indicada para projetos que requerem baixa latência e tempo real, mas não para projetos que requerem alta portabilidade e facilidade de desenvolvimento.

- C: É uma linguagem de baixo nível, mas com alguns recursos de alto nível. Ela é muito rápida e poderosa, mas também propensa a erros e vulnerabilidades. Ela é usada para criar sistemas operacionais, compiladores, jogos e aplicações gráficas. Ela é indicada para projetos que requerem performance e flexibilidade, mas não para projetos que requerem segurança e robustez.
- Java: É uma linguagem de alto nível e orientada a objetos. Ela é muito portátil e versátil, mas também lenta e pesada. Ela é usada para criar aplicações móveis, web, corporativas e distribuídas. Ela é indicada para projetos que requerem compatibilidade e escalabilidade, mas não para projetos que requerem rapidez e simplicidade.
- Ruby: É uma linguagem de alto nível e multiparadigma. Ela é muito produtiva e flexível, mas também lenta e inconsistente. Ela é usada para criar aplicações web, scripts e testes automatizados. Ela é indicada para projetos que requerem agilidade e criatividade, mas não para projetos que requerem eficiência e estabilidade.
- Python: É uma linguagem de alto nível e multiparadigma. Ela é muito popular e versátil, mas também lenta e menos otimizada. Ela é usada para criar aplicações web, científicas, educacionais e de inteligência artificial. Ela é indicada para projetos que requerem facilidade de aprendizado e diversidade de bibliotecas, mas não para projetos que requerem velocidade e precisão.
- Haskell: É uma linguagem de alto nível e funcional. Ela é muito elegante e expressiva, mas também complexa e restritiva. Ela é usada para criar aplicações matemáticas, lógicas, concorrentes e paralelas. Ela é indicada para projetos que requerem abstração e correção, mas não para projetos que requerem interação e pragmatismo.
- Lisp: É uma linguagem de alto nível e funcional. Ela é muito poderosa e dinâmica, mas também antiga e esotérica. Ela é usada para criar aplicações de inteligência artificial, processamento de linguagem natural e metaprogramação. Ela é indicada para projetos que requerem inovação e adaptação, mas não para projetos que requerem popularidade e padronização.



## Conclusão

Em conclusão, a programação é uma habilidade que permite criar soluções inovadoras, automatizar tarefas, melhorar processos e se comunicar com as tecnologias digitais. A programação envolve o uso de algoritmos, que são sequências de instruções para resolver problemas ou executar tarefas, e de linguagens de programação, que são formas de comunicação entre humanos e máquinas. Existem vários tipos de linguagens de programação, cada uma com suas características, vantagens e desvantagens. Para escolher uma linguagem de programação para um projeto, é preciso considerar vários fatores, como o objetivo, o mercado, o tempo e o custo.

A programação é uma área em constante evolução e expansão, que oferece diversas oportunidades e desafios para quem se dedica a ela. A programação é responsável por muitas das invenções e transformações que mudam o mundo todos os dias, como a internet, os smartphones, os jogos, as redes sociais, a inteligência artificial e muito mais. A programação também permite expressar a criatividade, a lógica e o aprendizado contínuo.

Por isso, aprender programação é uma forma de se preparar para o futuro e participar da construção de um mundo melhor. A programação pode ser divertida, gratificante e empolgante, além de abrir as portas para diversas carreiras e profissões. A programação é uma linguagem universal que pode conectar pessoas de diferentes culturas, países e interesses.