

LÓGICA DE PROGRAMAÇÃO

SENAI

SUMÁRIO

PARA COMEÇAR	3
SITUAÇÃO-PROBLEMA	4
DESAFIO 1.....	6
O QUE É LÓGICA?	7
LÓGICA DE PROGRAMAÇÃO.....	8
SOFTWARES	9
APLICATIVOS	10
SISTEMAS OPERACIONAIS.....	12
DRIVERS.....	16
ALGORITMOS	18
DESCRÍÇÃO NARRATIVA	20
FLUXOGRAMA.....	20
PSEUDOCÓDIGOS	24
LINGUAGEM DE PROGRAMAÇÃO.....	25
NESTE DESAFIO.....	27
DESAFIO 2.....	28
TÉCNICAS DE PROGRAMAÇÃO.....	28
TIPOS DE DADOS	29
VARIÁVEIS E CONSTANTES.....	32
ESTRUTURAS DE DECISÃO.....	34
ESTRUTURAS DE REPETIÇÃO	40
OPERADORES	44
NESTE DESAFIO.....	46
DESAFIO 3.....	47
LINGUAGENS DE PROGRAMAÇÃO	47
SEMÂNTICA.....	48
SINTAXE	50
INDENTAÇÃO DE CÓDIGO	51
CARACTERÍSTICAS DAS LINGUAGENS	52
TIPOS DE LINGUAGENS	53
NÍVEIS DE LINGUAGENS DE PROGRAMAÇÃO	55
IMPLEMENTAÇÃO.....	56
FERRAMENTAS DE DESENVOLVIMENTO	57
CONVERSÃO DE LINGUAGENS.....	60
APLICAÇÕES DE PROGRAMAÇÃO	61
FERRAMENTAS PARA DESENVOLVIMENTO JAVASCRIPT	63
criando nosso projeto	74
NESTE DESAFIO.....	91
PARA CONCLUIR.....	92
REFERÊNCIAS	93
CRÉDITOS.....	95

PARA COMEÇAR

Esta etapa aborda conceitos relacionados à **Lógica de Programação**

No decorrer de seus estudos, esperamos que você desenvolva as seguintes capacidades:

- Aplicar técnicas de programação na elaboração de algoritmos inerentes aos sistemas de TI.
- Aplicar linguagens de programação para elaborar programas e sistemas de TI.

Para desenvolver tais capacidades, você deverá estudar os seguintes temas:

- Lógica de programação e algoritmos.
- Softwares.
- Técnicas e linguagens de programação.

Esse estudo será necessário para que você resolva a situação-problema a seguir. Então, avance para conhecê-la.

SITUAÇÃO-PROBLEMA

A empresa Savinis, focada no desenvolvimento de *software* de alta performance, contratou você para realizar o desenvolvimento de um sistema de controle de estoque de peça, que será disponibilizado para os funcionários da empresa contratante, o que, até então, era realizado manualmente.

Esse sistema terá como função principal a de controlar a entrada e saída de peças do estoque, além de atender aos seguintes requisitos:

- a empresa possui apenas uma peça a ser controlada;
- o sistema deverá solicitar sempre o saldo inicial da peça uma única vez a cada vez que o sistema é iniciado;
- o sistema deverá ter dois tipos de entrada de dados:
 - » 1 = compra (entrada) de peças
 - » 2 = venda (saída) de peças
- enquanto o usuário não encerrar a entrada de dados, o sistema deverá continuar solicitando nova entrada de dados:
 - » os dados de entradas são: tipo de entrada e quantidade de peças;
 - » caso a entrada seja do tipo 1 (compra), somar a quantidade ao estoque da peça;
 - » caso a entrada seja do tipo 2 (venda), subtrair a quantidade do estoque da peça caso a quantidade de entrada seja inferior ou igual ao saldo de estoque. Caso seja informado uma quantidade maior que o saldo da peça, apresentar a mensagem "Saldo insuficiente" e desconsiderar atualização de saldo;
 - » a cada entrada de dados, apresentar o saldo atualizado do estoque;
 - » ao final de cada entrada de dados, perguntar se o usuário deseja continuar (s) ou não (n) a entrada de dados;
- caso o usuário encerre o sistema, apresentar a mensagem "Sistema encerrado".

Para criação desse sistema de cadastro, você, enquanto programador(a) responsável, deverá solucionar os seguintes desafios:

Desafio 1

Descrever a sequência de passos lógicos necessários para criação do sistema solicitado pela empresa.

Desafio 2

Aplicar técnicas de programação e ferramentas para desenvolvimento do código.

Desafio 3

Aplicar linguagem de programação para implementação do sistema.

DESAFIO 1

Nesta etapa, você deverá resolver o Desafio 1:

- Descrever a sequência de passos lógicos necessários para criação do sistema solicitado pela empresa.

Para isso, você estudará os seguintes conteúdos:

- Lógica de programação e algoritmos;
- Softwares.



O QUE É LÓGICA?



A lógica é o campo de estudo que utiliza princípios e conhecimentos para se atingir um raciocínio correto.

O filósofo Copi (1978) define lógica como:

"estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto."

A utilização do raciocínio lógico é tão comum e natural ao ser humano que nem percebemos.

Pense, por exemplo:

Quais são os passos necessários para passar por uma porta que está trancada?

Para isso, deve-se se executar a seguinte sequência:



1. Colocar a chave na fechadura.
2. Girar a chave para destrancar a porta.
3. Girar a maçaneta.
4. Puxar a porta para abri-la.

A ação de abrir uma porta trancada parece simples, mas envolve uma sequência lógica de passos. Se você tentar girar a maçaneta, antes de destrancar a porta com a chave, o resultado não será o mesmo.

Podemos citar várias outras situações cotidianas que envolvem raciocínio lógico, como tomar banho, cozinhar e dirigir um carro.

Você sabia?

O filósofo grego Aristóteles (384–322 a.C.) foi quem iniciou os estudos da Lógica. O conjunto de sua obra é conhecida como "lógica aristotélica" ou "lógica clássica".



No século XIX, o filósofo alemão Gottlob Frege desenvolveu um método chamado de cálculo de predicados, que analisa proposições linguísticas por meio de processos dedutivos matemáticos, contribuindo para criação dos códigos de programação de computadores.

LÓGICA DE PROGRAMAÇÃO

A lógica de programação surgiu a partir dos princípios da lógica e consiste em uma técnica de encadeamento do pensamento para atingir um determinado objetivo ou solucionar um problema.

Podemos dizer que a lógica é a primeira etapa da programação em si. Isso quer dizer que, antes de começar a escrever o código, você deve pensar quais as questões que devem ser resolvidas, estudar quais as soluções possíveis e planejar todas as etapas da solução.

Para isso, é necessário utilizar uma sequência lógica, que é um conjunto de passos a serem executados.

Quando falamos em sequência lógica, estamos falando de algoritmos.

Saiba que uma das ações mais comuns de um(a) programador(a) é buscar as melhores soluções e encontrar os algoritmos mais adequados para a criação de programas de computadores, soluções e serviços.

SOFTWARES

Você viu que um software é um agrupamento de instruções ou comandos escritos em uma linguagem de programação que são lidos pelo computador e possibilitam seu funcionamento.

Em outras palavras, software é um produto virtual usado para descrever programas, aplicativos, scripts, macros e instruções de código embarcado diretamente (*firmware*), a fim de determinar o que uma máquina deve fazer.



Software embarcado ou *firmware* é um conjunto de instruções operacionais programadas pelo fabricante diretamente no hardware do equipamento, para manter a configuração básica das funções.

Isso significa que os códigos transcritos por esse tipo de programa são fundamentais para iniciar e executar os hardwares e seus recursos, fornecendo informações idênticas sempre que o dispositivo for ligado.

E você sabe dizer qual é a diferença entre **hardware** e **software**?

Hardware é a parte física do computador, ou seja, o conjunto de aparelhos eletrônicos, peças e equipamentos que fazem o computador funcionar. Monitor, placa de vídeo, processador, mouse, disco rígido e teclado são exemplos de hardware.

Software é a parte lógica do computador, ou seja, são os programas que fazem com que a máquina funcione, como aplicativos e sistemas operacionais, desenvolvidos por meio de códigos e linguagem de programação.

TIPOS DE SOFTWARES

Os softwares são divididos em três principais categorias, são elas:

Programação:

São softwares que, a partir de linguagens de programação, como Java, Python, Swift, são utilizados para o desenvolvimento de outros programas.

Sistema:

Softwares de sistema ou de base permitem a execução de outros softwares, como os de aplicação e os de programação. São responsáveis pela comunicação entre o computador, que só entende linguagem de máquina, e o usuário, ou seja, interpreta nossas ações e as transforma em códigos binários. Sistemas operacionais, como Windows, MacOS, Linux, iOS, Android são exemplos de softwares de sistema.

Aplicação:

Os softwares de aplicação realizam ações específicas solicitadas pelos usuários. Trata-se de programas que são executados dentro do sistema operacional. Word, Excel, paint, bloco de notas, calculadora e jogos são exemplos deste tipo de software.

APLICATIVOS

Aplicativos ou App são softwares presentes em dispositivos móveis, como smartphones, tablets, smarts TVs, relógios inteligentes, entre outros, que desempenham vários tipos de tarefas.

Apresentam interface amigável e simples para facilitar a interação com o usuário. Alguns já vêm instalados e outros podem ser adquiridos em lojas de aplicativos e podem, ainda, apresentar versões gratuitas ou pagas.

WhatsApp, Uber e Ifood são alguns exemplos desses aplicativos.



Você sabia?

Qual a diferença entre um celular e um smartphone?



O smartphone possui sistema operacional, como os computadores, e o celular comum não possui. O celular realiza tarefas mais simples, como digitar e enviar mensagens. Já o smartphone, além realizar essas tarefas, possui várias outras funções, como acesso a internet, jogos, loja de aplicativos etc.

TIPOS DE APLICATIVOS

Dependendo de como serão utilizados, o desenvolvimento de um aplicativo requer recursos e tecnologias específicas.

Atualmente, existem basicamente 3 tipos de aplicativos móveis: **nativos**, **web** e **híbridos**.

Nativos

São desenvolvidos para um determinado sistema operacional, como Android e iOS. Devem ser programados na linguagem do seu respectivo sistema, como Java e Kotlin no Android e Objective-C e Swift no iOS – mas há também outras linguagens para outros tipos de sistemas operacionais.

Como são programados exclusivamente para um determinado sistema operacional, podem funcionar sem conexão com internet, acessam todos os sensores do dispositivo (câmeras, GPS etc.), são mais rápidos e apresentam melhor experiência para o usuário, o qual consegue acessar todos os seus recursos.

Nesse tipo de aplicação, os programadores utilizam o **Ambiente de Desenvolvimento Integrado** (IDE - *Integrated Development Environment*, em inglês). Trata-se de um pacote de software que consolida as ferramentas básicas necessárias para escrever e testar software.

Web Apps

São desenvolvidos para serem abertos no navegador do smartphone. Trata-se de uma programação que reconhece que o usuário está acessando de um dispositivo móvel e, por isso, o layout adapta-se a ele. Diferentemente dos aplicativos nativos, os Web Apps necessitam de acesso à internet, não conseguem utilizar todas as funcionalidades do dispositivo e são mais lentos. Sua programação é feita utilizando a linguagem HTML5, o Cascading Style Sheets (CSS) e o JavaScript.

Híbridos

São uma mistura de aplicativo nativo e Web App desenvolvido para ser reconhecido por qualquer sistema operacional. Também são programados por meio da linguagem HTML5, CSS e JavaScript, assim como o site mobile, mas seu código é alocado dentro de um container, integrando as funcionalidades do dispositivo móvel, o que proporciona uma experiência melhor ao usuário do que os Web Apps. Podem ser baixados nas lojas de aplicativos e o acesso à internet pode ou não ser necessário. O objetivo desses aplicativos é apresentar uma única versão para vários sistemas.

SISTEMAS OPERACIONAIS

Um sistema operacional, também chamado de SO, é um tipo de software de sistema que controla praticamente todos os processos de um computador.

Em outras palavras, sistema operacional é o software responsável por fazer a ponte, a interface entre o usuário e o hardware, gerenciando recursos do sistema e do hardware.

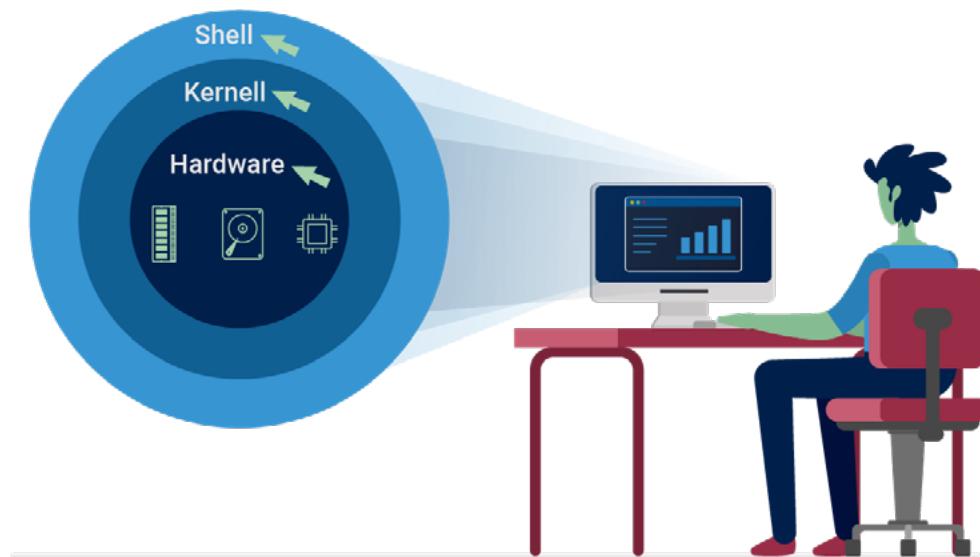


Algumas funções básicas dos sistemas operacionais são:

- controlar acesso ao hardware;
- prover interface ao usuário;
- gerenciar programas;
- gerenciar arquivos e pastas.

O usuário interage com a interface do SO, mas não precisa gerenciar os recursos.

Podem-se destacar três estruturas fundamentais para o funcionamento adequado do computador: **shell**, **kernell** e **hardware**.



SHELL

É a interface de interação com o usuário, ou seja, onde ele realiza a inserção de dados para solicitar as tarefas e ações desejadas.

Há dois tipos de shell:

- **CLI (Comand Line Interface)** – interface de linha de comando, também chamada de modo texto.
- **GUI (Graphical User Interface)** – interface gráfica do usuário, também chamada de modo gráfico.

KERNELL

Realiza o papel de interação entre o hardware e o software.

HARDWARE

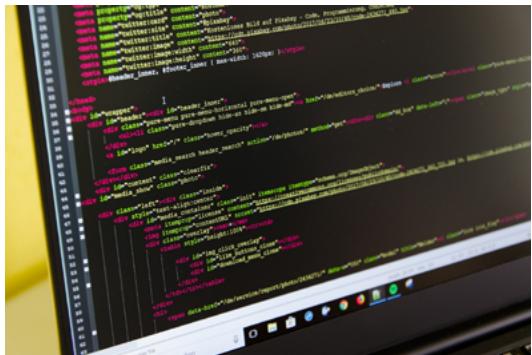
São os elementos físicos, como os componentes eletrônicos.

CARACTERÍSTICAS ENTRE AS INTERFACES GUI E CLI

GUI

Características da interface gráfica do usuário:

- Exibição de tela em um monitor.
- Utilização do mouse para seleção e execução de programas.
- Utilização do teclado para inserção de dados.



Fonte: pixabay.com

CLI

Características da interface por linha de comando:

- Exibição de tela em um monitor.
- Utilização do teclado para inserção de textos e comandos.
- Utilização do teclado para execução de programas baseados em CLI.

CATEGORIAS DE SO

O sistema operacional pode ser classificado em duas categorias: para desktops e para servidores.



Sistemas operacionais para desktops

Os sistemas operacionais para desktop são amplamente utilizados para ambientes de pequeno porte, conhecidos como SOHO (*Small Office and Home Office* – Escritórios de Pequeno Porte e Domésticos), nos quais há um número pequeno de estações de trabalho que não requerem um servidor para serviços centralizados.

Confira algumas características do SO para desktop:

- É compatível apenas com um único usuário.
- Executa aplicações de um único usuário.
- Compartilha arquivos e pastas em uma rede de pequeno porte, com segurança limitada.

Sistemas operacionais para servidores

Os sistemas operacionais para servidores, também conhecidos por NOS (*Networking Operational System* – Sistema Operacional de Rede), proporcionam diversas funcionalidades e aprimoramentos para que seja possível gerenciar um ambiente de sistemas operacionais desktops em uma rede. Com esses sistemas, é possível, por exemplo, centralizar e controlar as ações permitidas de um usuário inserido em um domínio e disponibilizar diversos serviços de rede, como atribuição dinâmica de endereçamento IP, resolução de domínios, compartilhamento de arquivos, filas de impressão, entre outros.

Confira algumas características do SO para servidores:

- É compatível com vários usuários.
- Executa aplicações multiusuários.
- Fornece mais segurança, se comparado ao SO para desktops.

Os sistemas operacionais para desktops mais comuns são: Windows (Microsoft), Linux (é de código aberto), MacOS (Apple).

Windows (Microsoft)

O Windows da Microsoft surgiu no começo da década de 1980 e teve diversas versões ao longo dos anos.

Intuitivo, de interface gráfica simples e amigável, com uma suíte de softwares de aplicação ampla e compatível com computadores de diversos fabricantes, o Windows se tornou o SO para desktop residencial mais popular, tanto que a maioria dos computadores domésticos já vem com ele instalado.

A Microsoft também oferece uma versão Windows para servidores.

MacOS (Apple)

O macOS (anteriormente Mac OS X e posteriormente OS X) da Apple também surgiu na década de 1980 e é o sistema operacional de todos os computadores Apple. Um grande impedimento para a popularização é a falta de compatibilidade com dispositivos de outras marcas.

A Apple também oferece uma versão para servidores, chamada macOS Server (anteriormente Mac OS X Server e OS X Server).

Linux

É um sistema operacional de código aberto, o que significa que pode ser modificado e distribuído por qualquer pessoa. Apesar de gratuito, não é muito usado em computadores domésticos porque exige um certo grau de conhecimento para instalação. Em contrapartida, o Linux é hegemonic entre os servidores de empresas, pois é fácil de personalizar.

As versões mais populares são Ubuntu, Debian, Linux Mint e Fedora.

DRIVERS

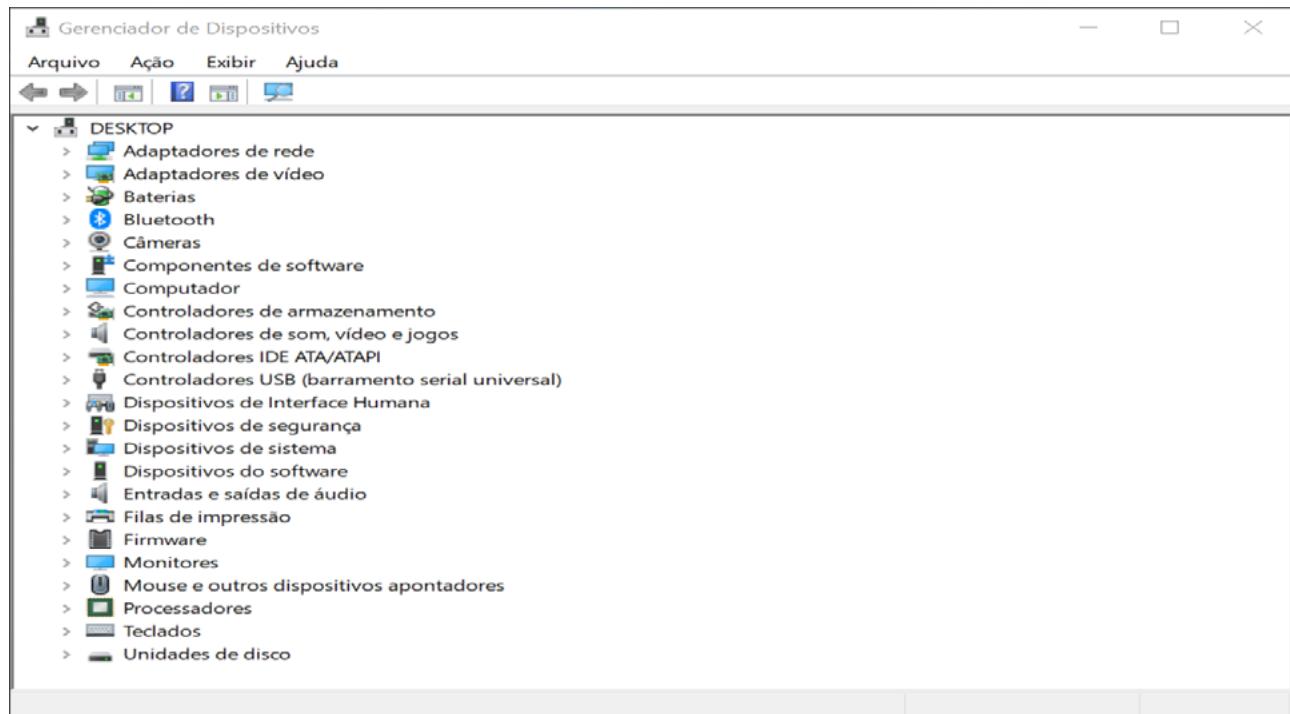
Os drivers são softwares ou pequenos elementos de softwares que ligam o SO aos dispositivos físicos do computador.

Os drivers são responsáveis pela comunicação entre os sistemas operacionais e um componente de hardware.



Então, se você conectar um dispositivo, como uma impressora ou uma placa de vídeo, e não instalar o driver apropriado, o computador pode não reconhecer aquele componente que foi conectado, e não poderá utilizá-lo.

Você pode acessar os drivers de seu computador por meio do gerenciador de dispositivos. A imagem a seguir mostra o gerenciador de dispositivos de um SO Windows.



Importante!

Compreender as diferenças entre hardware, software e driver e os conceitos apresentados irão permitir a você, programador ou programadora, o melhor gerenciamento dos recursos e da manipulação dos itens necessários para você desenvolver o seu projeto. Isso fará com que você compreenda, ao desenvolver um código lógico, como o seu computador estará processando e compartilhando essa informação.

Ao desenvolver um projeto prático, utilizar recursos de maneira errada pode prejudicar a performance do seu software e do seu hardware.



ALGORITMOS

Conforme você viu, o desenvolvimento de qualquer software é realizado por meio dos algoritmos, descrevendo uma sequência de passos lógicos necessários para a execução de uma tarefa que consiste em:



É importante salientar que uma tarefa pode ser composta de várias pequenas ações e cada uma dessas ações tem seu próprio conjunto de instruções ou algoritmo a ser seguido.

Para exemplificar, considere o seguinte problema:

NOTAS

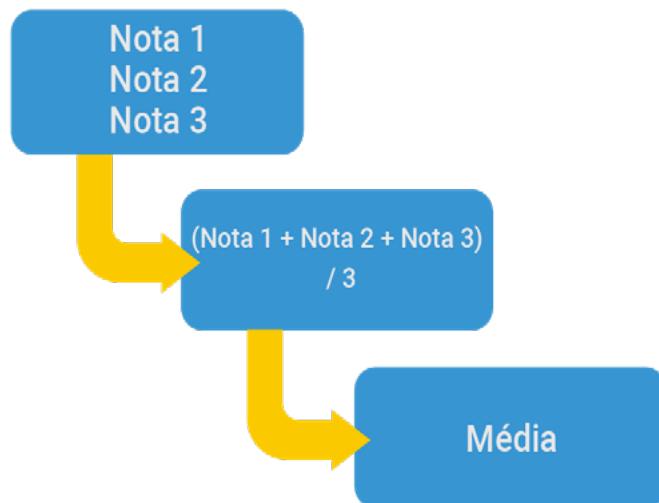
Você precisa criar um algoritmo que, a partir da soma de 3 notas semestrais de um aluno, calcule sua média final para, na sequência, informar se o aluno foi reprovado ou aprovado, considerando as seguintes referências:

- de 7 a 10 – aprovado.
- abaixo de 7 – reprovado.

Para resolução desse problema, serão necessários dois algoritmos, sendo que cada um resolverá um pequeno problema da situação.

1º ALGORITMO

Calculará a média do aluno. Ele deve possuir como entrada as 3 notas obtidas durante o semestre. O processamento deverá realizar o cálculo da média, somando as notas e dividindo o resultado por 3. Por fim, a saída (ou resultado) será a média obtida pelo aluno.



2º ALGORITMO

Verificará a situação final do aluno. Para isso, a entrada deste algoritmo será a média obtida com o primeiro algoritmo. O processamento será a verificação dessa média, para descobrir se foi maior ou igual a 7 ou menor que 7. Com essa verificação, é possível definir a saída deste algoritmo, informando se o aluno foi aprovado ou não.



E COMO REALIZAR A REPRESENTAÇÃO DOS ALGORITMOS?

A representação dos algoritmos pode ser realizada por meio de **descrição narrativa, fluxograma e pseudocódigos**.

Siga em frente para conhecer cada uma delas.

DESCRÍÇÃO NARRATIVA

Utiliza palavras para expressar os algoritmos e, por isso, é exclusivamente usada para fins didáticos. O algoritmo apresentado para fazer café é um exemplo desse tipo de representação.

O exemplo abaixo mostra o algoritmo do problema sobre notas, representado por meio da descrição narrativa:



1. Obter as 3 notas do aluno.
2. Somar as 3 notas.
3. Dividir o resultado por 3.
4. Se a média for maior ou igual a 7 = aluno foi aprovado.
5. Se a média for menor que 7 = aluno foi reprovado.

FLUXOGRAMA

Fluxograma é um tipo de diagrama ou uma representação gráfica que descreve as diferentes ações a serem realizadas durante a execução de um algoritmo.

Ele auxilia na elaboração do raciocínio lógico a ser seguido para a resolução de um problema, mostrando visualmente como o nosso código deve se comportar nas diversas situações e as diferentes saídas que ele terá dentro do nosso programa.

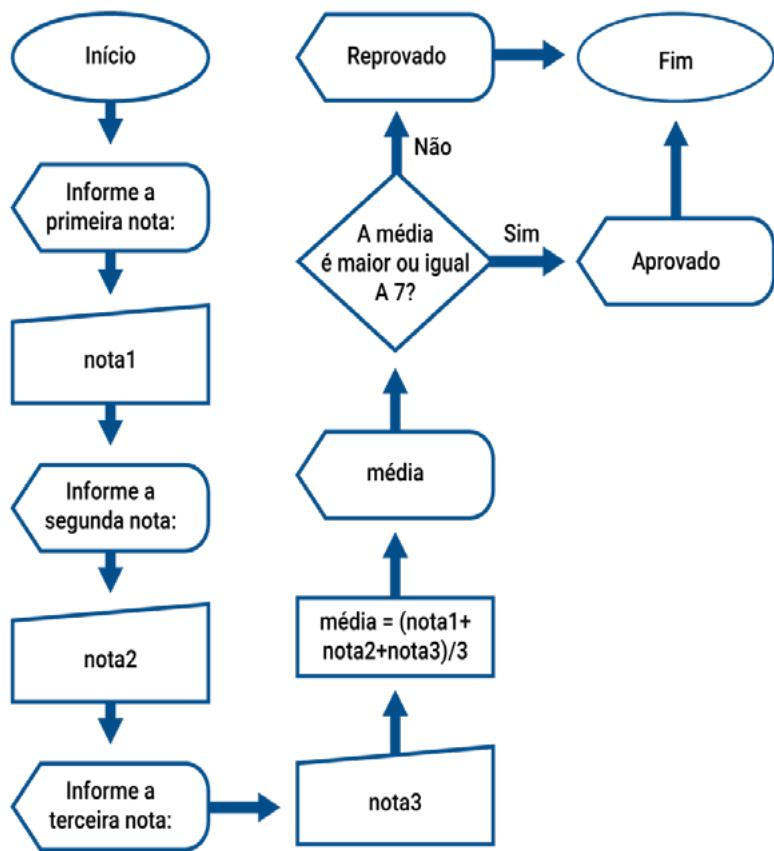
Por exemplo: não temos a água para realizar o café, o que fazer?

Dessa forma, usando as linguagens de programação, escreveremos ações para que nosso código tenha o comportamento específico, exatamente como exibido no fluxograma, e seja concluído com sucesso.

A tabela mostra as simbologias mais comuns utilizadas para representação de algoritmos por meio de fluxogramas:

Representação gráfica	Item	Descrição
	Início/fim	Todo fluxograma deve iniciar e encerrar com este símbolo. O fluxograma deve conter apenas um início; porém, poderá possuir mais de um fim, pois pode se dividir durante o processo.
	Leitura	Representa uma entrada do usuário, quando o programa fará uma leitura de uma informação digitada pelo usuário.
	Escrita	Representa a impressão de alguma informação na tela pelo programa com o objetivo de informar o usuário.
	Seta de fluxo	Representa o caminho do fluxograma. A partir do início, a leitura do fluxograma é feita seguindo as setas do fluxo.
	Processo	Utilizado quando algo deve ser processado pelo programa, por exemplo, para cálculos matemáticos.
	Decisão	Divide a execução do fluxograma em dois caminhos. Sempre que for utilizado, uma pergunta deve ser feita. Caso a resposta seja verdadeira, o fluxograma segue por um caminho, caso contrário, segue por outro. Essa é a única situação em que devem ser utilizadas duas setas de fluxo a partir de uma figura geométrica.

O fluxograma a seguir mostra a representação do algoritmo do problema sobre notas:



Perceba que o fluxograma mostra o passo a passo de cada ação. Primeiro, solicita-se as 3 notas, utilizando um nome genérico, como notas 1, 2 e 3. Depois, é realizado um processo para calcular a média [$\text{média} = (\text{nota 1} + \text{nota 2} + \text{nota 3}) / 3$], cujo resultado fornece a média do aluno.

Além de calcular a média, o fluxograma também apresentará se o aluno foi reprovado ou aprovado, considerando a média 7 para aprovação.

TESTANDO NA PRÁTICA

Para testar a execução e a eficácia do fluxograma, simule o passo a passo e identifique se o resultado final está correto ou não, utilizando os seguintes valores de notas:

- Nota 1: 7
- Nota 2: 8
- Nota 3: 8

Dica!

Há vários programas gratuitos na internet, com opções online e offline para o desenvolvimento de fluxogramas. Conheça alguns:



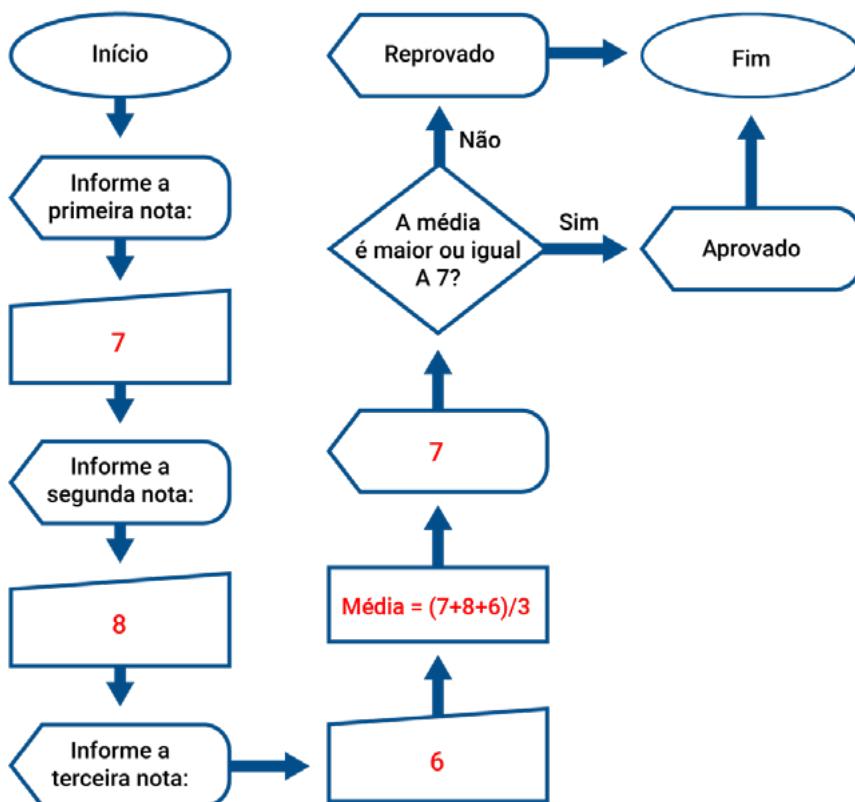
Acesse o link do **LucidChart** : <https://www.lucidchart.com/pages/>

Acesse o link do **Draw.io** : <https://app.diagrams.net/>

E então? Qual foi a média desse aluno? Ele foi aprovado ou reprovado?

Se você encontrou a média 7, parabéns você acertou! O aluno foi aprovado.

Caso você tenha encontrado outro valor, refaça os cálculos e consulte o fluxograma abaixo.



PSEUDOCÓDIGOS

É uma forma de representação de algoritmo semelhante à linguagem de programação, porém utilizando palavras no idioma escolhido.

O pseudocódigo apresenta 3 etapas:

1. Identificação do algoritmo.
2. Declaração das variantes.
3. Corpo do algoritmo.



A seguir, veja o exemplo do cálculo da média representado pelo pseudocódigo:

1. **algoritmo** "CalcularMédia"
2. **var**
3. nota1, nota2, nota3, media: real
4. **início**
5. **escrever** ("**digite** a primeira nota:")
6. **ler** (nota1)
7. escrever ("digite a segunda nota:")
8. **ler** (nota2)
9. escrever ("digite a terceira nota:")
10. **ler** (nota3)
11. media<-(nota1+nota2+nota3) /3
12. escrever (media)
13. se media>=7 então
14. escrever ("Aprovado")
15. senão
16. escrever ("Reprovado")
17. fimse
18. **fim**

Linha 1: Identificação do algoritmo.

Linhas 2 e 3: declaração das variantes, ou seja, notas 1, 2 e 3.

Linhas 4 a 18: corpo do algoritmo: notas, cálculo da média, decisão e verificação.

ENTENDENDO ALGUNS TERMOS

Você percebeu que foram utilizados alguns termos e símbolos específicos para representação dos algoritmos em pseudocódigos. Conheça melhor cada um deles:

Algoritmo

Comando que define o nome do programa. Deve vir entre aspas duplas.

Var

Comando que especifica a área em que as variáveis serão declaradas.

Início

Comando que informa o início do programa. É nesse bloco que ficarão os comandos e a lógica que será utilizada para criar o algoritmo.

Escrever

Comando que escreve na tela alguma informação ao usuário.

Ler

Comando que lê o que foi digitado pelo usuário.

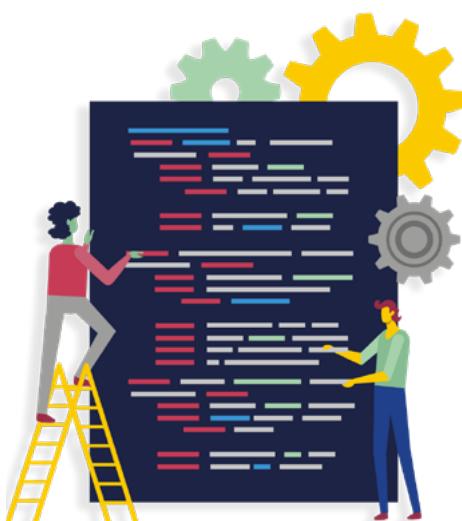
Fim

Comando que informa que é o final do algoritmo.

LINGUAGEM DE PROGRAMAÇÃO

É um método padronizado, formado por um conjunto de regras sintáticas e semânticas, símbolos e códigos que torna possível a execução do algoritmo.

É por meio dessa linguagem de programação que o programador se comunica com a máquina, para um site fazê-lo funcionar, desenvolver softwares, aplicativos para sistemas operacionais, jogos etc.



Existem vários tipos de linguagem de programação e você estudará mais sobre eles no Desafio 3, pois, antes, é necessário que você conheça algumas técnicas de programação comuns em todas as linguagens.

Por ora, conheça como o exemplo do cálculo da média ficou após implementado na linguagem JavaScript.

```
1. <script>
2.
3.     n1 = parseFloat(prompt("digite a primeira nota"));
4.     n2 = parseFloat(prompt("digite a segunda nota"));
5.     n3 = parseFloat(prompt("digite a terceira nota"));
6.     media = (n1 + n2 + n3) / 3;
7.
8.     if(media >= 7)
9.         alert("Aprovado");
10.    else
11.        alert("Reprovado");
12.
13. </script>
```

NESTE DESAFIO...

LÓGICA DE PROGRAMAÇÃO| DESAFIO 1



Você percebeu que a lógica de programação usa o raciocínio lógico de forma estruturada e planejada na hora de escrever os códigos, por meio de algoritmos.

Algoritmo é a base de todo o sistema. Consiste em um conjunto de tarefas que devem ser realizadas de maneira sequencial e planejada para resolver algum problema, criar produtos e serviços. Ele pode ser representado por descrição

narrativa, fluxograma, pseudocódigos e são executados por meio das linguagens de programação. Todos os tipos de softwares são basicamente algoritmos.

Falando em softwares, você estudou o quanto importante é entender alguns conceitos básicos, como hardwares, aplicativos, sistemas operacionais e drivers, para utilizar os recursos de maneira assertiva.

NO PRÓXIMO DESAFIO...

Vamos conhecer outros tópicos importantes dentro das técnicas de programação.

DESAFIO 2

No Desafio 1, você estudou como descrever e representar a sequência de passos lógicos ou algoritmos necessários para a execução de uma tarefa em qualquer sistema, seja um software, aplicativo, site etc.

Agora, nesta etapa, você irá mergulhar na parte computacional e conhecerá os recursos e comandos para resolver o Desafio 2:

- Aplicar técnicas de programação e ferramentas para desenvolvimento do código.

Para isso, você estudará os seguintes conteúdos:

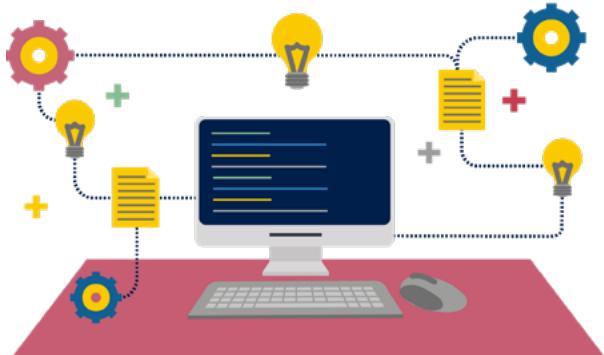
- Tipos e estrutura de dados;
- Variáveis e constantes;
- Estrutura de decisão;
- Estrutura de repetição;
- Operadores aritméticos, lógicos e relacionais.

TÉCNICAS DE PROGRAMAÇÃO

Agora que já sabemos o que é um algoritmo, precisamos conhecer algumas técnicas de programação.

Técnicas de programação são conceitos, recursos e boas práticas para o desenvolvimento de programas de computadores.

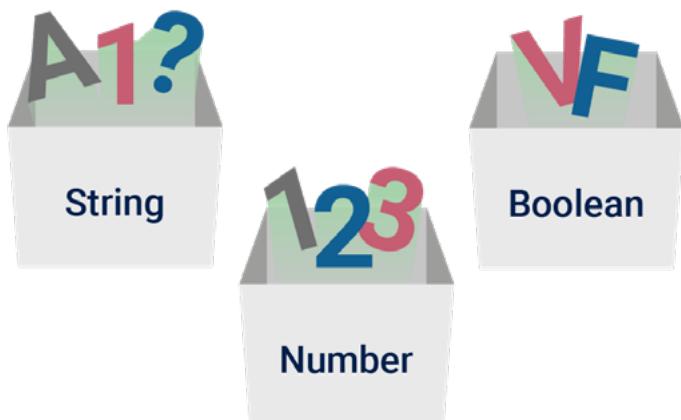
Com as técnicas de programação, como sintaxe e recursos específicos ou comuns entre as linguagens de programação, podemos criar as soluções para os nossos projetos, produtos e/ou serviços.



As técnicas de programação que abordaremos agora são:

- tipos e estrutura de dados;
- variáveis e constantes;
- fluxograma;
- estrutura de decisão;
- estrutura de repetição;
- operadores.

TIPOS DE DADOS



Os dados são todas as informações relevantes para a programação que devem ser armazenadas, pois serão usadas durante a execução do algoritmo ou programa.

Os dados podem ser de vários tipos, como caracteres ou números. Isso varia de acordo com o sistema operacional e a linguagem de programação usada.

Vamos conhecer três tipos de dados comuns para exemplificar.

string

O dado do tipo "***string***" armazena **caracteres** e é muito comum nas linguagens de programação. Porém, há diferenças: algumas linguagens de programação são case-sensitive, isto é, diferenciam caracteres maiúsculos de minúsculos, como a linguagem C, o JavaScript e o PHP. Outras são case insensitive, como Pascal e Delphi, que não diferenciam maiúsculas de minúsculas.

number

Para armazenar números, o JavaScript, por exemplo, usa o dado do tipo "**number**" para números menores que 253 e do tipo "**BigInt**" para números maiores que 253. No caso do PHP, o dado do tipo "**int**" armazena os números naturais e seus simétricos negativos, e o dado do tipo "**float**" é usado para números reais.

boolean

O dado do tipo "**boolean**" armazena apenas os valores **true** ou **false** (verdadeiro ou falso) e está presente em quase todas as linguagens de programação.

ESTRUTURA DE DADOS

Estruturas de dados definem a organização e suas operações (métodos de acesso) sobre um determinado conjunto de informações. Essa organização permite um melhor processamento e é usada para grandes volumes de dados.

Por exemplo: você deseja armazenar o nome de todos os alunos da sua turma. O sistema deve definir a organização (como salvar) e, consequentemente, o acesso ao conjunto de operações sobre esses dados (inserir ou remover um aluno, editar um nome ou trocar a posição de um dado).

Há vários tipos de estrutura de dados.

Lista (list)

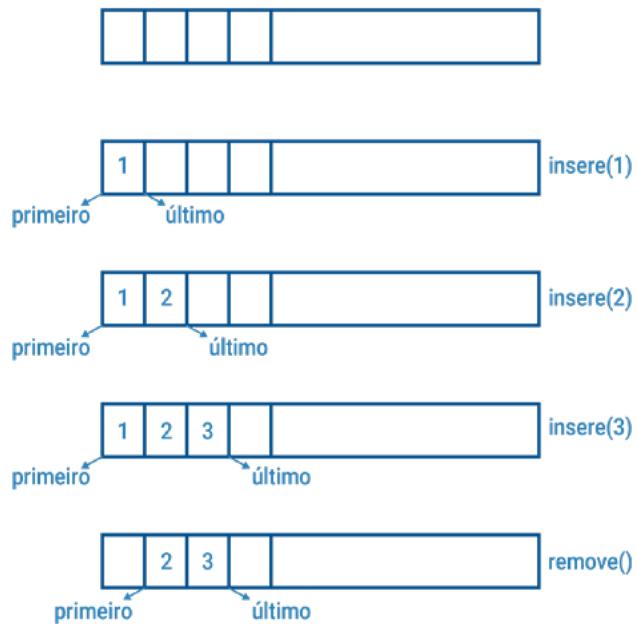
Uma lista armazena dados do mesmo tipo em sequência. Os dados não precisam estar necessariamente armazenados em sequência física na memória do computador, mas há uma sequência lógica entre eles. Cada elemento da lista é chamado de nó.

Em uma lista sequencial ou contígua, os nós estão em sequência lógica e física. Em uma lista encadeada, não há a necessidade da sequência física, apenas a sequência lógica deve ser mantida e o último nó é uma célula nula.

As listas são subdivididas em filas, pilhas e vetores, dependendo do acesso aos nós.

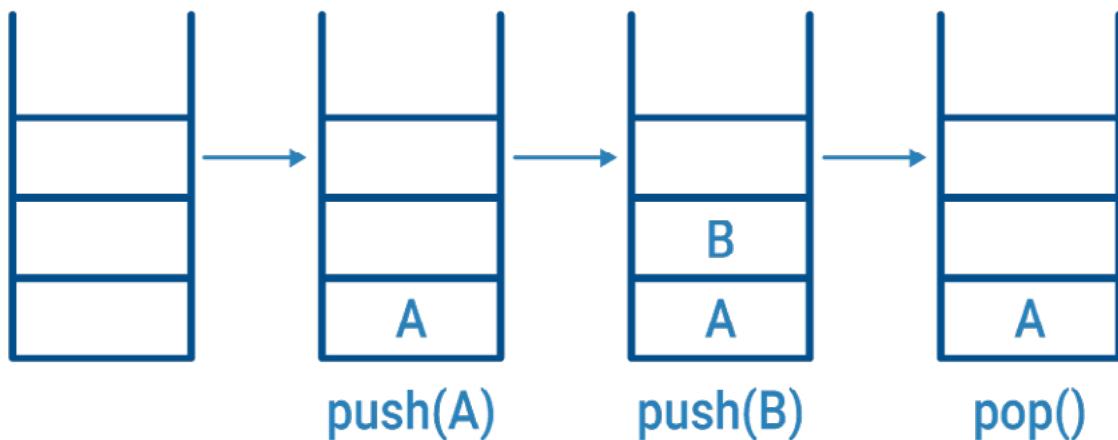
Fila (queue)

A fila é uma estrutura do tipo *first in first out* (FIFO), na qual o primeiro elemento a ser inserido será o primeiro a ser retirado. Assim como em uma fila de pessoas, o primeiro que chegou no balcão, será o primeiro a ser atendido, e assim por diante.



Pilha (stacks)

A pilha é uma estrutura do tipo *last in first out* (LIFO). O último elemento a ser inserido será o primeiro elemento a ser retirado. Podemos ter acesso ao topo dessa pilha, inserir um novo item, remover o item corrente. É como uma pilha de pratos: colocamos ou retiramos pratos no topo da pilha.



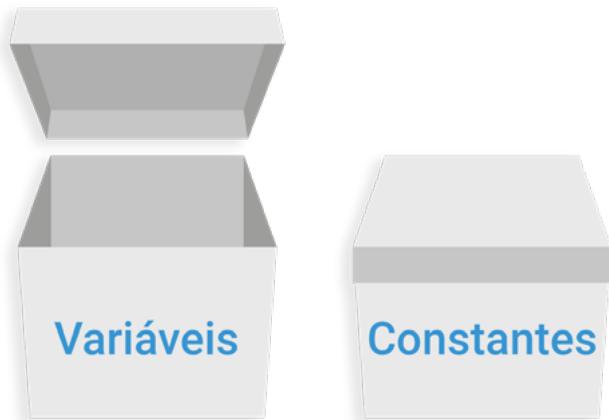
Vetor (array ou matriz)

Um vetor é uma estrutura que armazena uma sequência de objetos do mesmo tipo, em posição consecutiva lógica e física, sendo possível alcançar qualquer item diretamente. Não é necessário passar pelo primeiro ou último elemento.

VARIÁVEIS E CONSTANTES

O valor dos tipos de dados é salvo em uma variável, ou seja, em um espaço de memória do computador.

As variáveis devem ser declaradas com nome, valor e tipo. O nome é atribuído ao declarar a variável. O valor pode ser atribuído ou calculado. O tipo depende do valor e da linguagem da programação usada.



A sintaxe dos comandos, inclusive o da declaração de variáveis, também depende da linguagem de programação usada.

Por exemplo: eu desejo armazenar o valor "Helena" em uma variável de nome "**nomeDoEstudante**". Como o valor é formado de caracteres, o tipo da variável deve ser *string*. No quadro a seguir, temos dois exemplos de declaração de variáveis em duas linguagens de programação diferentes (em preto). O texto em cinza claro são comentários (indicados pelo sinal //) e não fazem parte do comando.

1. // linguagem X
2. var nomeDoEstudante: String = "Helena";
3. // var é a declaração de variável
4. // nomeDoEstudante é o nome atribuído
5. // string é o tipo de dado, pois é formado de caracteres
6. // Helena é o valor
- 7.
8. //linguagem Y
9. \$nomedoestudante = 'Helena'

10. // \$ é a declaração de variável
11. // nomedoestudante é o nome atribuído
12. // atribui o tipo de acordo com o valor
13. // Helena é o valor

Durante a execução do algoritmo ou programa, o valor da variável pode mudar. Em um programa que calcula a idade atual de uma pessoa, por exemplo, a variável que guarda o valor da data atual mudará todos os dias.

Caso o valor de uma variável não possa ser alterado, indicamos que ela é uma constante. Como a data de nascimento de uma pessoa, no caso do programa que calcula a idade. Assim como a variável, a declaração da constante varia de acordo com a linguagem de programação usada.

APLICANDO ALGORITMOS

As informações que fornecemos ao computador são os dados de entrada da nossa aplicação, um processamento é feito com base na informação fornecida, e a saída é o resultado a partir do processamento.



Os valores dos dados de entrada e saída são armazenados em variáveis ou constantes. O processamento é feito através da aplicação ou programa de computador, baseado nos algoritmos da programação.

E a partir dessas informações armazenadas, podemos criar condicionais em nosso código de processamento para seguirmos ações distintas em nossa programação.

E, assim como na declaração de variáveis e constantes, a sintaxe para cada condicional depende da linguagem de programação escolhida. Apesar das especificidades de cada linguagem, a lógica de programação é a mesma, ou seja, o planejamento macro do algoritmo é o mesmo, mas a escrita dos comandos é diferente.

PENSE NISSO...

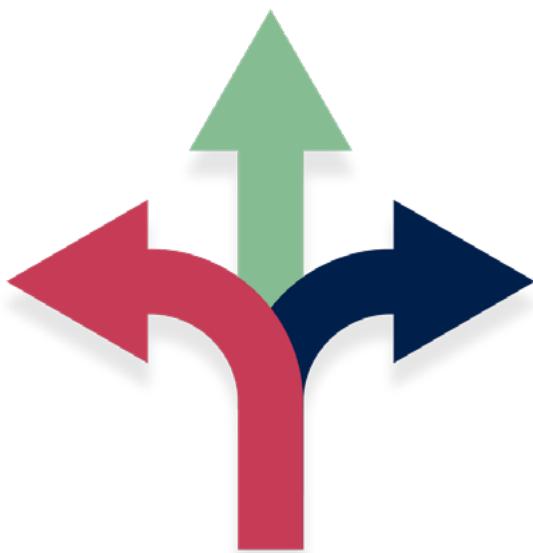
Imagine que uma concessionária vai realizar um evento de test drive e o cadastro de participantes é feito através do site. Independentemente da linguagem de programação escolhida para o sistema de cadastro, todas devem atender ao requisito de só aceitar participantes com habilitação permanente de motorista e, consequentemente, maiores de 18 anos.

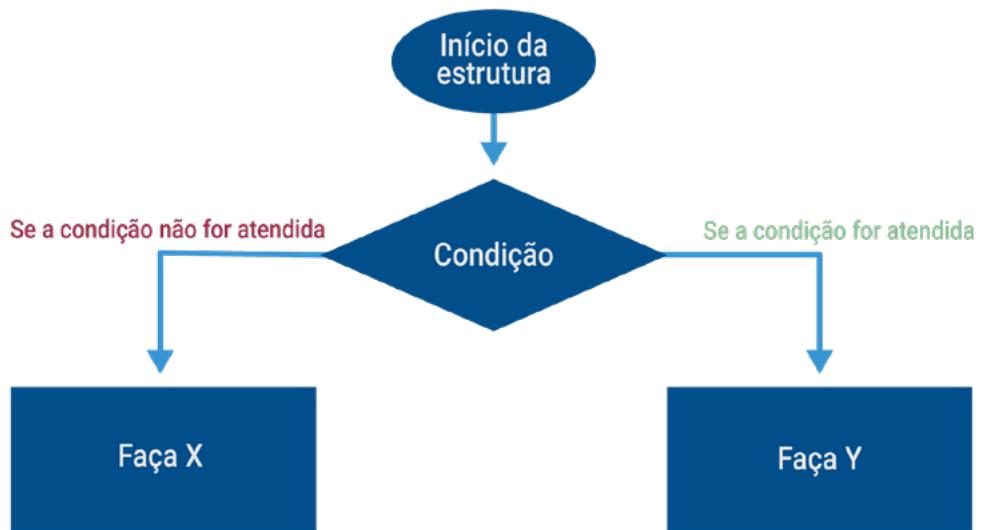
E se um menor de idade tentar fazer o cadastro? Como fazer essa validação? Vamos aprender?

ESTRUTURAS DE DECISÃO

As estruturas de decisão (também conhecidas como estruturas condicionais) usam comandos de decisão para testar uma ou mais condições e especificam uma instrução (ou um conjunto de instruções) se o resultado do teste for verdadeiro, e outra instrução (ou conjunto de instruções) caso o resultado do teste seja falso.

Em outras palavras, a estrutura de decisão é baseada em uma condição: se a condição for atendida, o programa segue um caminho e, se a condição não for atendida, o programa segue outro caminho. O fluxograma é a representação gráfica de uma estrutura de decisão.





COMANDOS DE DECISÃO

A grosso modo, um comando de decisão especifica uma condição a ser testada e indica uma instrução caso a condição seja atendida e outra instrução caso a condição não seja atendida. A sintaxe do comando varia de acordo com a linguagem de programação usada.

Vamos abordar os comandos mais comuns, que são:

- SE (if)
- SE/SE NÃO (if/else)
- SE aninhado (else if)
- CASO (switch)

COMANDO SE (IF)

O mais simples dos comandos de decisão aparece nas linguagens de programação como `if` e pode ser traduzido como "SE". Note que aqui só há instrução caso a condição seja atendida, então não há ação específica e o restante do código é executado.

Código genérico

`if (condição) comando`

Texto

SE (senha correta) exibir "bem-vindo"

Linguagem X

1. Var senha = a1234
2. if (senha = a1234) console.log "bem-vindo"
3. //console.log é um comando para exibir uma mensagem

COMANDO SE/SE NÃO (IF/ELSE)

Aqui, o comando "SE" vem acompanhado de uma instrução caso a condição não seja atendida, como uma bifurcação no caminho.

Código genérico

```
if (condição) comando  
else comando
```

Texto

SE (senha correta) exibir "bem-vindo"

SE NÃO exibir "senha incorreta"

Linguagem X

1. Var senha = a1234
2. if (senha = a1234) console.log "bem-vindo"
3. else console.log "senha incorreta"

COMANDO SE ANINHADO (IF/ELSE IF/ELSE)

E caso nenhuma das duas condições seja atendida, podemos utilizar o comando "SE" aninhado.

Código genérico

```
if (condição) comando  
else if (condição) comando  
else comando
```

Texto

SE (login correto) exibir "digite a senha"
 E SE (senha correta) exibir "bem-vindo"
 SE NÃO exibir "login ou senha incorretos"

Linguagem X

1. Var login = fulano
2. Var senha = a1234
3. if (login = fulano) console.log "digite a senha"
4. else if (senha = a1234) console.log "bem-vindo"
5. else console.log "login ou senha incorretos"

Importante!

Note que o código só vai verificar o segundo *if* se o primeiro for atendido, ou seja, só vai verificar se a senha está correta, se o login estiver correto. Se o login estiver incorreto, o código executará o *else*.



Se a estrutura fosse criada com dois *if* e um *else*, cada condição iria gerar uma mensagem na tela, pois a segunda condição seria verificada de qualquer maneira.



COMANDO CASO (SWITCH)

Este comando é muito utilizado quando uma quantidade maior de condições é necessária para ser utilizada em sua aplicação, quando há uma variedade maior na sua quantidade de alternativas necessárias.

Código Genérico

```
switch (condição)
{
    caso 1:
        comando;
        pare;
    caso 2:
        comando;
        pare;
    caso 3:
        comando;
        pare;
    padrão:
        comando;
}
```

Texto

```
verifique (fruta)
    caso banana:
        mostrar preço da banana;
        pare;
    caso maçã:
        mostrar preço da maçã;
        pare;
    caso uva:
        mostrar preço da uva;
        pare;
    padrão:
        mostrar "Não temos esse produto";
```

Linguagem X

```
1. switch (fruta)
2. {
3.     case banana:
4.         mostrar "Bananas custam R$10 a dúzia";
5.         break;
6.     case maçã:
7.         mostrar "Maçãs custam R$15 reais a dúzia";
8.         break;
9.     case uva:
10.        mostrar "Uvas custam R$20 o kilo";
11.        break;
12.    default:
13.        mostrar "Não temos essa fruta";
14. }
```

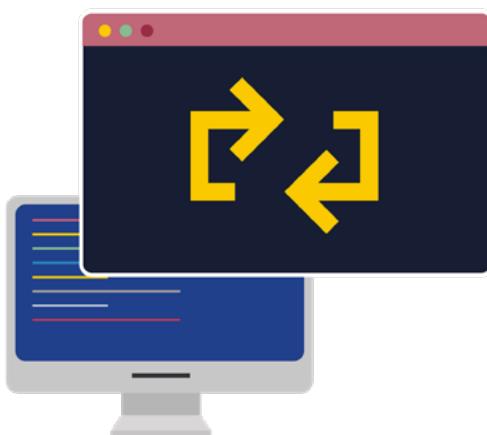
Importante!

A instrução opcional *break* associada a cada *case* garante que o programa saia da condicional *switch* e execute a instrução que segue logo após o *switch*. Caso *break* seja omitido, o programa continua a execução para a próxima instrução dentro de *switch*.

No exemplo, "fruta" é avaliado como "Bananas", o programa corresponde o valor com o *case* "Bananas" e executa a instrução associada. Quando *break* for encontrado, o programa para (*break*), saindo de *switch* e executa a instrução localizada após o *switch*. Se *break* fosse omitido, a instrução para "Maçãs" também seria executada.



ESTRUTURAS DE REPETIÇÃO



As estruturas de repetição permitem repetir um comando ou um conjunto de instruções, de acordo com uma condição ou um contador.

Essas estruturas têm usos diversos, como tecidos com desenhos em padrão, renderização de estruturas metálicas, jogos em geral, embalagem de produtos em lotes, entre outros.

PENSE NISSO...

Por exemplo, quero um código para escrever a frase "Eu sou capaz" 10 vezes. Qual dessas opções parece ser melhor?

Digitar dez vezes o mesmo comando

Escreva "eu sou capaz"
Escreva "eu sou capaz"

OU

Usar uma estrutura de repetição

Escreva "eu sou capaz" 10 vezes

Assim como a declaração de variáveis ou os comandos de decisão, a sintaxe das estruturas de repetição depende da linguagem de programação usada.

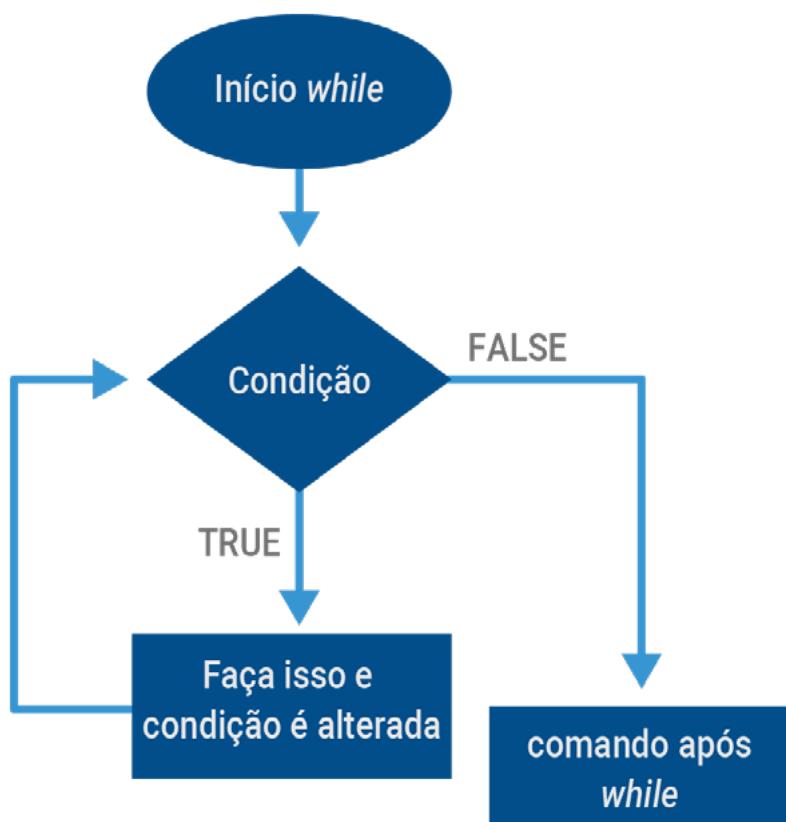
Vamos abordar três comandos de repetição muito comuns:

- ENQUANTO (while)
- FAÇA ENQUANTO (do while)
- PARA (for)

COMANDO ENQUANTO (WHILE)

Este comando repete a instrução enquanto a condição for verdadeira. Quando a condição passa a ser falsa, os comandos do while não são executados e são executados os comandos após o while. A condição do while deve ser alterada dentro do while, como um contador.

Note que a condição é checada antes da instrução dentro do while.



Código genérico

```
while (condição) {
    comando;
    altera contador;
}
```

Texto

```
ENQUANTO (contador < 10) {
    escreva "eu sou capaz"
    soma + 1 ao contador
}
```

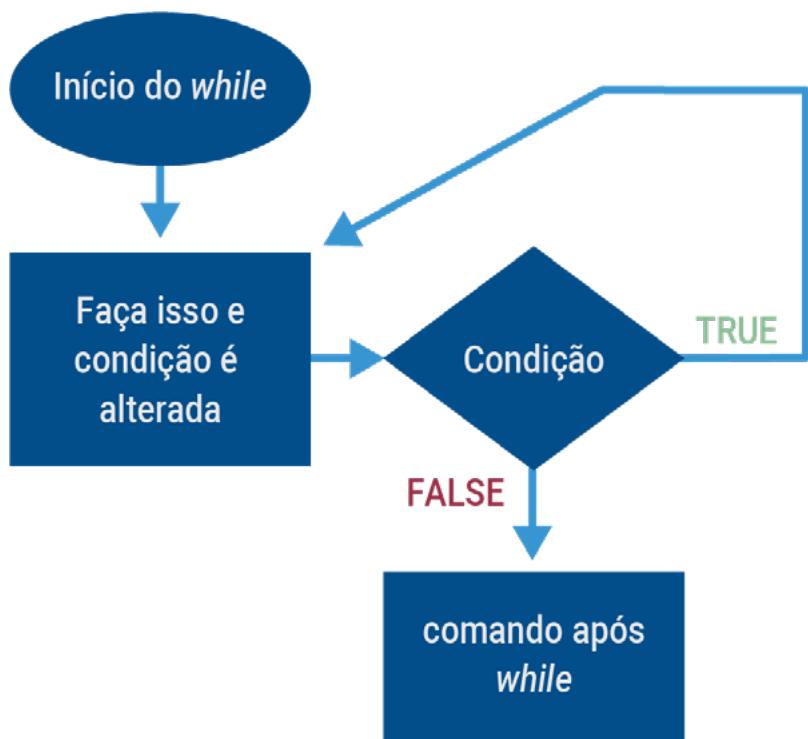
Linguagem X

1. Var contador = 1
2. while (contador < 10) {
3. console.log "eu sou capaz"
4. contador = contador + 1
5. }

COMANDO FAÇA ENQUANTO (DO WHILE)

Assim como o while, este comando repete a instrução enquanto a condição for verdadeira, porém a condição é verificada após a instrução do while. Portanto, se o contador for alterado dentro da instrução, isso afetará a verificação da condição.

O comando do while é usado quando é necessário executar o bloco de instruções interno pelo menos uma vez.



```
Código genérico
do {
    comando
}
while (condição);
```

Texto

```
FAÇA {
    escreva "eu sou capaz"
    soma um ao contador
}
ENQUANTO (contador < 10)
```

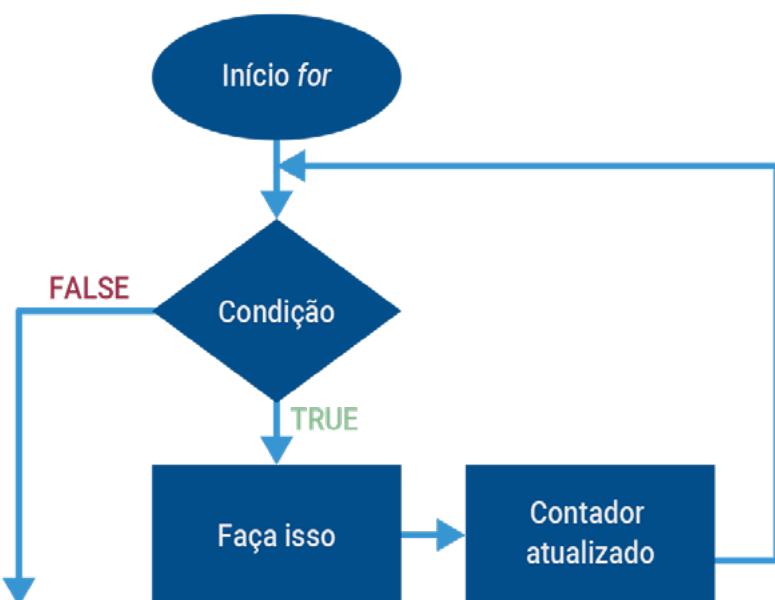
Linguagem X

1. Var contador = 1
2. do {
3. console.log "eu sou capaz"
4. contador = contador + 1
5. }
6. while (contador < 10)

COMANDO PARA (FOR)

O comando for é usado para repetir a instrução por um número determinado de vezes. É usado para repetições simples, pois em repetições mais complexas é recomendado usar o while.

É necessário usar uma variável como contador, que sofrerá um acréscimo ou decréscimo ao final do bloco de instruções interno.



Código genérico

```
for (contador, condição, incremento) {  
    comando  
}
```

Texto

```
para (contador = 0, contador < 10, contador = contador +  
1) {  
    escreva "eu sou capaz"  
}
```

Linguagem X

1. for (contador = 0, contador < 10, contador++){
2. console.log "eu sou capaz"
3. }

OPERADORES

Operadores são símbolos que auxiliam na sintaxe de comandos, expressões e funções.

OPERADORES ARITMÉTICOS

São os símbolos que representam os cálculos básicos matemáticos.

Operação	Sinal	Exemplo	Resultado
Adição	+	1+10	11
Subtração	-	10-1	9
Multiplicação	*	10*2	20
Divisão	/	10/2	5
Resta da divisão inteira	%	10%4	2
Incremento	++	6++	7
Decremento	--	6--	5

OPERADORES RELACIONAIS

São os símbolos que representam os cálculos básicos matemáticos.

Significado	Operador	Exemplo
Igual	<code>==</code>	<code>X == 10</code>
Diferente	<code>!=</code>	<code>X != 10</code>
Menor que	<code><</code>	<code>X < 10</code>
Menor ou igual que	<code><=</code>	<code>X <= 10</code>
Maior que	<code>></code>	<code>X > 10</code>
Maior ou igual que	<code>>=</code>	<code>X >= 10</code>

OPERADORES LÓGICOS

São os símbolos que representam os cálculos básicos matemáticos.

Significado	Operador	Exemplo
E (AND)	<code>&&</code>	<code>(X < 10) && (X > 0)</code>
OU (OR)	<code> </code>	<code>(X < 10 X > 0)</code>
Negação	<code>!</code>	<code>!true == false</code>

NESTE DESAFIO...

LÓGICA DE PROGRAMAÇÃO | DESAFIO 2



Você percebeu que, para nos auxiliar na criação de algoritmos, usamos as técnicas de programação, como fluxogramas, estruturas de decisão e repetição, além de elementos e conceitos que nos auxiliam na sintaxe dos comandos, como tipos e estrutura de dados, variáveis, constantes e operadores.

NO PRÓXIMO DESAFIO...

Realizaremos a implementação do código, utilizando uma linguagem de programação específica para criar nosso sistema.

DESAFIO 3

Agora que você já sabe aplicar técnicas de programação para o desenvolvimento de códigos, nesta etapa, você deverá:

- Utilizar linguagem de programação para implementação do sistema.

Para isso, você estudará os seguintes conteúdos:

- Definição, características e níveis de linguagem de programação;
- Etapas do processo de conversão: interpretação, ligação, compilação e montagem;
- Características das linguagens de programação;
- Semântica, indentação, bibliotecas e APIs;
- Técnicas de programação: Java, C#, JavaScript ou Python;
- Frameworks;
- Aplicações de programação: Assembly, C, C++, C#, Visual Basic, Java, Python, PHP, JavaScript.

LINGUAGENS DE PROGRAMAÇÃO

Conforme você notou, uma linguagem de programação é formada por um **conjunto de regras, sintaxe e semântica** para escrevermos o código fonte de um programa (nossos algoritmos).

Quando falamos em **sintaxe e semântica**, você sabe o que significa?



Você sabia?

Os computadores entendem somente sequências de números binários, ou seja, 0 e 1. Imagine se os programadores precisassem utilizar números binários como 1011101110110111011001100 para cada instrução do processador e para cada endereço de memória a ser acessado.



Por isso, para facilitar esse processo, as linguagens de programação começaram a ser desenvolvidas. Embora se diferenciem em sintaxe e recursos, existe um ponto em comum, que é a existência de um compilador ou editor para escrever o programa que interpretará os comandos programados e os transformará em binários para que as instruções sejam entendidas pelo processador.

SEMÂNTICA

A semântica de uma língua refere-se ao significado das palavras.

Em tecnologia, semântica está relacionada ao contexto e ao significado que uma determinada estrutura representa dentro de um código, ou seja, o significado de cada trecho de código.

Um ponto importante quando nos referimos a semântica, é que, quando falamos em uma determinada linguagem, além de pensar em sua estrutura, temos de pensar em seu significado, ou seja, como o computador vai compreender aquela informação.



'um mamão' é diferente de 'uma mão'



Vamos ao exemplo?

Para especificarmos o espaço para o logo e nome de uma empresa, ou seja, o cabeçalho da página, podemos utilizar a estrutura `<header> </header>` para representação.

O meu código HTML	O que é interpretado pelo navegador
<pre> 1. <!doctype html> 2. <html> 3. <head> 4. <header>Título</header> 5. </head> 6. <body> 7. </body> 8. </html> </pre>	Título

Em outras palavras, o código pode estar escrito de uma maneira que não tenha nenhum sentido. Alguns exemplos de erros de semântica podem ser:

- dividir um número por uma *string*;
- fazer cálculos com variáveis que não sejam números;
- usar variáveis não declaradas;
- dividir zero por zero.

Veja na imagem a seguir um exemplo de erro de semântica no código.

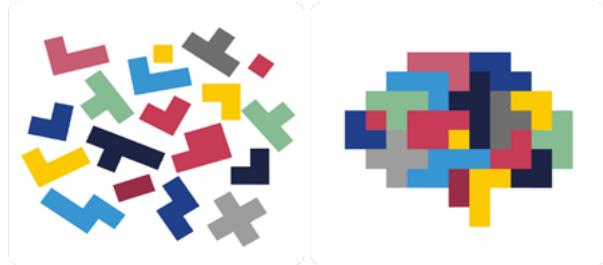
```

10  const nome = "cris";
11  // divisão entre tipos distintos - string por um número
12  console.log(nome/idade);

```

SINTAXE

A sintaxe de uma língua trata-se das regras que regem a disposição das palavras em uma frase, de uma forma lógica, entre as múltiplas combinações possíveis para transmitir um significado completo e compreensível.



Eu **comprei** o livro ontem.

Sujeito | verbo | predicado

Por exemplo, em língua portuguesa, utilizamos, geralmente, a seguinte estrutura: Alguns elementos dessa frase podem ser dispostos de outra forma, sem que haja alteração de sentido:

- Ontem, eu comprei o livro
- Eu comprei, ontem, o livro

Já outros, se forem alterados, tornarão a frase sem sentido:

- Comprei o ontem eu livro

Portanto, em nossa língua, há algumas **regras que determinam o modo como as palavras podem combinar-se para transmitir um sentido completo.**

Da mesma forma ocorre com as linguagens de programação. Cada uma apresenta estruturas sintáticas e semânticas específicas, cujo objetivo é facilitar as instruções de processamento ao computador.

Alguns exemplos de erros de sintaxe podem ser:

- parênteses que abrem mas não fecham;
- falta ou duplicidade de ponto e vírgula, dois pontos, chaves e afins;
- uma palavra-chave sendo usada numa posição inesperada;
- erros de digitação.

Veja na imagem a seguir um erro de sintaxe:

```
1 // erros
2 const idade = 18;
3 // parênteses sem fechar
4 if (idade >= 18 {
5   console.log("você tem permissão");
6 } else {
7   console.log("sem permissão");
8 }
```

INDENTAÇÃO DE CÓDIGO

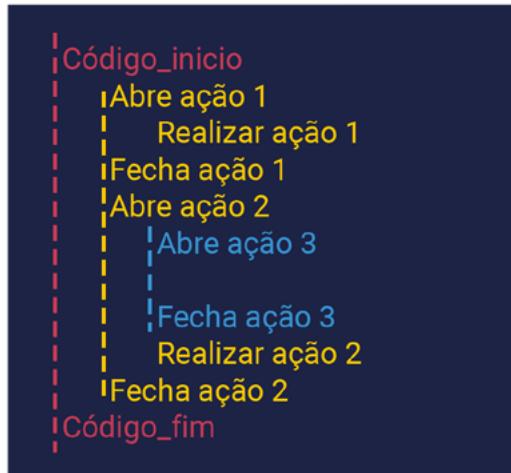
Além da sintaxe e semântica, o código/algoritmo precisa estar organizado e estruturado, de modo que haja uma hierarquia entre suas partes, facilitando sua legibilidade. Esse processo chama-se indentação ou identação.



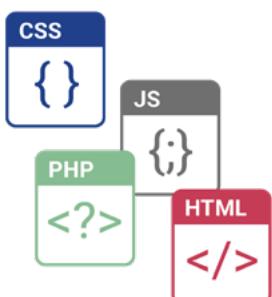
É como se fossem os espaçamentos do parágrafo para identificar o seu início. São utilizados para especificar, em algumas linguagens, o início e o fim de uma instrução.

Observe nesse exemplo que, com a indentação correta, é mais fácil perceber se o algoritmo está bem organizado e se a estrutura do código está correta.

Perceba que as ações 1 e 2 estão no mesmo nível, mas a ação 3 está dentro da ação 2.



CARACTERÍSTICAS DAS LINGUAGENS



As linguagens são utilizadas para diferentes propósitos e apresentam diferentes características.

```
<div class="container">
  <div class="row">
    <div class="col-md-6 col-lg-8"> <!-- BEGIN NAVIGATION
      <nav id="nav" role="navigation">
        <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="home-events.html">Home Events</a></li>
          <li><a href="multi-col-menu.html">Multiple Column Men
          <li class="has-children"> <a href="#" class="current">
            <ul>
              <li><a href="tall-button-header.html">Tall But
              <li><a href="image-logo.html">Image Logo</a>
              <li class="active"><a href="tall-logo.html">Ta
            </ul>
          </li>
          <li class="has-children"> <a href="#">Carousels</a>
          <li><a href="variable-width-slider.html">Variab
        </ul>
      </nav>
    </div>
  </div>
</div>
```

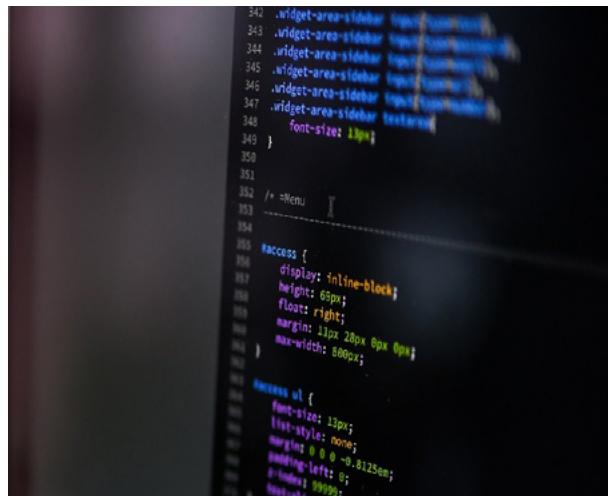
Marcação

Linguagens de marcação são utilizadas para definir a estrutura de uma página ou de um documento, por exemplo. Ela é como se fosse o esqueleto de nosso corpo que nos molda. A linguagem de marcação HTML (*Hyper Text Markup*, ou Linguagem de Marcação de Hipertexto, português) é uma das mais comuns.

Fonte: Pixabay

Folha de estilos

Linguagens de folha de estilos são utilizadas para estilizar os elementos de uma página desenvolvida com uma linguagem de marcação. Exemplo: utilizamos CSS (*Cascading Style Sheet*, ou Folhas de Estilo em Cascatas, em português) para estilizar páginas desenvolvidas em HTML. É como se fosse a roupa que nós vestimos.



Fonte: Pixabay



Fonte: Pixabay

Programação

Conforme você viu, as linguagens de programação são um conjunto de regras, sintaxes para escrevermos instruções, que farão com que o computador ou software execute alguma ação.

TIPOS DE LINGUAGENS



Além de apresentar diferentes propósitos, as linguagens de programação podem ser estruturadas, orientadas a objetos e funcionais:

Estruturada

Trabalha apenas com um arquivo, codificando o algoritmo linha após linha seguindo uma sequência lógica de evento. Exemplos: **C#, Delphi, Dart, Java, Kotlin, PHP, TypeScript.**

A programação estruturada é composta por três tipos básicos de estruturas, conforme você estudou:

- **sequências:** são os comandos a serem executados.
- **condições:** sequências que só devem ser executadas se uma condição for satisfeita (exemplos: if-else, switch, entre outros).
- **repetições:** sequências que devem ser executadas repetidamente até uma condição ser satisfeita (for, while, do-while etc.).

Orientada a objetos

Trabalha com uma abordagem de reusabilidade de código, com o objetivo de simplificar a manutenção de um programa. Baseado na composição e interação entre diferentes unidades dentro do seu software. Exemplos: **C++, VB .NET, Java, Object Pascal, Objective-C, Python, SuperColider, Ruby, Smalltalk**, entre outras. As imagens a seguir mostram a diferença entre uma programação estruturada e uma orientada a objetos.



Funcional

Atua voltada à construção por meio de funções matemáticas (ou seja, quando chamadas mais de uma vez, possuem o mesmo resultado de saída). São aplicadas para resolver problemas que envolvem principalmente paralelismo e/ou processamento de quantidades grandes de dados. Exemplos: **Prolog, Lisp, Scheme, ML, Miranda, Haskell, Elixir.**

```
public int doFunction(int x){
    return (x * x) + 3;
}
```

```
System.out.println(doFunction(2)); // A saída será "7"!
```

NÍVEIS DE LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação também podem ser classificadas em dois níveis: de alto nível ou de baixo nível.

Alto nível

Linguagens de alto nível são mais intuitivas e amigáveis para o programador. Ela é mais semântica e mais próxima à linguagem natural, simplificando o desenvolvimento de algoritmos, por isso são mais utilizadas.

JavaScript, Java, C#, PHP são alguns exemplos desse tipo de linguagem.

```
1  <!doctype html>
2  <html>
3
4  <head>
5      <meta charset=utf-8>
6      <title>Exemplo de HTML</title>
7      <style>
8          p {
9              font-family: monospace;
10         }
11     </style>
12   </head>
13
14   <body>
15       <p>HTML</p>
16       <p>Hello, World!</p>
17   </body>
18
19   </html>
```

Baixo nível

Linguagens de baixo nível são voltadas para o entendimento da máquina. A sintaxe costuma ser mais complexa e menos intuitiva para programador. O tempo de aprendizado é usualmente maior em relação às linguagens de alto nível.

Assembly é um exemplo de linguagem de baixo nível. Veja na tabela a seguir alguns comandos em Assembly e seus equivalentes para a máquina.

Assembly	Código da máquina
add \$t1, t2, \$t3	04CB: 0000 0100 1100 1011
addi \$t1, t2, 60	16BC: 0001 0110 1011 1100
and \$t3, \$t1, \$t2	0299: 0000 0010 1001 1001
andi \$t3, \$t1, 5	22C5: 0010 0010 1100 0101

IMPLEMENTAÇÃO

Algumas linguagens podem possuir mais de uma implementação para o código ser executado, sendo caracterizadas como **compilada e interpretada**.

Compilada

O verbo "compilar" pode transmitir a ideia de converter um objeto em outro. Assim, podemos dizer que o processo de compilação envolve **converter o nosso código para linguagem de máquina por meio de um programa chamado compilador**.

Uma das vantagens da compilação é que os erros aparecem no processo de compilação e não no processo de execução (no momento em que a linguagem for utilizada).

Ada, ALGOL, BASIC, C, C++, C#, CLEO, COBOL são alguns exemplos desse tipo de linguagem.

Interpretada

Uma linguagem interpretada executa um processo similar. Ela será **interpretada por meio de um programa chamado interpretador, que irá converter nosso código para linguagem de máquina**. O interpretador realiza a conversão linha por linha, e não o programa inteiro, diferentemente do que ocorre com o compilador.

O processo de interpretação permite uma inicialização mais rápida e mais fácil para a execução do projeto. Sua desvantagem é que, caso tenha algum erro em sua aplicação, ele só será apresentado no momento da execução.

ActionScript, Java, Python, APL, ASP são alguns exemplos desse tipo de linguagem.

É comum encontrar cenários de discussão se uma linguagem é compilada ou interpretada.

FERRAMENTAS DE DESENVOLVIMENTO

Para a programação de qualquer sistema, existem ferramentas e recursos disponíveis que facilitam muito a rotina de trabalho do programador, permitindo que o desenvolvimento seja mais fácil e rápido, como bibliotecas, frameworks e APIs.



BIBLIOTECAS

Uma biblioteca é uma coleção de funções pré-escritas por outro programador, que realizam ações pré-definidas e podem ser reutilizadas em diferentes projetos, sem que você precise reescrever um código. Trata-se de funções que já existem no programa e que você pode usar.

Por exemplo: para calcular a raiz quadrada na linguagem de programação C, você pode utilizar o `sqrt` da biblioteca `math.h`.

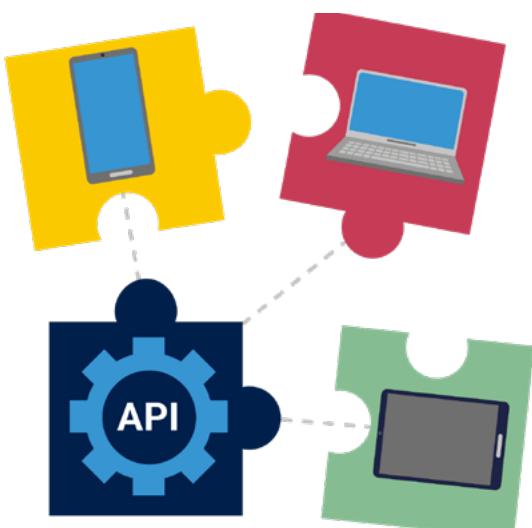
As bibliotecas são nativas da respectiva linguagem, mas você também pode importar bibliotecas que não são nativas ou criar bibliotecas internas no seu projeto.

Importante!

Dependendo do contexto inserido, você pode encontrar diferentes significados para o mesmo termo.



API



API significa Application Programming Interface ou Interface de Programação de Aplicativo, em português. Trata-se de um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na web, permitindo a comunicação entre softwares, por meio de protocolos e recursos (definições) bem estruturados.

O Google Maps é exemplo de APIs. Provavelmente você já visitou algum site que possui uma página em que é possível visualizar o mapa do Google Maps para saber

a localização do estabelecimento e verificar qual o melhor trajeto para chegar até lá. Portanto, por meio de seu código original, o Google Maps permite que outros sites e aplicações utilizem seus dados, adaptando-o para utilizar esse serviço.

FRAMEWORKS

Frameworks usualmente possuem um modelo ou estrutura de códigos prontos a serem seguidos.

Por exemplo, imagine a tela para realizar uma compra em um site. Normalmente, essas telas são muito parecidas em todas as aplicações, possuindo uma foto do produto, sua descrição técnica, um campo para selecionar a quantidade, a forma de pagamento, o endereço de entrega e um botão para finalizar a compra.



Pensando nisso, alguns programadores criaram um framework que implementa essas funções no seu código. Assim, não é necessário reescrevê-lo sempre que for preciso criar uma tela de compra.

É importante ressaltar que para cada linguagem de programação, haverá frameworks diferentes, mesmo que exerçam a mesma função, assim como alguns frameworks externos ao seu código fonte podem engessar o seu código, devido a não apresentarem flexibilidade em suas utilizações.

Seguem alguns exemplos de frameworks:

- Java – Hibernate e Spring
- Ruby – Ruby on Rails
- JavaScript – AngularJS
- Python – Django
- PHP – Zend e Laravel
- C# – ASP .NET
- CSS – Bootstrap

Os exemplos a seguir mostram a linguagem Java, com e sem framework:

Sem framework

```
1  @WebServlet("/teste")
2  public class TesteServlet extends javax.servlet.http.HttpServlet {
3      @Override
4      protected void doGet(HttpServletRequest request,
5          HttpServletResponse response) {
6              //...
7      }
8  }
```

Com framework

```
1  @Controller //Define que minha classe será um controller
2  public class HomeController {
3
4      @RequestMapping("/home") //Define a url que quando for requisitada chamara o metodo
5      public ModelAndView home(){
6          //Retorna a view que deve ser chamada, no caso home (home.jsp) aqui o .jsp é omitido
7          return new ModelAndView("home");
8      }
9
10 }
```

Saiba mais

Conheça mais frameworks usados no mundo da programação:

<https://bit.ly/3D46lyT>

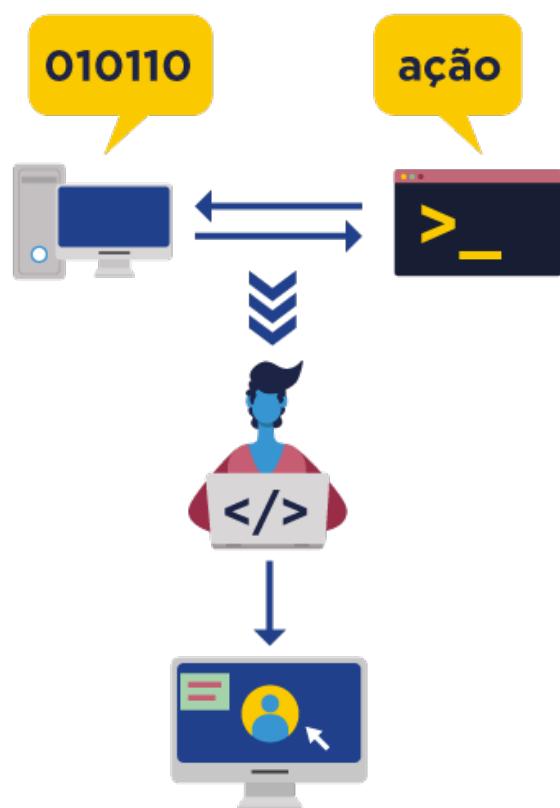


CONVERSÃO DE LINGUAGENS

Como você já aprendeu, para executar qualquer tarefa, o computador necessita das instruções precisas e ordenadas, fornecidas pelos softwares. A ligação entre computador e software é feita por meio de uma linguagem de programação.

Mas, como fazer essa "ponte", se a máquina usa uma linguagem de programação de baixo nível e o software usa uma linguagem de programação de alto nível?

No início da computação, os códigos eram escritos de forma binária (0 e 1) em cartões perfurados, ou seja, cada comando gerava uma extensa sequência de 0 e 1. Numa linguagem da máquina, um instrução corresponde a uma ação do computador. Imagine que cada marca ou modelo de computador tem sua própria linguagem, ou seja, a extensa sequência binária de um mesmo comando é diferente para cada marca ou modelo.



Era um processo complexo, extenso, lento e caro. Esse cenário levou à necessidade de otimizar e baratear todo programação. Assim surgiram as linguagens de programação de alto nível e os processos de conversão de linguagem.

Devido à diversidade de linguagens de programação de baixo e alto nível, há vários tipos de conversão.

Tradução ou compilação

Tradução ou compilação é o processo de passar a linguagem de alto nível para linguagem de baixo nível por meio de diversas etapas de análise e traduções intermediárias. Cada instrução do código fonte gera várias instruções para a máquina.

Montagem

A montagem é um tipo de tradução na qual cada instrução do código gera apenas uma instrução para a máquina.

Ligação

A ligação é como chamamos a "ponte" entre as etapas intermediárias de tradução.

Interpretação

A interpretação é a conversão feita a partir do código fonte, comando por comando, em tempo de execução. Não há fases distintas, nem etapas intermediárias. A compilação é imediata na execução, ou seja, cada comando é lido, verificado, convertido e executado antes que o próximo comando seja lido.

APLICAÇÕES DE PROGRAMAÇÃO

São muitas as linguagens e termos utilizados no mundo da programação. Sempre existem discursos sobre qual linguagem é melhor, qual está em alta ou até qual linguagem pode possivelmente deixar de existir. O ponto é que as tecnologias e os novos recursos sempre estão sendo criados e só a experiência e as referências podem te ajudar a escolher a que melhor resolve seu problema.



Conheça a seguir algumas linguagens utilizadas e suas aplicações:

Linguagem	Características	Aplicações
Javascript	Uma das linguagens de programação mais utilizadas, tanto para desenvolvimento do front quanto back-end.	Desenvolvimento web, aplicativos, softwares.
Java	Utilizada em diferentes plataformas (Windows, macOS e Linux). Apresenta um ecossistema rico com robustez e escalabilidade para sistemas sofisticados.	Desenvolvimento de aplicativos e softwares.
PHP	Uma das linguagens mais utilizadas em desenvolvimento web e cenários e-commerce, gerenciador de conteúdo e criação de sites.	Desenvolvimento web.
C#	Uma das linguagens mais utilizadas em desenvolvimento back-end, desktop, web, aplicações mobile e multiplataformas. Usualmente fica entre as linguagens mais utilizadas no mundo do desenvolvimento.	Desenvolvimento de softwares, jogos.
Assembly	Utilizado muito para manipulação de hardware, ou instruções específicas do processador/sistemas embarcados de baixo nível.	Desenvolvimento de compiladores.
C	É muito utilizado em <i>sistemas embarcados</i> ¹ .	Desenvolvimento de sistemas operacionais, compiladores.
C++	É uma das linguagens utilizadas no desenvolvimento de jogos (é da Microsoft) e criação de interfaces gráficas.	Jogos, GUI (interfaces gráficas).
Visual Basic	"Uma linguagem de programação simples, mas poderosa, que você pode usar para estender aplicativos do Office." - Microsoft.	Roda como se fosse interno, para Office, sendo muito utilizado com Excel, VBA.

¹ Programas e sistemas embutidos em microprocessadores, que executam tarefas específicas em um aparelho.

Python	É muito utilizada e difundida no contexto de ciência de dados, machine learning.	Desenvolvimento web, inteligência artificial, ciência de dados.
--------	--	---

Para resolução da situação-problema apresentada, vamos aplicar a linguagem de programação JavaScript para implementação do sistema.

FERRAMENTAS PARA DESENVOLVIMENTO JAVASCRIPT

Para desenvolver aplicações em JavaScript, utilizaremos duas ferramentas gratuitas:

- Visual Studio Code
- Node.js

VISUAL STUDIO CODE

O Visual Studio Code, ou VSCode, é um editor de código fonte desenvolvido pela Microsoft e tem se popularizado entre os programadores, por ser uma ferramenta leve e que está disponível tanto para Windows, quanto para MacOS e Linux.

Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe (faz com que o editor exiba automaticamente o texto em diferentes cores/estilos), complementação inteligente de código, snippets (pequena região de código fonte reutilizável) e refatoração de código (processo de alterar um software de uma maneira que não mude o seu comportamento externo e ainda melhore a sua estrutura interna). Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalho e preferências.

NODE.JS

O Node.js é um ambiente para criar aplicações em JavaScript.

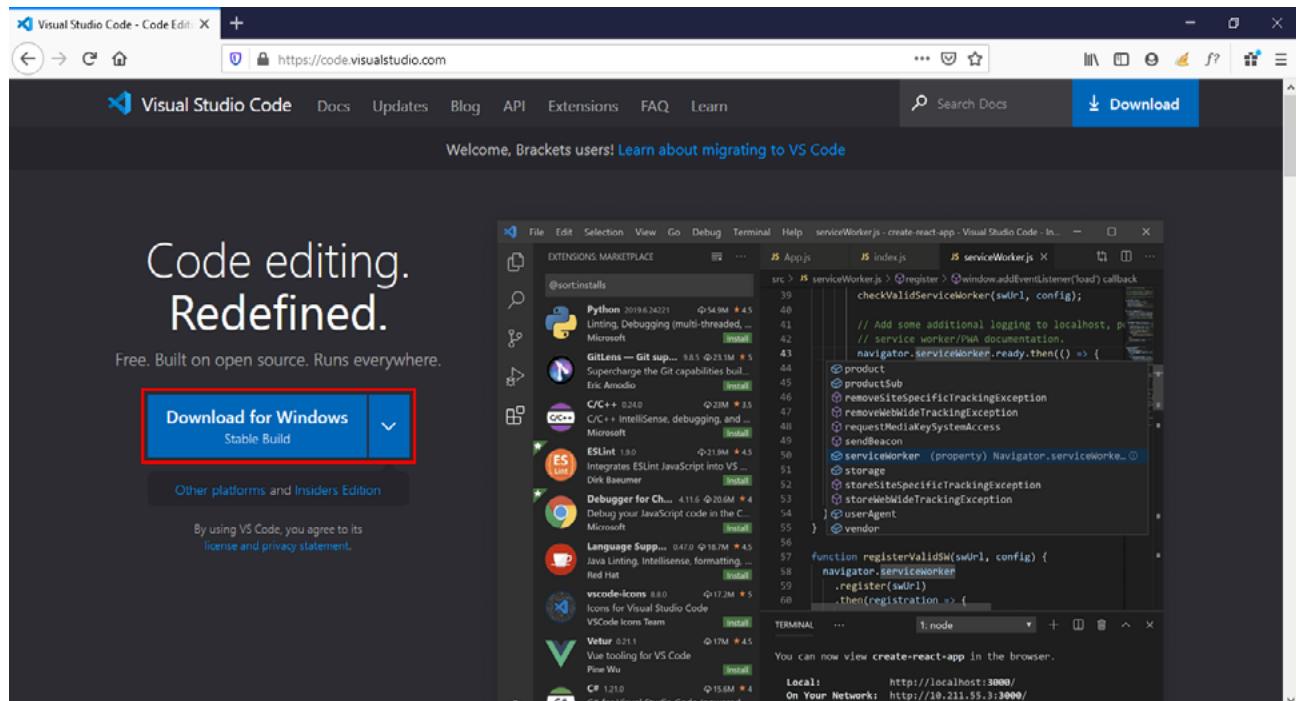
Ele foi desenvolvido para construirmos aplicações escaláveis que são executadas ao lado do servidor. Uma das principais características do Node.js que o diferenciam de linguagens tradicionais é o fato de ser single-thread, ou seja, ele só irá tratar uma requisição por vez, enquanto que nas outras linguagens, como PHP, Java, C#, a execução é *multithread*, o que significa que, para cada requisição recebida, é criada uma nova thread para tratá-la, o que consome mais memória, deixando o processamento mais lento.

INSTALAÇÃO DOS SOFTWARES

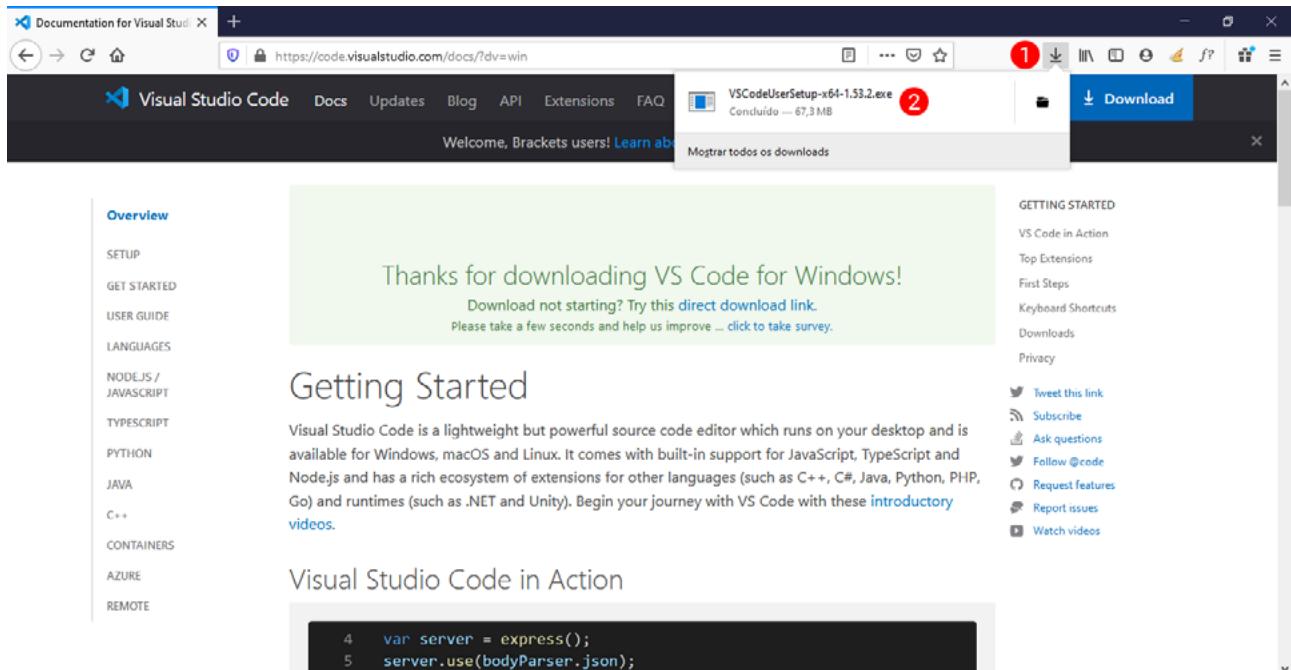
Visual Studio Code

Para instalar o Visual Studio Code, acesse o endereço <https://code.visualstudio.com/> e clique em "download for Windows". Caso você não utilize Windows, irá aparecer a mensagem conforme o sistema operacional que você utiliza.

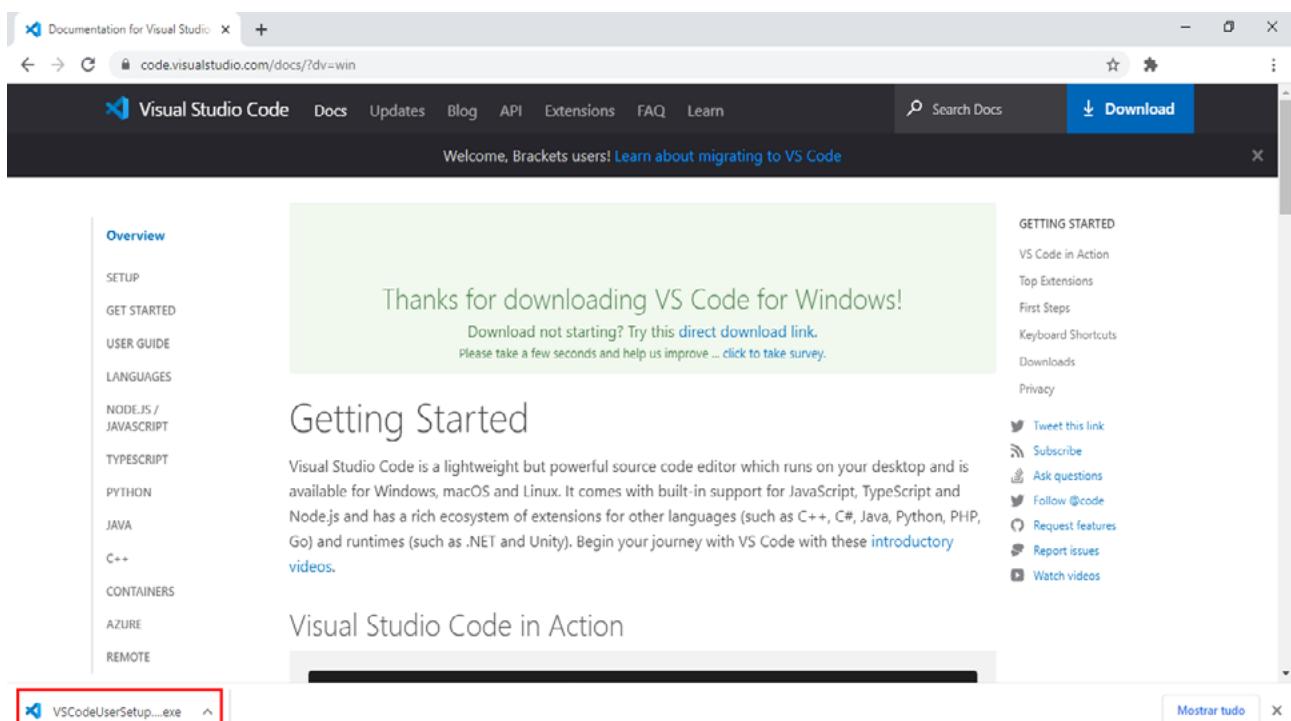
- Clique em "Download for Windows".



- Caso você use o Mozilla Firefox para fazer o download:
1 - clique no botão de exibição do progresso de downloads; e
2 - clique sobre o nome do arquivo para executá-lo.

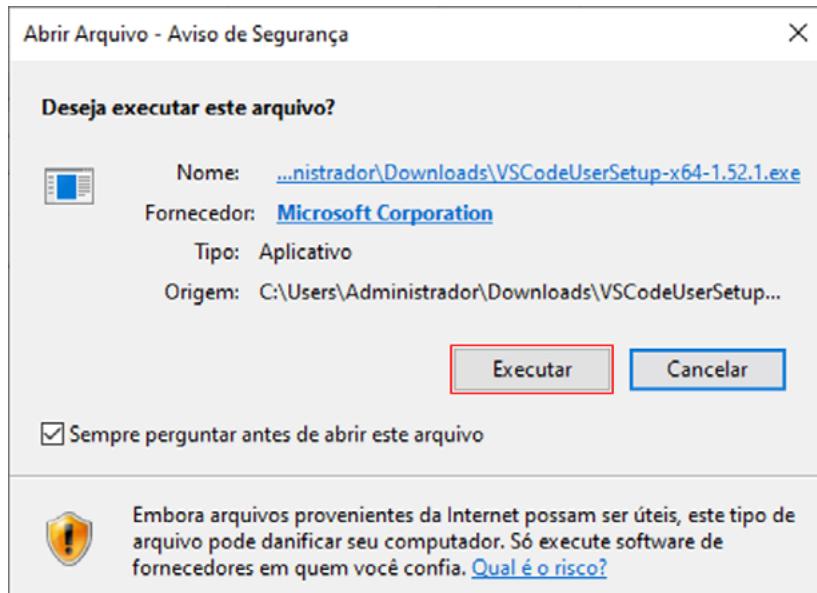


- Caso utilize o navegador Google Chrome, após realizar o download, basta clicar no nome do arquivo na barra inferior do navegador.

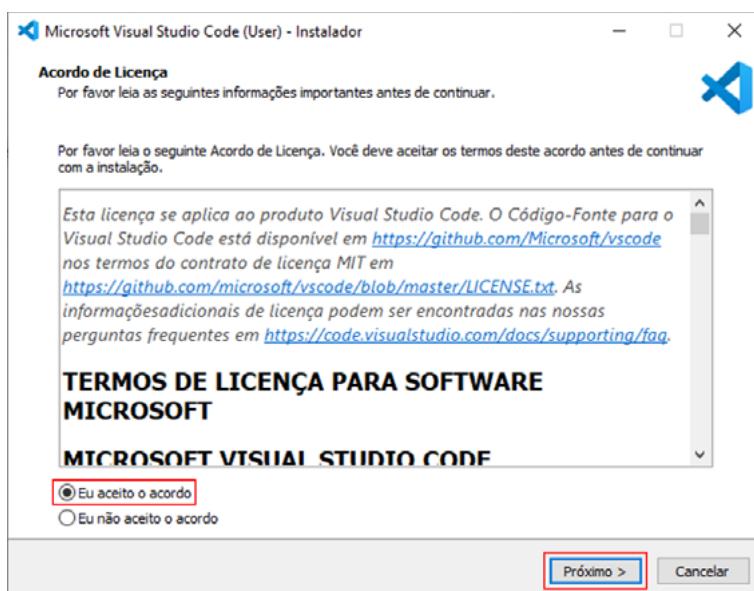


Acompanhe o passo a passo da instalação.

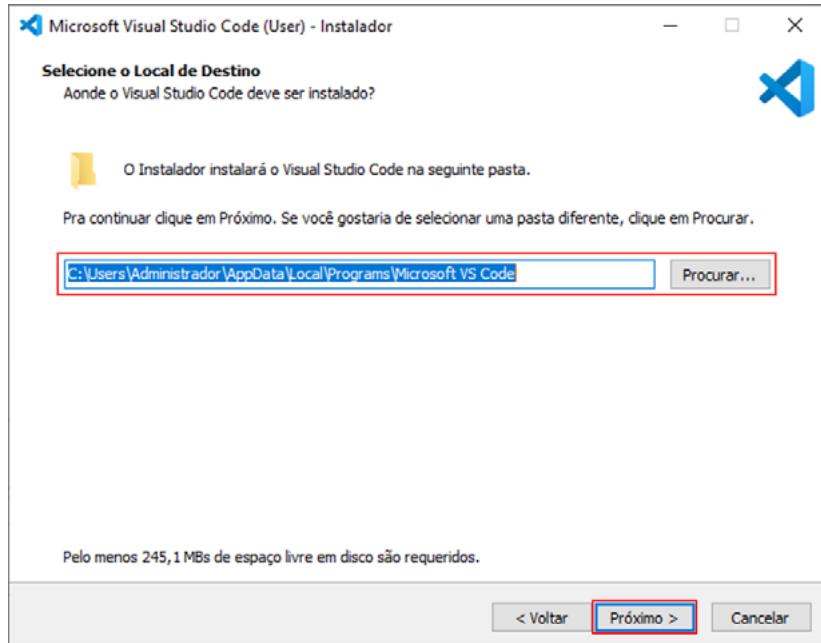
- Clique em "**Executar**".



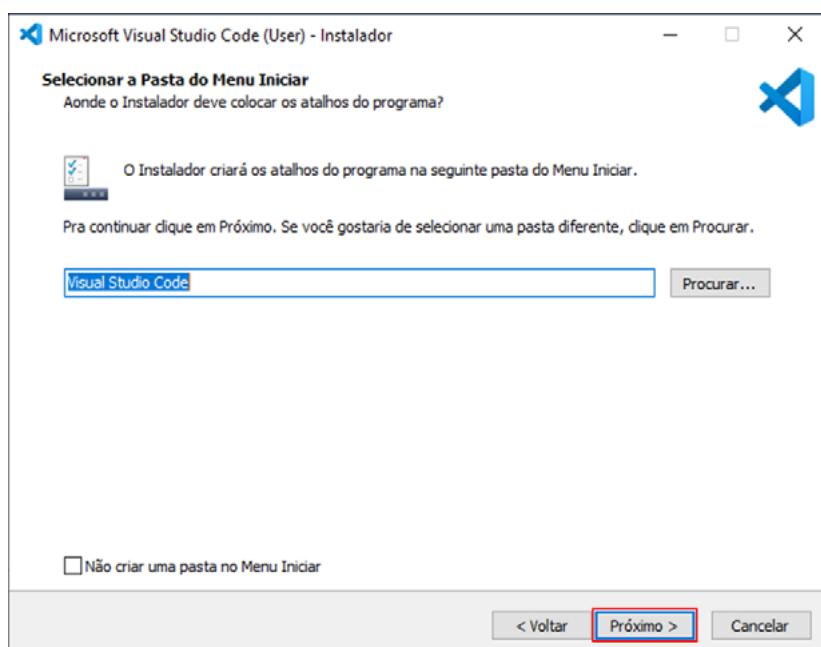
- Clique em "**Eu aceito o acordo**" e em "**Próximo**".



- Você pode escolher o lugar em que deseja instalar/salvar o Visual Studio Code. Iremos manter o padrão em nosso ambiente.
Clique em "**Próximo**".

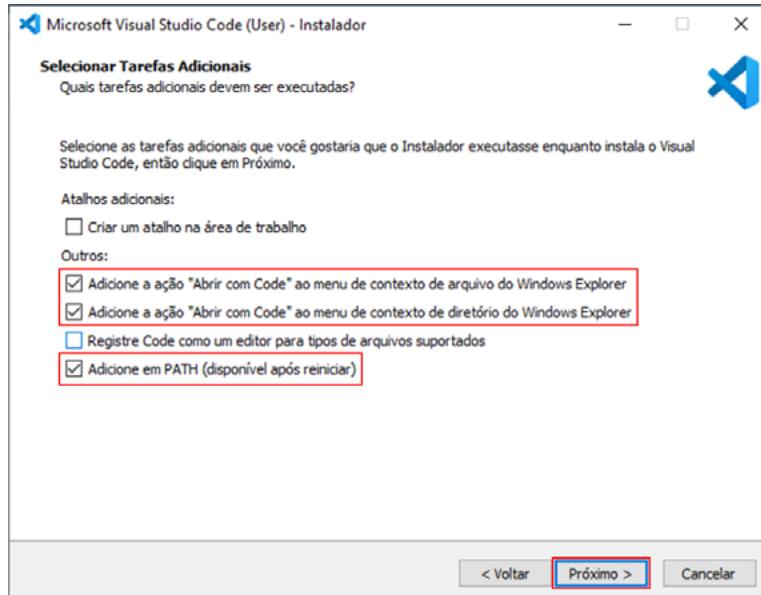


Clique em "Próximo".

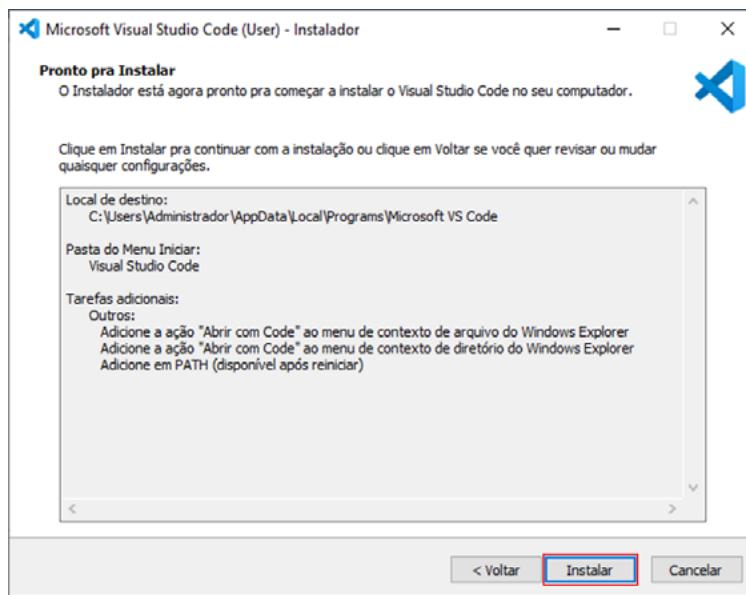


- Selecione as três configurações destacadas no exemplo da imagem ao lado para abrir o VSCode em qualquer diretório diretamente pelo clique do mouse.

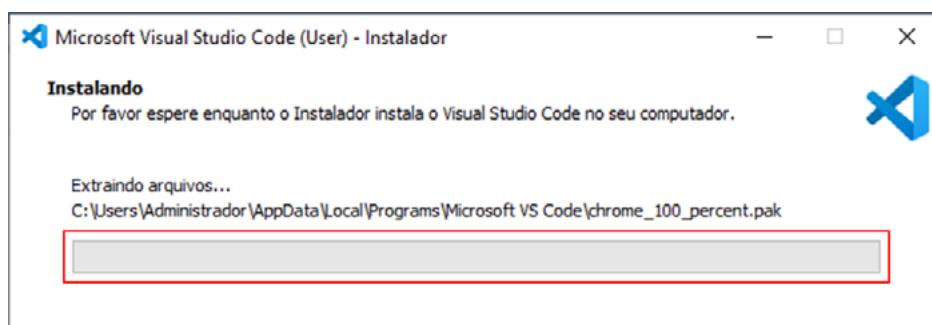
- Clique em "Próximo".



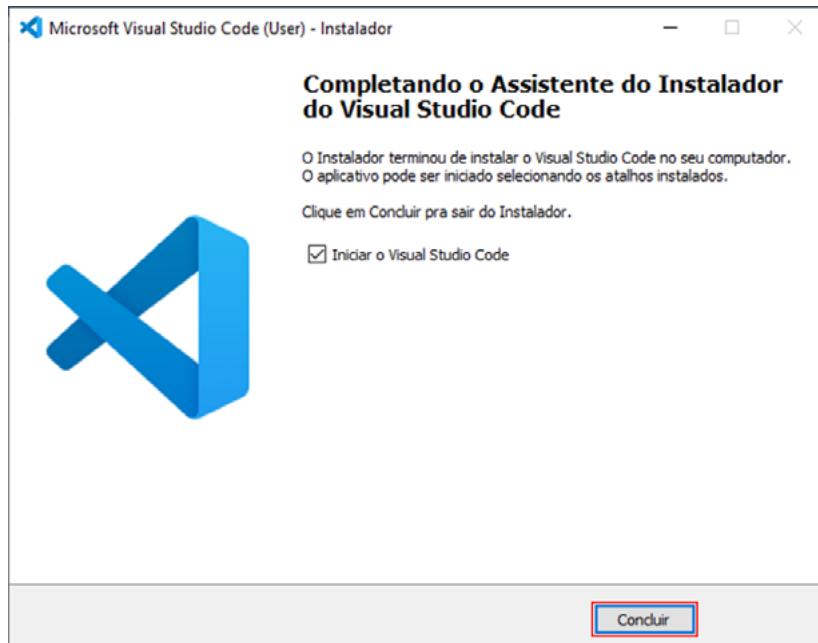
- Clique em "Instalar".



- Aguarde a barra da instalação ser completada.



- Pronto! Clique em "**Concluir**" para finalizar a instalação.



NODE.JS

Para instalar o Node, acesse o endereço <https://nodejs.org/en/download/> e clique em Windows Installer ou na versão correspondente do seu sistema operacional.

- Clique em "**Windows Installer**".

A screenshot of a web browser displaying the Node.js download page at https://nodejs.org/en/download/. The page features the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. The "Downloads" section highlights the "Latest LTS Version: 14.16.0 (includes npm 6.14.11)". It offers three main download paths: "LTS Recommended For Most Users", "Current Latest Features", and "Source Code". The "Windows Installer (.msi)" option under the LTS path is highlighted with a red box. Below the download sections, there is a table for Windows and macOS installers:

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

32-bit

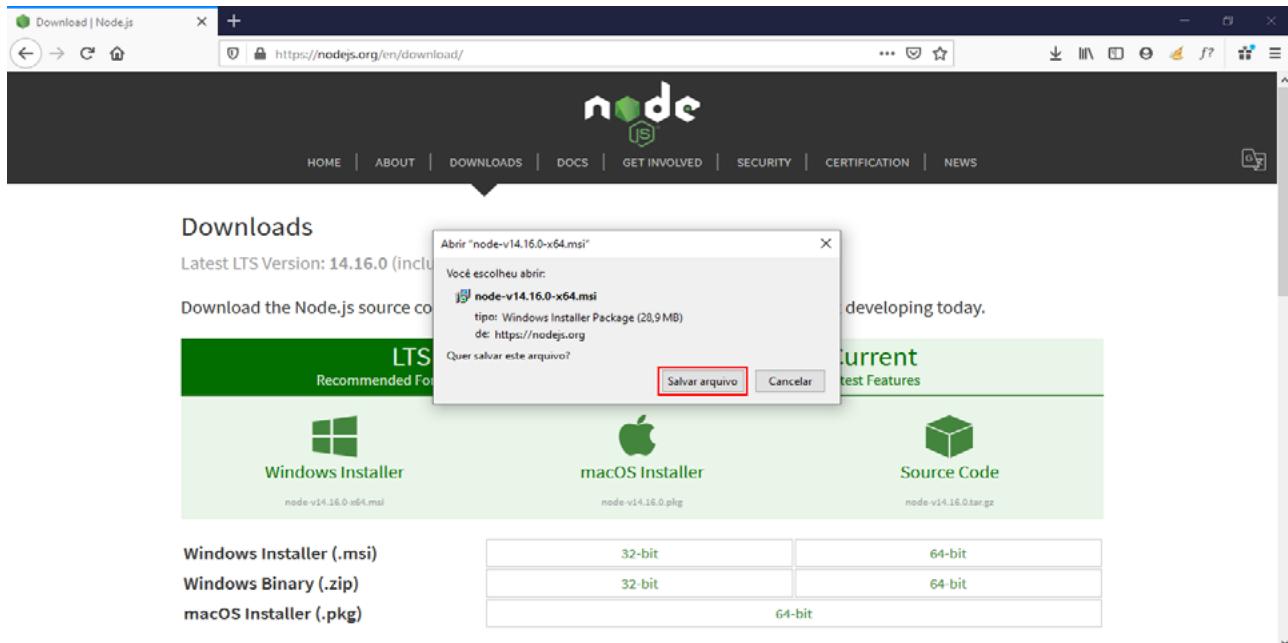
32-bit

64-bit

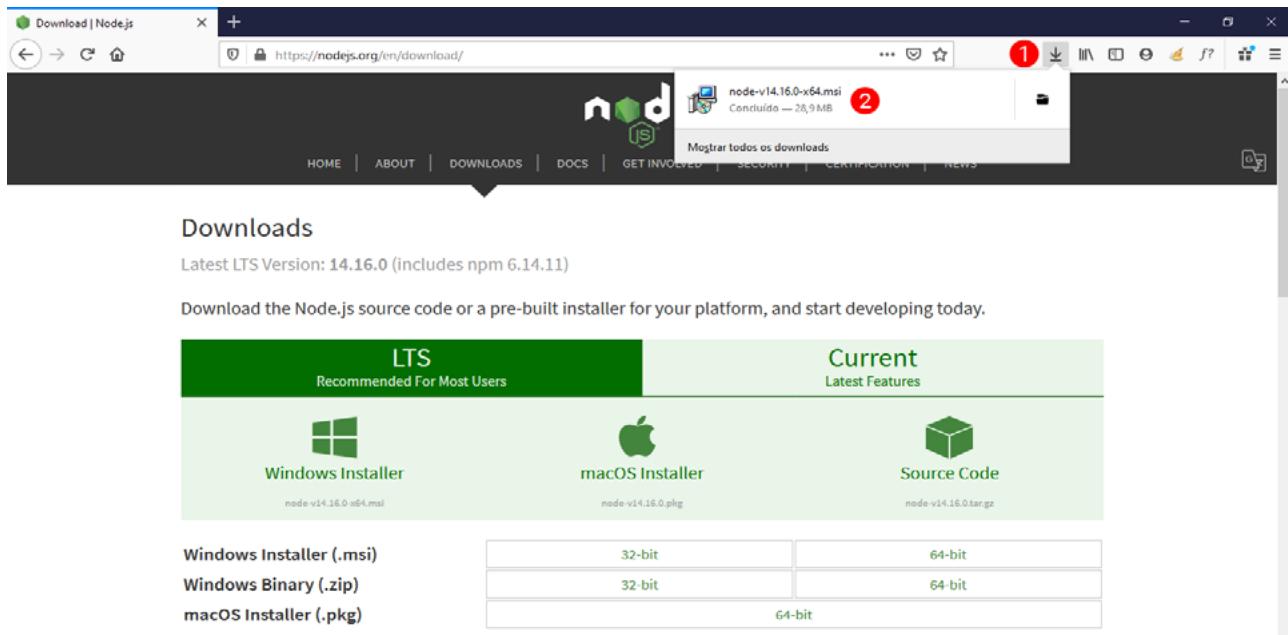
64-bit

64-bit

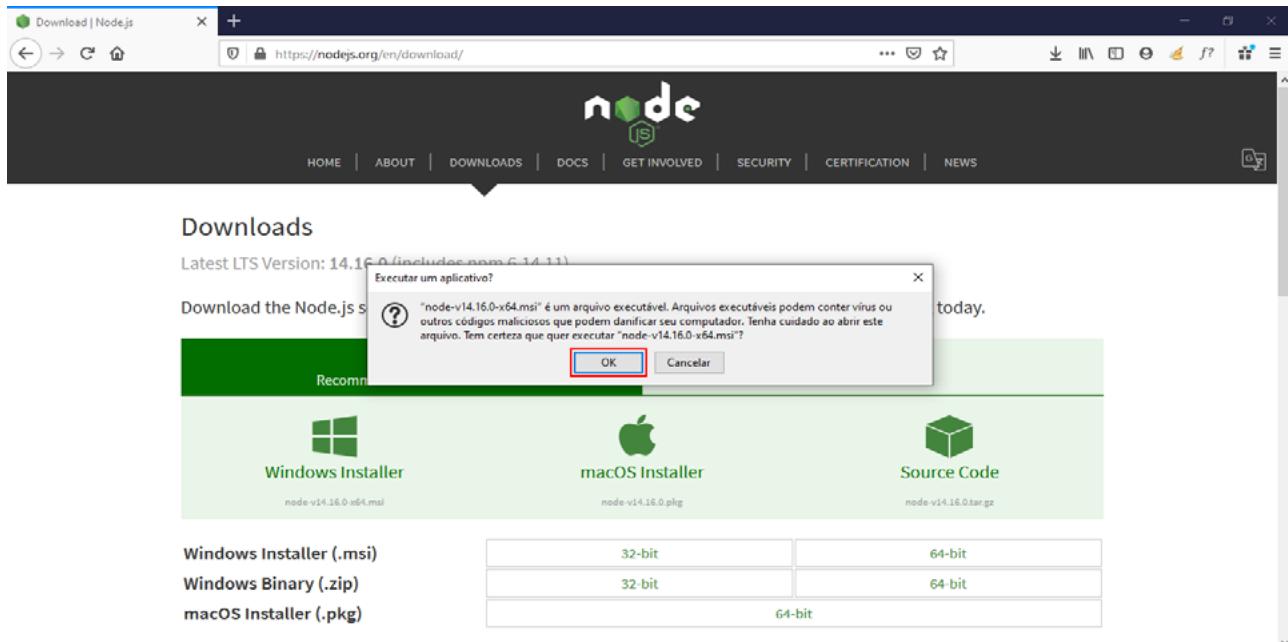
- Clique em "Salvar arquivo".



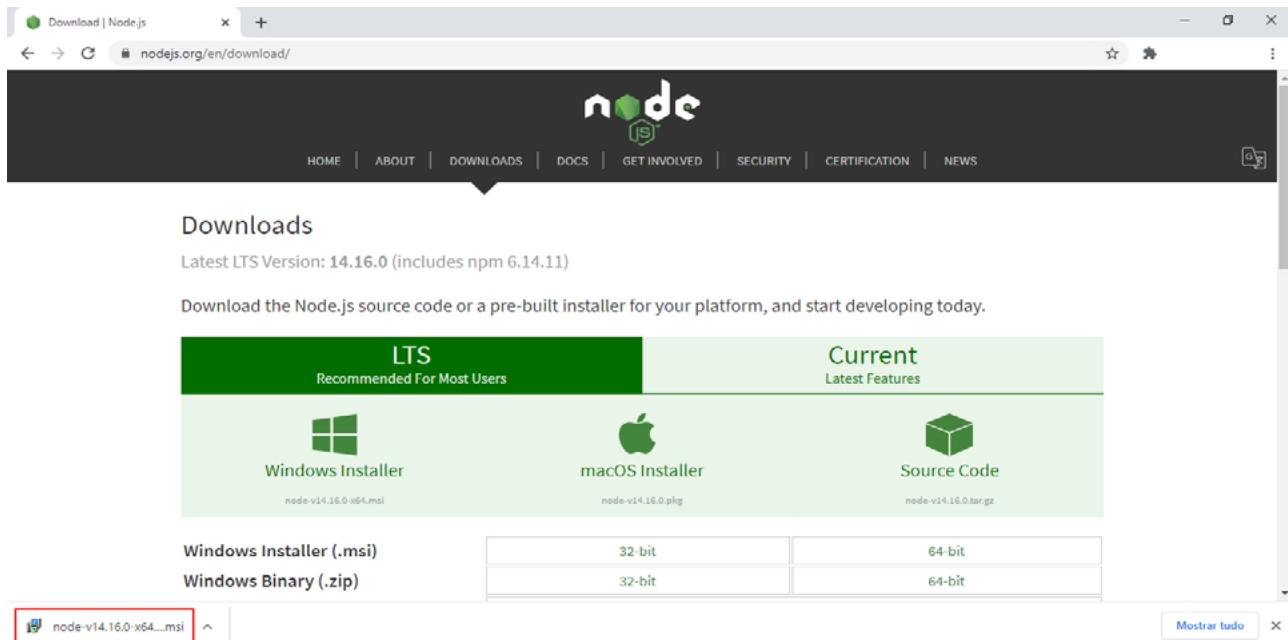
- Caso você use o Mozilla Firefox para fazer o download:
- 1 - clique no botão de exibição do progresso de downloads; e
- 2 - clique sobre o nome do arquivo para executá-lo.



- Dependendo da versão do Windows, um alerta de segurança é exibido na tela em função de ser um arquivo executável. Clique em "OK"

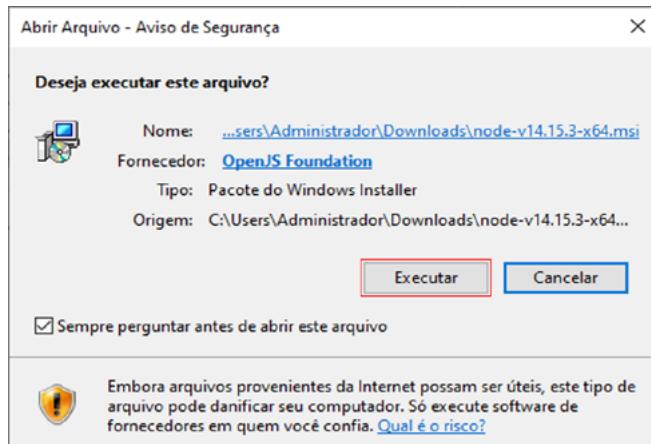


- Caso utilize o navegador Google Chrome, após realizar o download, basta clicar no nome do arquivo na barra inferior do navegador.

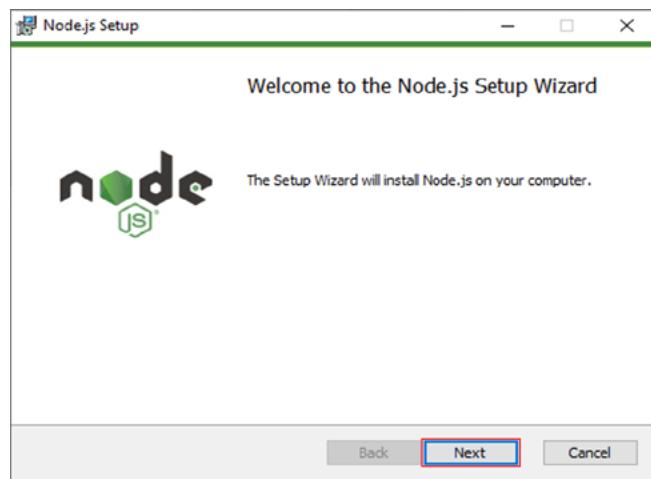


Acompanhe o passo a passo da instalação.

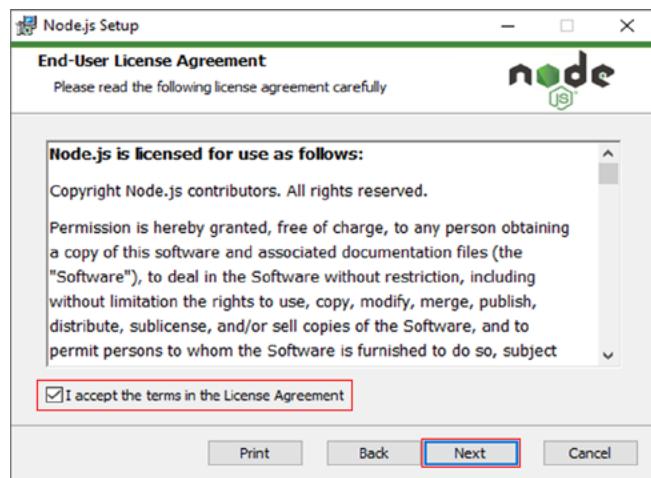
- Clique em "Executar".



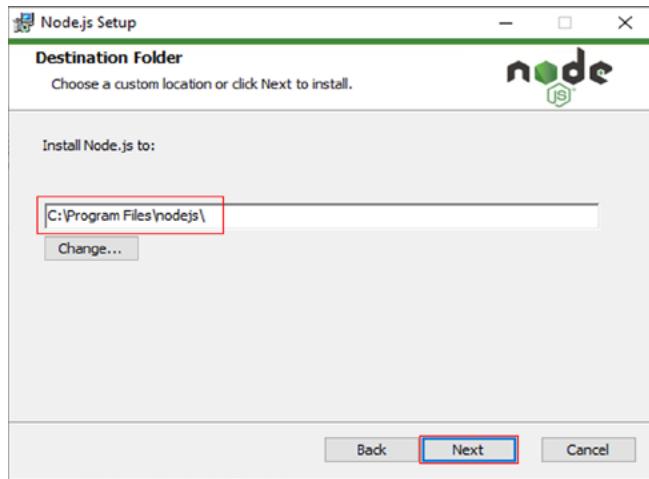
- Clique em "Next".



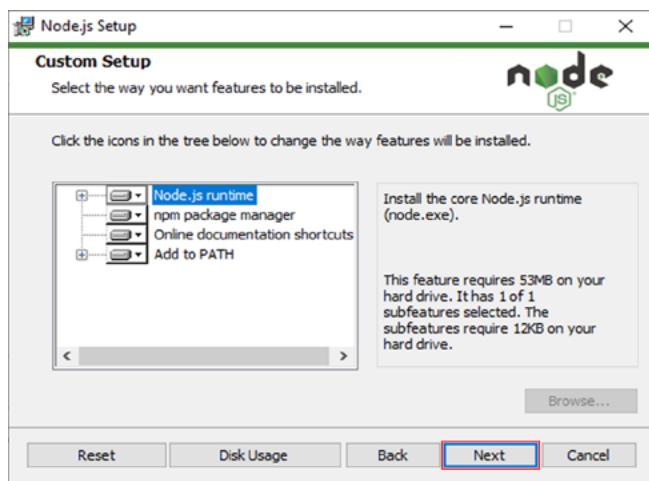
- Após a leitura, selecione a opção de aceite dos termos e clique em "Next".



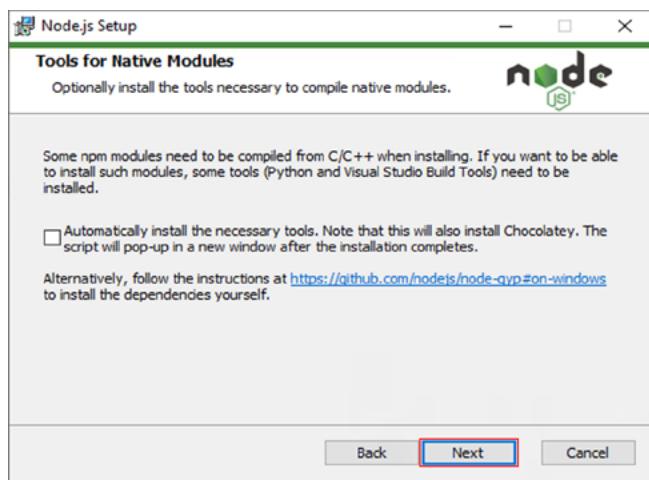
- Manteremos a instalação na pasta padrão. Clique em "Next".



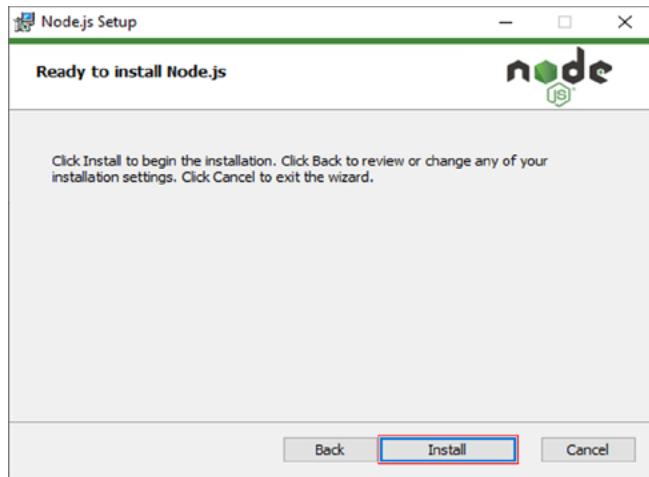
- Clique em "Next" novamente.



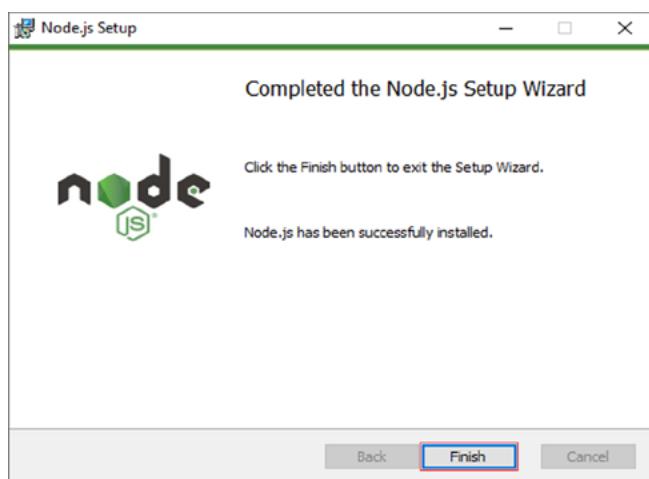
- Clique em "Next".



- Clique em "Install".



- Pronto! Clique em "Finish" para finalizar a instalação.



CRIANDO NOSSO PROJETO

Para iniciar o nosso projeto, será necessário criar uma pasta, abri-la dentro do Visual Studio Code além de criar o arquivo que será executado.

No exemplo que apresentaremos, será criada uma pasta chamada de "resolucao-problemas" que estará localizada dentro da pasta "Documentos". Logo abaixo mostraremos os passos para criar a pasta e o arquivo **index.js**.



Você sabia?

Por que index.js?



A palavra index vem do Inglês e quer dizer Índice. Traduzindo para a Internet, o arquivo index, seria a página principal, que armazena o índice (links) e as principais funções de todo o site.

O index.html é o nome de arquivo mais utilizado em sites estáticos, porém, além dele, você pode utilizar outras extensões dependendo da linguagem utilizada, por exemplo: index.php, default.php, index.asp, default.asp, etc.

Como utilizaremos a linguagem JavaScript, iremos nomear o nosso arquivo como index.js, a fim de manter esse padrão.

Veja o que faremos:



Problema/Desafio

Você deverá:

1. guardar uma lista de estudantes com o nome de cada aluno/aluna;
2. apresentar o nome de cada aluno inserido.

Somente poderá ser inserido aluno na lista, caso a quantidade de alunos não tenha ultrapassado 5.



Revisitando os conhecimentos

Conforme você viu, nós utilizamos uma linguagem de programação para resolvermos problemas do cotidiano, de um produto ou solução.

Essas linguagens possuem uma sintaxe específica, utilizando os mesmos recursos vistos até aqui.

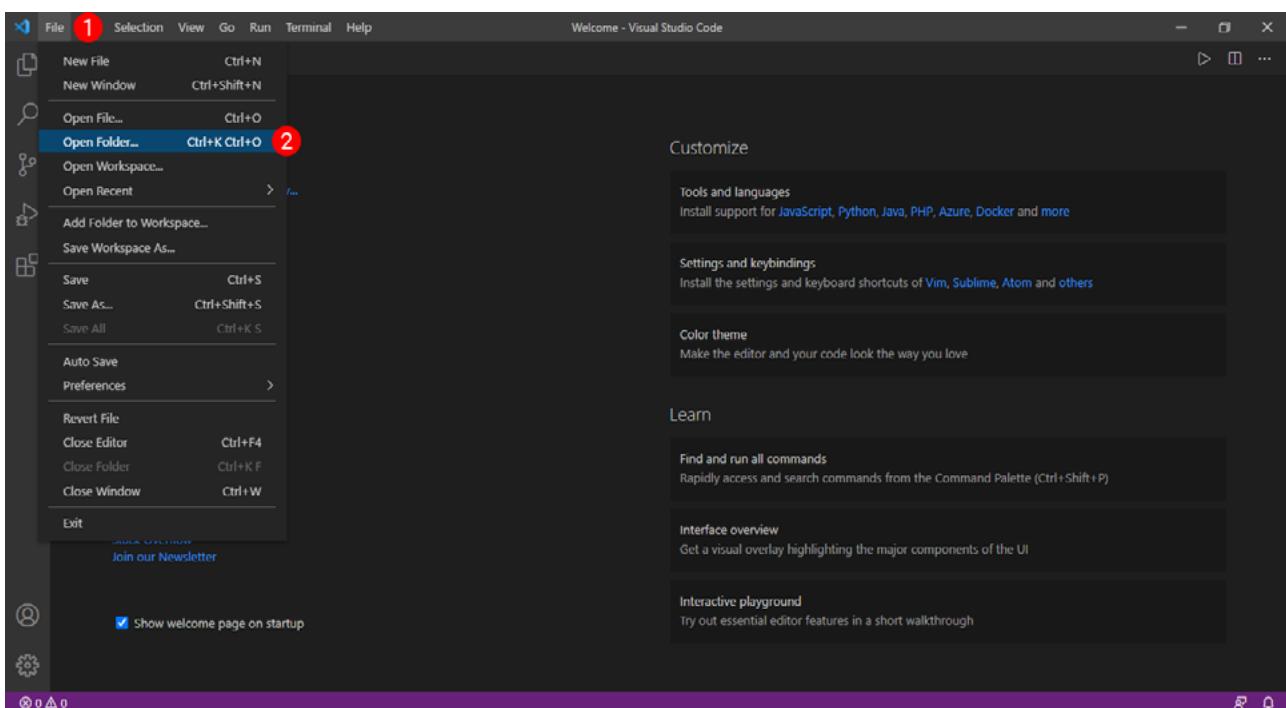
Estruturas de decisão, repetição, estruturas de dados e outros conhecimentos, como operadores lógicos e aritméticos.

Como estamos utilizando JavaScript, e ele é de uma tipagem dinâmica, esse valor poderia receber um número, por exemplo, sem a necessidade de especificar o tipo de dado.

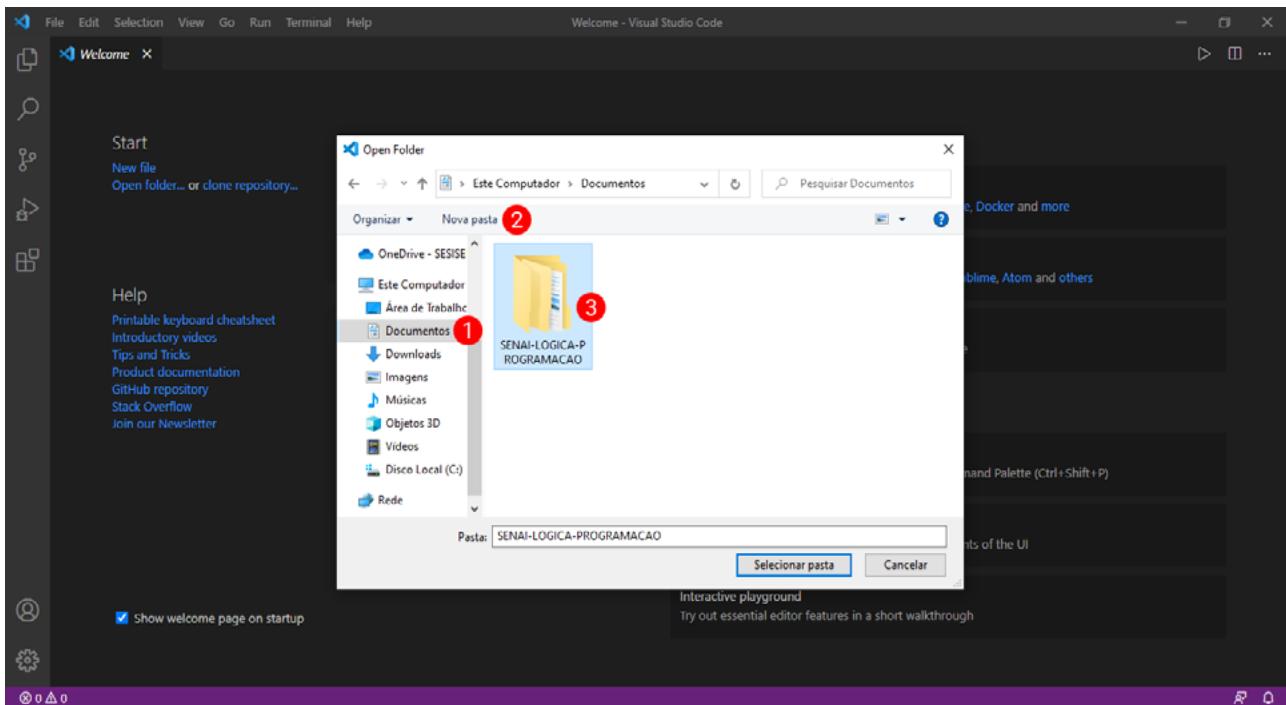
1. // variável, seu valor pode ser alterado no decorrer do programa
2. let nomeEstudante = "Helena";
3. console.log(nomeEstudante);
- 4.
5. // constante, seu valor não pode ser alterado
6. const nomeDoEstudante = "Helena";
7. console.log(nomeDoEstudante);
- 8.
9. //Em outras linguagens, portanto, você encontrará algo semelhante a:
10. String: nomeEstudante = "Helena";

CRIAÇÃO DA PASTA E DO ARQUIVO INDEX.JS NO VISUAL STUDIO CODE

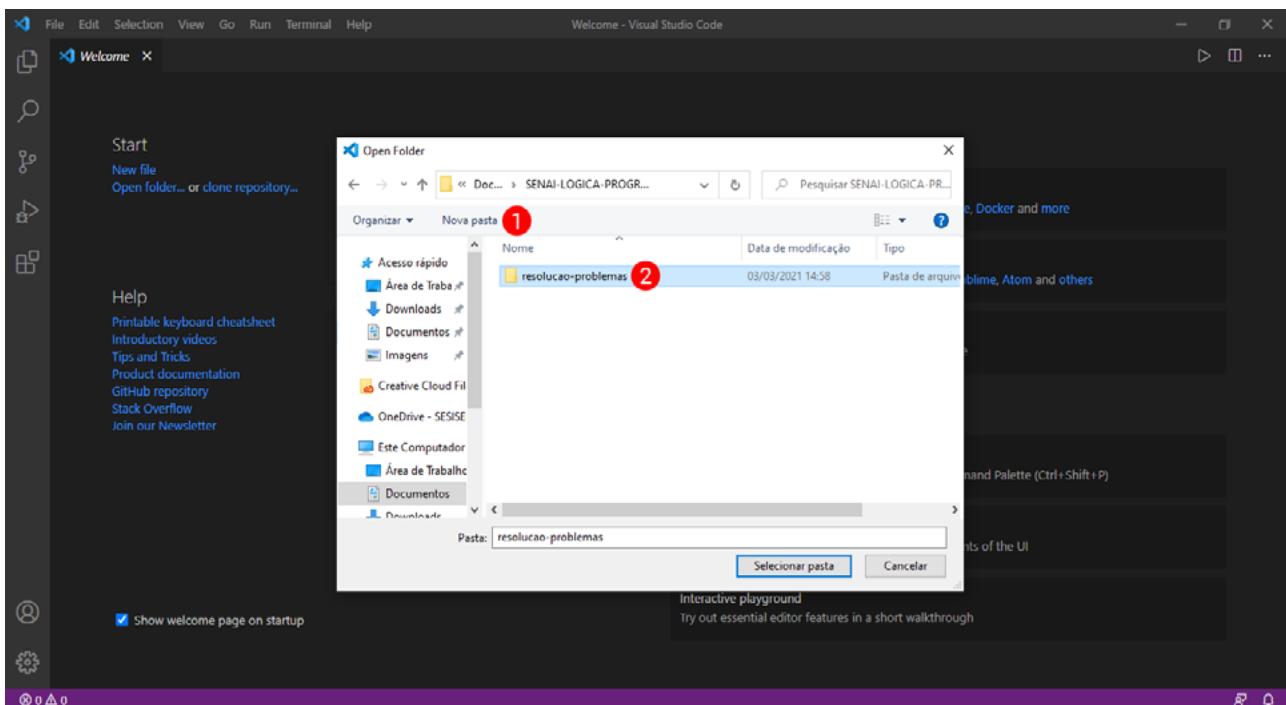
- 1. Dentro do editor Visual Studio Code, abra o menu "**File**".
- 2. Clique em "**Open Folder**".



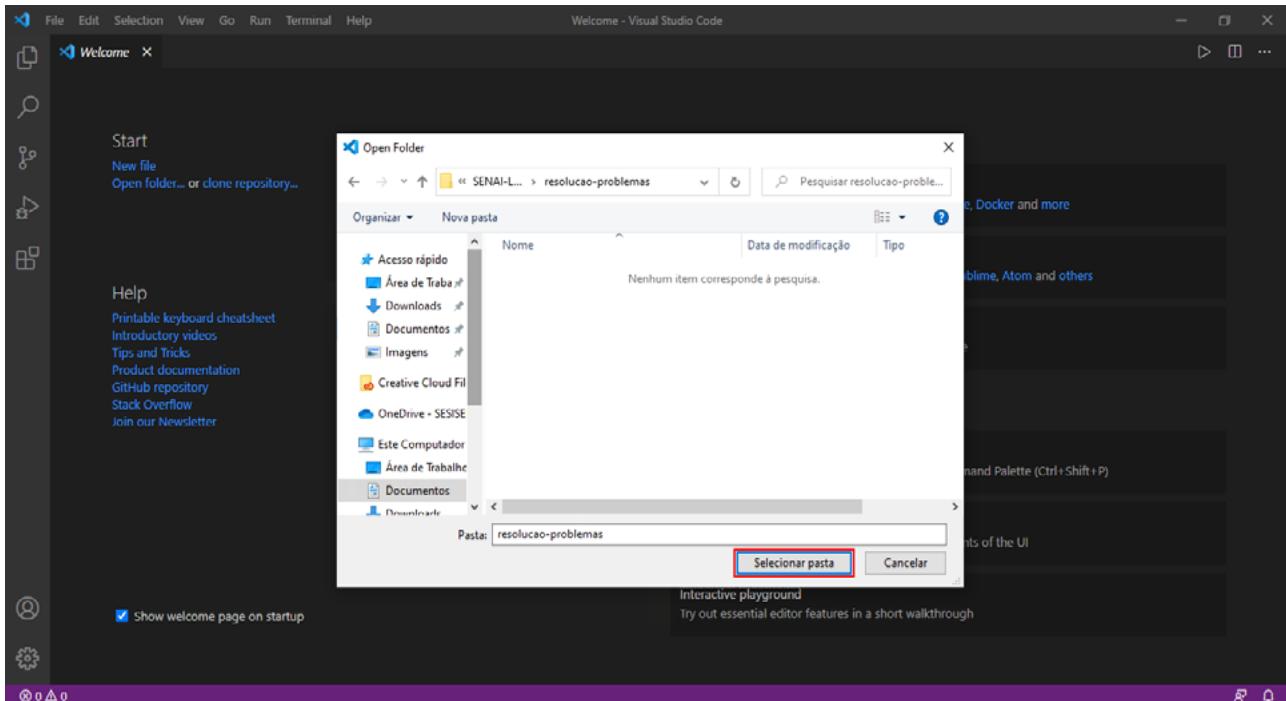
- 1. Selecione a pasta "**Documentos**".
- 2. Clique no botão "**Nova pasta**".
- 3. Nomeie a pasta com o nome "**SENAI-LOGICA-PROGRAMACAO**" e dê um duplo clique para acessá-la.



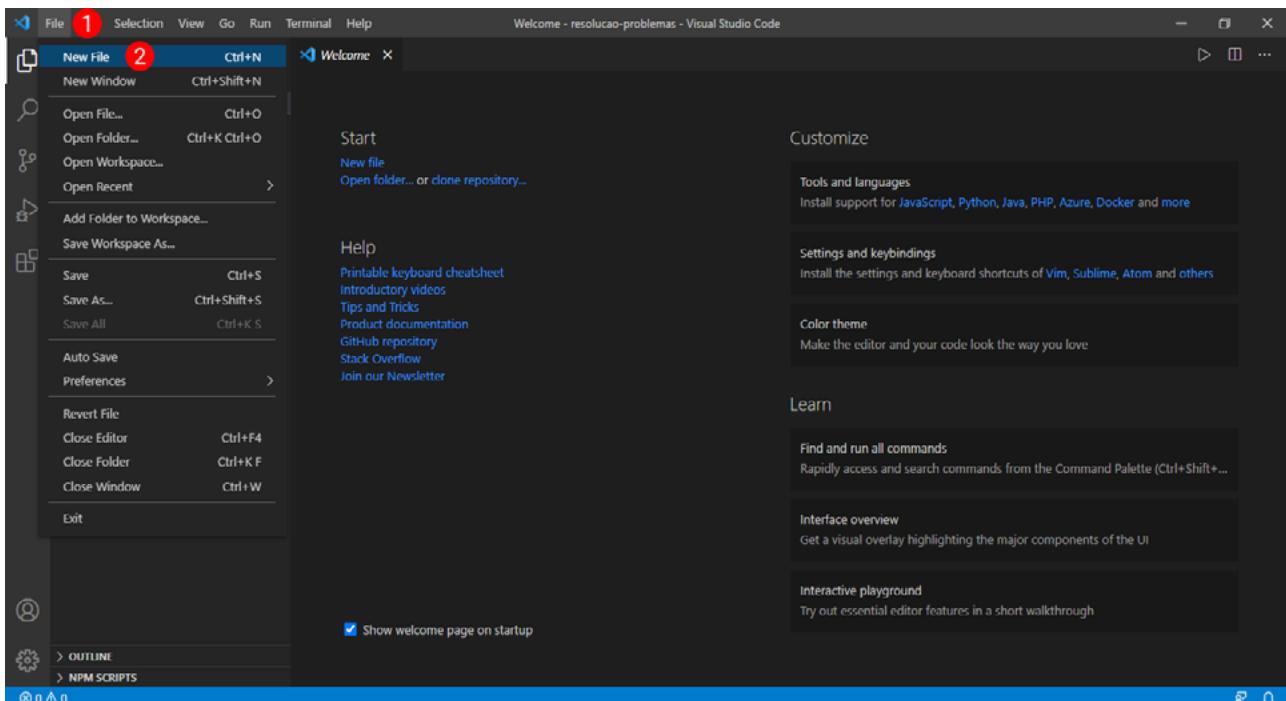
- 1. Clique no botão "**Nova pasta**".
- 2. Nomeie a pasta para "**resolucao-problemas**" e dê um clique duplo para acessá-la.



- Clique no botão "Selecionar pasta"

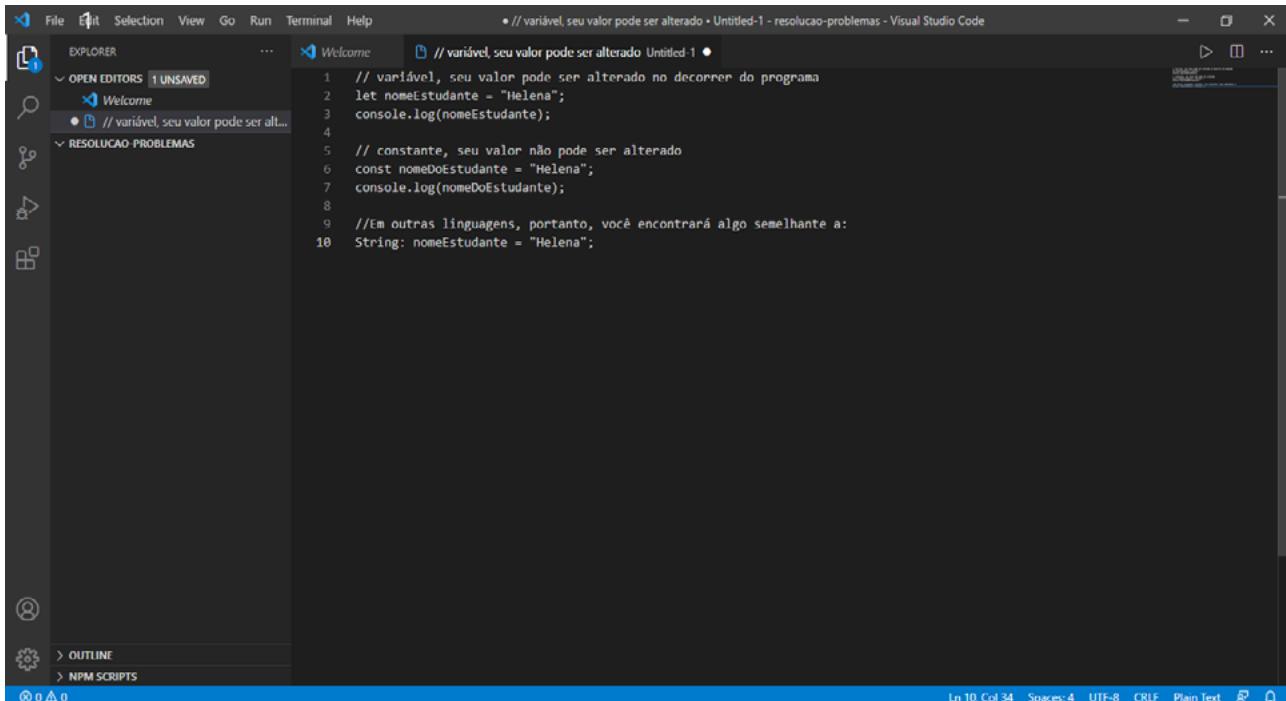


- 1. Clique novamente no menu "File".
- 2. Clique na primeira opção "New file" para criar um arquivo novo.



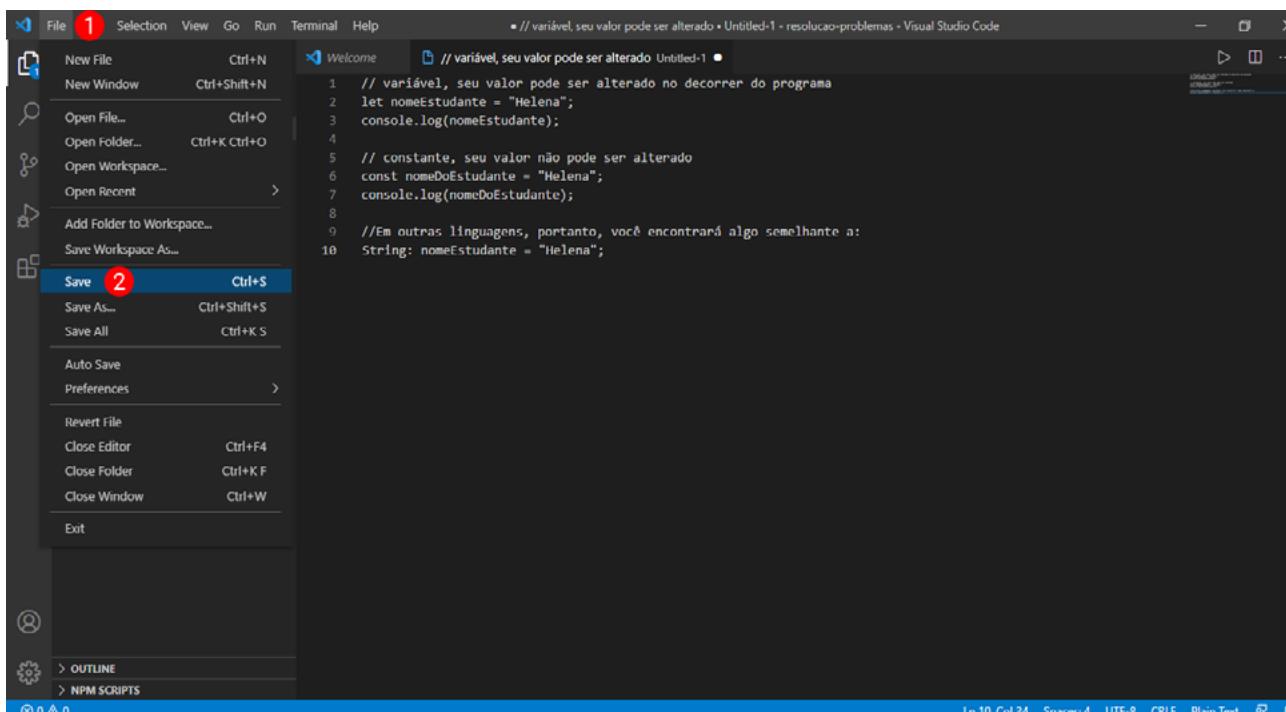
- Selecione, copie e cole o exemplo do código acima dentro da nova janela no VSCode.

Lógica de Programação - Desafio 3

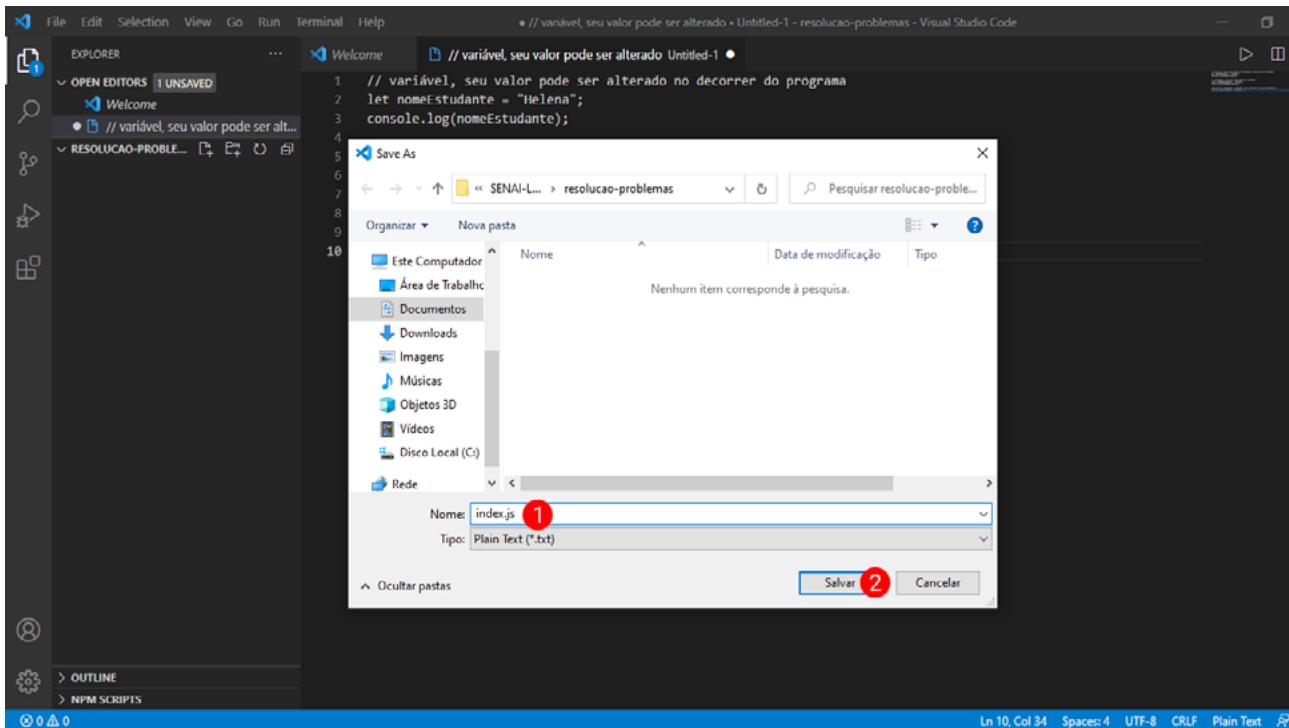


```
// variável, seu valor pode ser alterado Untitled-1 •
1 // variável, seu valor pode ser alterado no decorrer do programa
2 let nomeEstudante = "Helena";
3 console.log(nomeEstudante);
4
5 // constante, seu valor não pode ser alterado
6 const nomeDoEstudante = "Helena";
7 console.log(nomeDoEstudante);
8
9 //Em outras linguagens, portanto, você encontrará algo semelhante a:
10 String nomeEstudante = "Helena";
```

- Após colar o código do programa na nova janela do editor, clique em:
- 1. **"File"**
- 2. **"Save"**



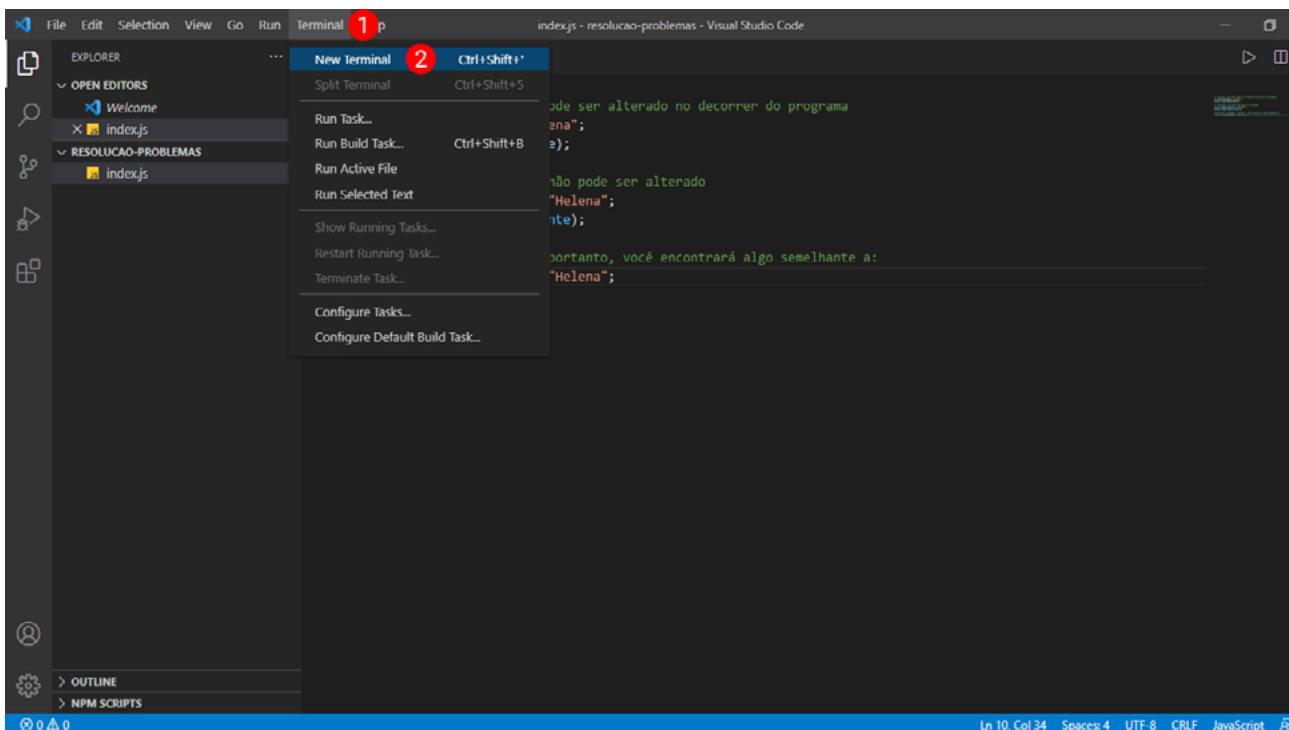
- 1. Nomeie o arquivo para "**index.js**"
- 2. Clique em "**Salvar**"



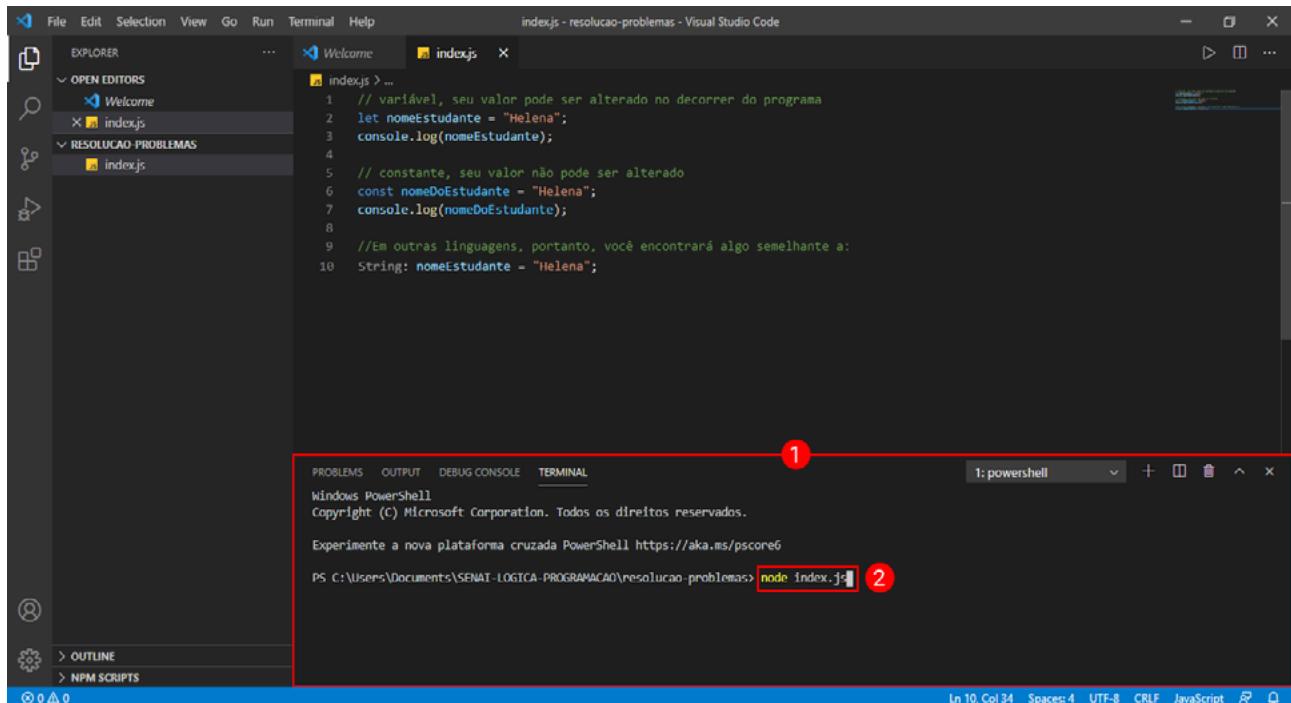
Para executar o código e ver o resultado de cada um dos itens que criamos, abra o terminal de comando na mesma pasta que foi criada a nossa estrutura inicial.

Siga o passo a passo para executá-lo:

- Após salvar o arquivo clique em:
- 1. **"Terminal"**.
- 2. **"New Terminal"**.



- 1. Aparecerá na parte inferior do editor a janela do "Terminal".
- 2. Digite o comando **node index.js** na linha de comando e pressione a tecla "Enter".



The screenshot shows the Visual Studio Code interface. In the center, there is a code editor window titled "index.js - resolucao-problemas - Visual Studio Code" containing the following JavaScript code:

```

1 // variável, seu valor pode ser alterado no decorrer do programa
2 let nomeEstudante = "Helena";
3 console.log(nomeEstudante);
4
5 // constante, seu valor não pode ser alterado
6 const nomeDoEstudante = "Helena";
7 console.log(nomeDoEstudante);
8
9 //Em outras linguagens, portanto, você encontrará algo semelhante a:
10 String: nomeEstudante = "Helena";

```

Below the code editor is the terminal window, which is highlighted with a red border. The terminal tab is selected, and it displays the command "node index.js" followed by the number "2" in a red circle, indicating the current input field. The terminal output shows the execution of the script:

```

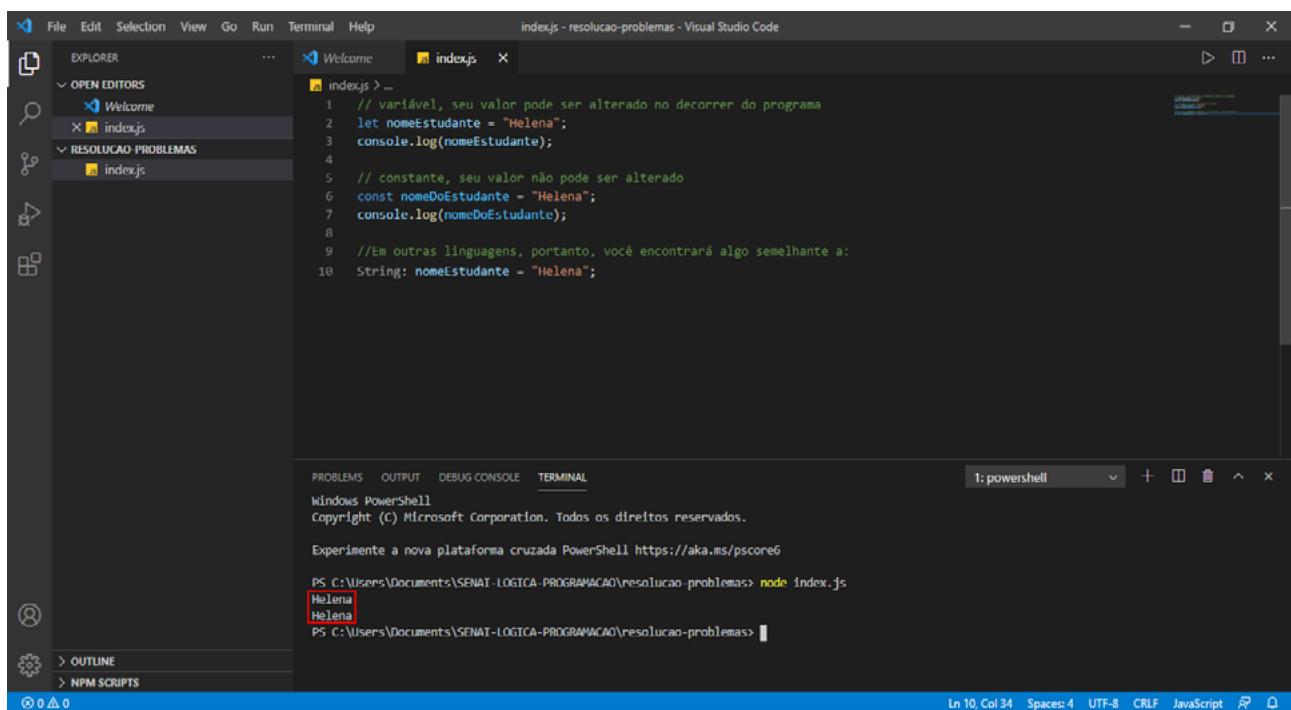
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js 2

```

- O resultado do processamento do código será exibido logo abaixo da linha de comando, conforme mostrado na imagem ao lado. Nesse caso, deverá aparecer o nome das duas primeiras variáveis com o nome **Helena**.



This screenshot is identical to the one above, showing the Visual Studio Code interface with the terminal window highlighted. The terminal output now includes the results of the script execution, with the word "Helena" highlighted in red, matching the red circle around the terminal tab in the previous screenshot.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js
Helena
Helena
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>

```

- 1. Caso o terminal apresente o erro: "**Cannot find module**"...
- 2. Clique em "**Open Folder**" e selecione a pasta onde se encontra o arquivo criado.

The screenshot shows the Visual Studio Code interface. In the terminal window, the following command was run:

```
PS C:\Users> node index.js
```

An error message is displayed:

```
Error: Cannot find module 'C:\Users\index.js'  
at Function.Module._resolveFilename (internal/modules/cjs/loader.js:880:15)  
at Function.Module._load (internal/modules/cjs/loader.js:725:27)  
at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:72:12)  
at internal/main/run_main_module.js:17:47  
code: 'MODULE_NOT_FOUND'  
requireStack: []  
}
```

The error message is highlighted with a red box and a red circle with the number 1. The 'Open Folder' button in the Explorer sidebar is also highlighted with a red box and a red circle with the number 2.

O próximo código deve armazenar uma lista de estudantes.

1. `let listaDeEstudantes = ["Helena", "Chico", "Mário"];`

Com as listas, é possível armazenar uma grande quantidade de dados e fazer a busca, adicionar novos itens, removê-los ou percorrê-los.

Utilizando as estruturas de repetição **for**, **do** e **while** estudadas no desafio anterior, é possível percorrer essa lista e apresentar o nome de cada aluno.

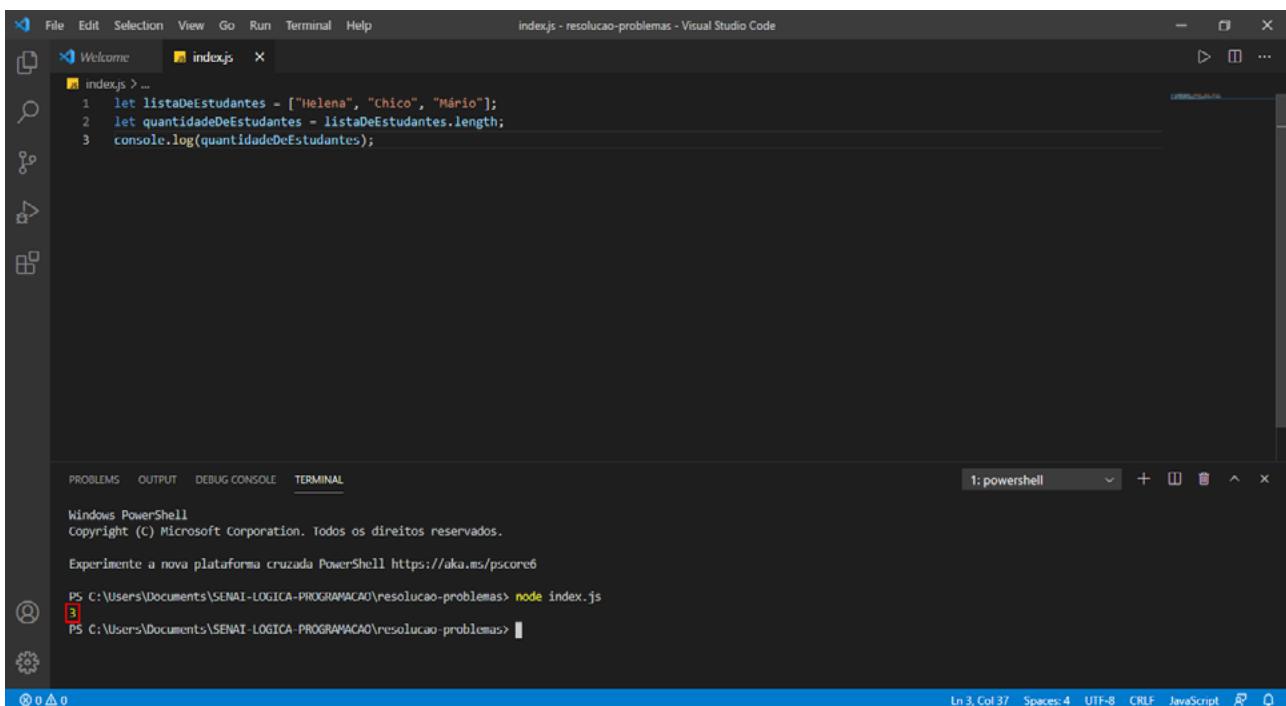
No próximo código será possível obter o resultado da quantidade de elementos que a **listaDeEstudantes** possui. Conforme comentado, listas, vetores ou outras estruturas de dados que você esteja utilizando nos proveem ações prontas para utilizarmos em nosso benefício.

Para descobrir o tamanho da lista **listaDeEstudantes**, especificada como array ou matriz, podemos fazer:

```
1. let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2. let quantidadeDeEstudantes = listaDeEstudantes.length;
3. console.log(quantidadeDeEstudantes);
```

Para simular o resultado, copie o código acima e cole no VSCode. Salve as alterações e digite na linha de comando do terminal **node index.js**.

Após a execução da rotina, o resultado exibido na janela do terminal será **3**.



Através do comando **for** é possível percorrer a partir do índice zero até o seu total, imprimindo os valores contidos a cada passo da *iteração*². Começando por Helena, passando em Chico e, por fim, Mário.

```
1. let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2. let quantidadeDeEstudantes = listaDeEstudantes.length;
3. for (let indice = 0; indice < quantidadeDeEstudantes;
   indice++) {
4.     const alunoCorrente = listaDeEstudantes[indice];
5.     console.log(alunoCorrente);
6. }
```

² Iteração é o processo chamado na programação de repetição de uma ou mais ações. É importante salientar que cada iteração se refere a apenas uma instância da ação, ou seja, cada repetição possui uma ou mais iterações.

ELEMENTO ≠ ÍNDICE

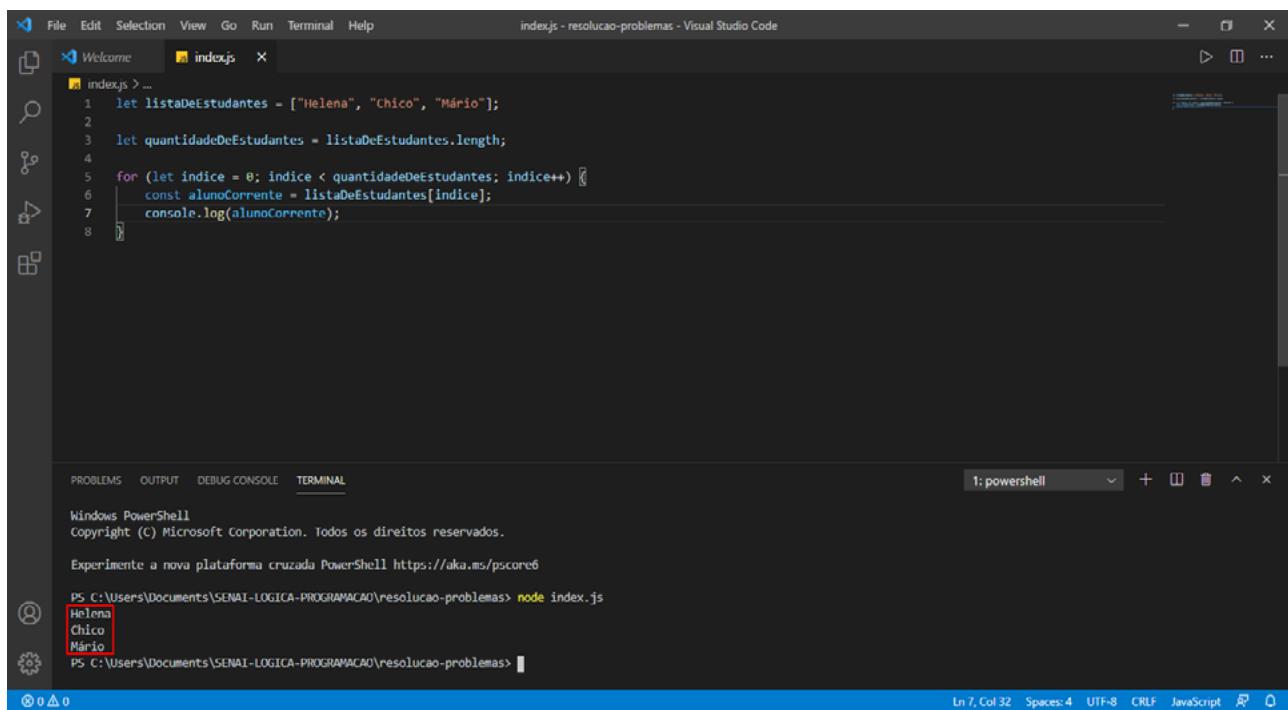
["Helena", "Chico", "Mário"] = 3

0 1 2

Um array ou matriz é uma coleção de um ou mais objetos do mesmo tipo, armazenados em endereços adjacentes de memória. Cada objeto é chamado de **elemento do array**. Da mesma forma que para variáveis simples, damos um nome ao array. O tamanho do array é o seu número de elementos. Cada elemento do array é numerado, usando um inteiro chamado de índice. Em C, a numeração começa com zero e aumenta de um em um. Assim, o último índice é igual ao número de elementos do array menos um.

Copie o código anterior e substitua no editor VSCode para executá-lo no terminal. Após colar o código e salvar as alterações, digite na linha de comando do terminal **node index.js**.

Após a execução do programa, o resultado será **Helena Chico Mário** impresso verticalmente.



```

File Edit Selection View Go Run Terminal Help
index.js - resolucao-problemas - Visual Studio Code
Welcome index.js
index.js > ...
1 let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2
3 let quantidadeDeEstudantes = listaDeEstudantes.length;
4
5 for (let indice = 0; indice < quantidadeDeEstudantes; indice++) {
6   const alunoCorrente = listaDeEstudantes[indice];
7   console.log(alunoCorrente);
8 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) Microsoft corporation. Todos os direitos reservados.
Experimente a nova plataforma cruzada PowerShell <https://aka.ms/powershell6>

PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js

Helena
Chico
Mário

PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>

Line 7, Col 32 Spaces: 4 UTF-8 CRLF JavaScript

É possível obter o mesmo resultado utilizando o comando **do**. Esta estrutura de repetição garante que o bloco de instruções seja executado no mínimo uma vez, já que a condição que controla o laço é testada apenas no final do comando.

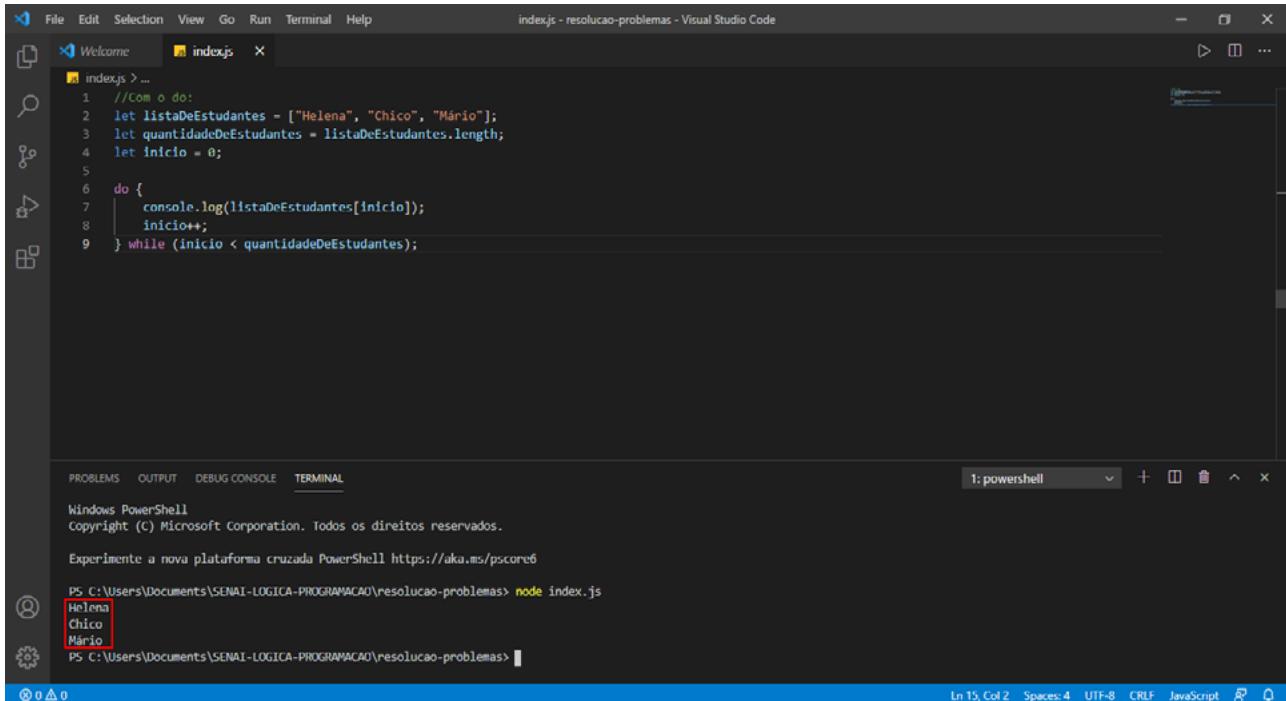
```
1. //Com o do:  
2. let listaDeEstudantes = ["Helena", "Chico", "Mário"];  
3. let quantidadeDeEstudantes = listaDeEstudantes.length;  
4. let inicio = 0;  
5.  
6. do {  
7.     console.log(listaDeEstudantes[inicio]);  
8.     inicio++;  
9. } while (inicio < quantidadeDeEstudantes);  
10.  
11. //-----//  
12.  
13. //Com o while:  
14. while (inicio < quantidadeDeEstudantes) {  
15.     console.log(listaDeEstudantes[inicio]);  
16.     inicio++;  
17. }
```

Caso queira testá-lo, copie os trechos dos códigos acima trocando as devidas estruturas e cole no VSCode para executá-lo. Salve as alterações e digite na linha de comando do terminal **node index.js**.

Após a execução do programa, o resultado será **Helena Chico Mário** impresso verticalmente.

Rotina com o comando **do**

Lógica de Programação - Desafio 3



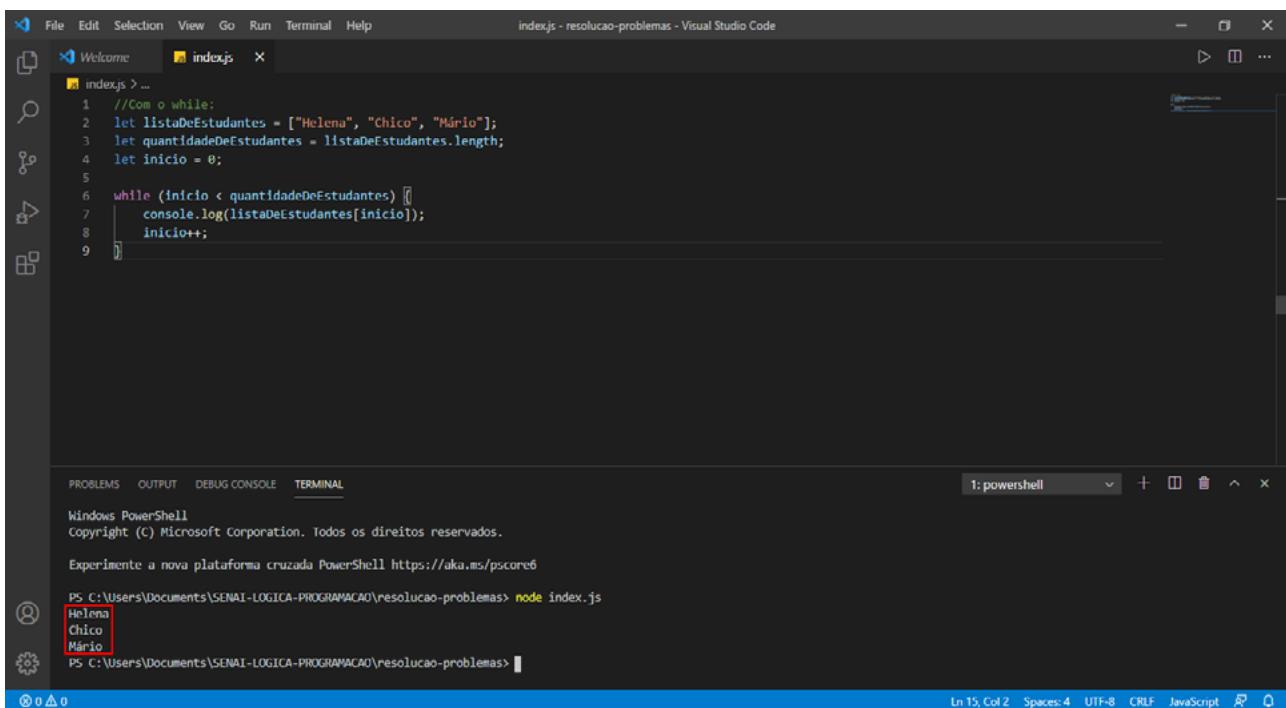
A screenshot of Visual Studio Code showing a code editor and a terminal window. The code editor has a file named 'index.js' open, containing the following code:

```
//Com o do:  
let listaDeEstudantes = ["Helena", "Chico", "Mário"];  
let quantidadeDeEstudantes = listaDeEstudantes.length;  
let inicio = 0;  
  
do {  
    console.log(listaDeEstudantes[inicio]);  
    inicio++;  
} while (inicio < quantidadeDeEstudantes);
```

The terminal window below shows the output of running the script with node index.js. The names Helena, Chico, and Mário are printed one by one, each on a new line.

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos os direitos reservados.  
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6  
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js  
Helena  
Chico  
Mário  
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>
```

Rotina com o comando while



A screenshot of Visual Studio Code showing a code editor and a terminal window. The code editor has a file named 'index.js' open, containing the following code:

```
//Com o while:  
let listaDeEstudantes = ["Helena", "Chico", "Mário"];  
let quantidadeDeEstudantes = listaDeEstudantes.length;  
let inicio = 0;  
  
while (inicio < quantidadeDeEstudantes) {  
    console.log(listaDeEstudantes[inicio]);  
    inicio++;  
}
```

The terminal window below shows the output of running the script with node index.js. The names Helena, Chico, and Mário are printed one by one, each on a new line.

```
Windows PowerShell  
Copyright (C) Microsoft corporation. Todos os direitos reservados.  
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6  
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js  
Helena  
Chico  
Mário  
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>
```

Saiba mais

A diferença entre o comando **do...while** e o **while** é justamente o local onde a condição que controla o laço é testada.



No **do...while** o bloco de comandos é executado pelo menos uma vez de forma obrigatória, independente do resultado da expressão lógica.

No comando **while** a condição é testada antes do bloco de instruções e, caso a condição seja falsa, a repetição não será executada.

No próximo exemplo, o laço em **for** exibirá a contagem de 1 até 10.

À variável **numero** será atribuído o valor inicial **1**, e a cada iteração será incrementado **1** a essa variável até que a condição **<=10** seja atendida.

1. `for (let numero = 1; numero <= 10; numero++) {`
2. `console.log(numero);`
3. `}`

Para testá-lo, copie o código acima e cole no VSCode para executá-lo. Salve as alterações e digite na linha de comando do terminal **node index.js**.

Após a execução do programa, o resultado será uma lista impressa verticalmente com os números de **1 a 10**.

```
indexjs - resolucao-problemas - Visual Studio Code
File Edit Selection View Go Run Terminal Help
indexjs > ...
1 for (let numero = 1; numero <= 10; numero++) {
2   console.log(numero);
3 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js
1
2
3
4
5
6
7
8
9
10

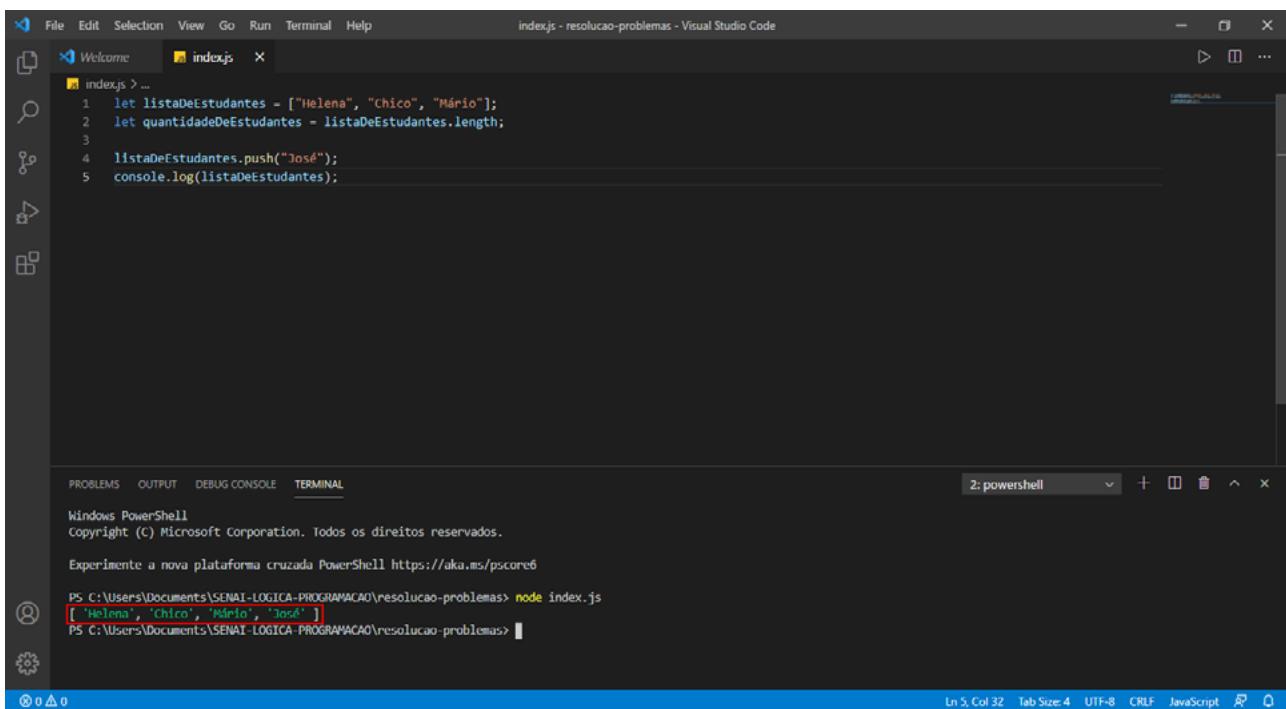
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>
```

As estruturas permitem que além dos recursos de verificar o tamanho da lista, também seja possível inserir um novo item ou remover.

1. let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2. let quantidadeDeEstudantes = listaDeEstudantes.length;
- 3.
4. listaDeEstudantes.push("José");
5. console.log(listaDeEstudantes);

Para realizar o teste, copie o código acima e cole no VSCode para executá-lo. Salve as alterações e digite na linha de comando do terminal **node index.js**.

Após a execução do programa, o resultado será **['Helena', 'Chico', 'Mário', 'José']** impresso na horizontal.



```
File Edit Selection View Go Run Terminal Help index.js - resolucao-problemas - Visual Studio Code
index.js > ...
1 let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2 let quantidadeDeEstudantes = listaDeEstudantes.length;
3
4 listaDeEstudantes.push("José");
5 console.log(listaDeEstudantes);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/powershell
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js
[ 'Helena', 'Chico', 'Mário', 'José' ]
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas>

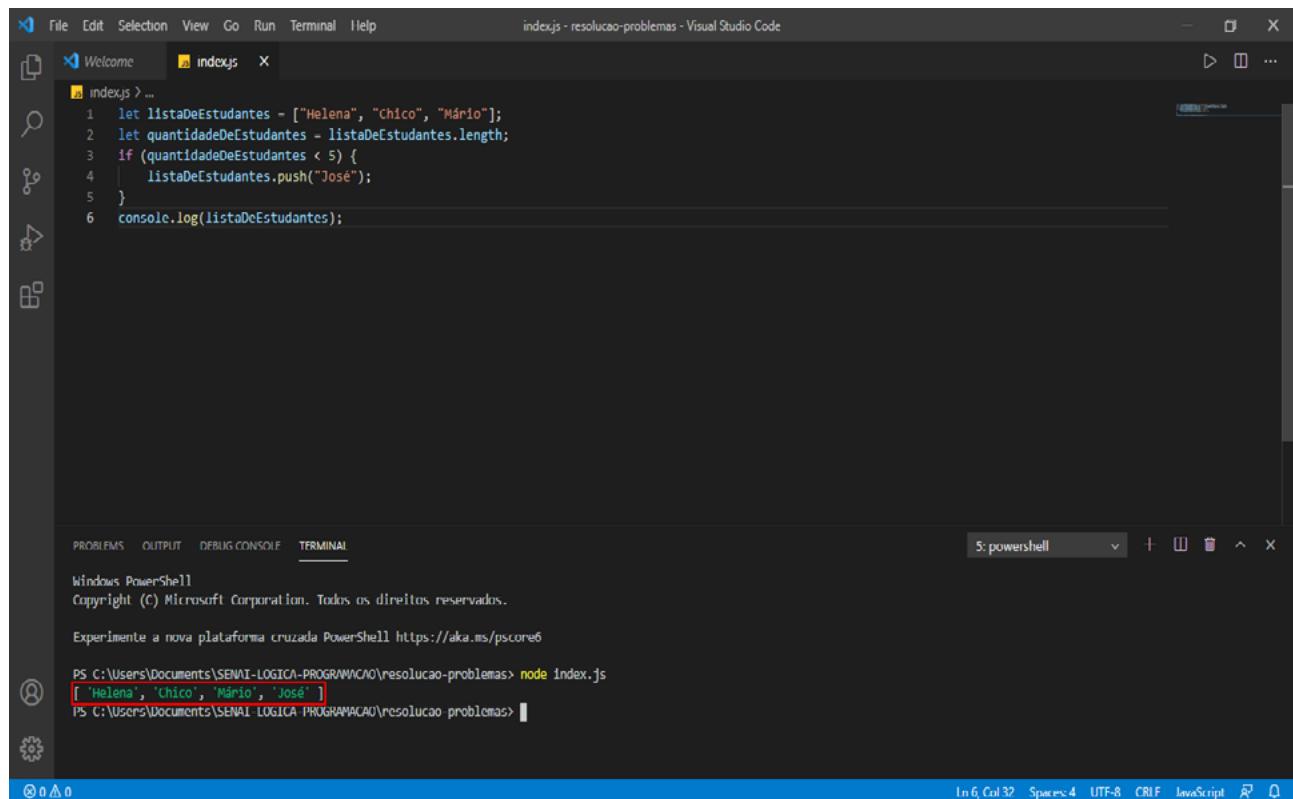
Ln 5, Col 32 Tab Size: 4 UTF-8 CRLF JavaScript
```

É possível também, incluir condicionais para verificar se é possível ou não adicionar um novo item. Por exemplo: se a lista possuir até quatro elementos, ainda será possível a inserção de um item.

1. let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2. let quantidadeDeEstudantes = listaDeEstudantes.length;
3. if (quantidadeDeEstudantes < 5) {

```
4.     listaDeEstudantes.push("José");
5. }
6. console.log(listaDeEstudantes);
```

Após a execução do programa, o resultado será **['Helena', 'Chico', 'Mário', 'José']** impresso na horizontal.



```
File Edit Selection View Go Run Terminal Help indexjs - resolucao-problemas - Visual Studio Code
Welcome indexjs

indexjs > ...
1 let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2 let quantidadeDeEstudantes = listaDeEstudantes.length;
3 if (quantidadeDeEstudantes < 5) {
4     listaDeEstudantes.push("José");
5 }
6 console.log(listaDeEstudantes);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao-problemas> node index.js
['Helena', 'Chico', 'Mário', 'José']
PS C:\Users\Documents\SENAI-LOGICA-PROGRAMACAO\resolucao_problemas>

In 6, Col 32  Spaces: 4  UTF-8  CRLF  JavaScript  ⚡  🌐
```

Adicionando dois nomes à lista, é atingido o número máximo de elementos para acrescentar um novo nome. Logo, será executado o bloco do senão da função, imprimindo o resultado que "Não é possível inserir mais alunos nessa turma".

```
1. let listaDeEstudantes = ["Helena", "Chico", "Mário",
"José", "Maria"];
2. let quantidadeDeEstudantes = listaDeEstudantes.length;
3. if (quantidadeDeEstudantes < 5) {
4.     listaDeEstudantes.push("João");
5.     console.log(listaDeEstudantes);
6. } else {
7.     console.log("Não é possível inserir mais alunos
nessa turma");
8. }
```

Para realizar o teste, copie o código acima e cole no VSCode para executá-lo. Salve as alterações e digite na linha de comando do terminal **node index.js**.

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file operations like Open, Save, and Close. The main editor window shows a file named 'index.js' with the following code:

```
index.js > ...
1 let listaDeEstudantes = ["Helena", "Chico", "Mário"];
2 let quantidadeDeEstudantes = listaDeEstudantes.length;
3 if (quantidadeDeEstudantes < 5) {
4     listaDeEstudantes.push("José");
5 }
6 console.log(listaDeEstudantes);
```

Below the editor is a tab bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected, showing a Windows PowerShell session. The session starts with the PowerShell logo and copyright information. It then shows the command 'node index.js' being run, followed by the output: '['Helena', 'Chico', 'Mário', 'José']'. The terminal window has a blue header bar with tabs for powershell, python, and bash.

NESTE DESAFIO...

LÓGICA DE PROGRAMAÇÃO| DESAFIO 03



Você aprendeu que há diversas linguagens de programação, cada uma com suas semânticas e sintaxes próprias. É fundamental manter a organização e a clareza do código, fazendo o uso correto da indentação.

Você estudou também as características, os tipos, níveis e tipos de implementação das linguagens de programação.

Por fim, para solucionar nosso desafio, você aprendeu a instalar as ferramentas para desenvolvimento JavaScript e deu início a um projeto.

PARA CONCLUIR...

PARABÉNS, VOCÊ CHEGOU AO FIM DO CONTEÚDO DE LÓGICA DE PROGRAMAÇÃO!

Durante seus estudos, você percebeu que o raciocínio lógico usado na Lógica de Programação faz parte do seu dia a dia. Em muitas situações você cria e executa algoritmos praticamente o tempo todo, mesmo sem perceber.

Você aprendeu que, para levar a lógica do dia a dia para a codificação, deve fazer uso das técnicas de programação. Em outras palavras, os fluxogramas e o conhecimento em linguagens de programação são a base para um algoritmo de software organizado e eficiente.

Sobre linguagens de programação, você estudou os tópicos comuns a todas elas, como estruturas de decisão e repetição, operadores, além de conceitos importantes como variáveis e constantes, tipos e estrutura de dados.

Você observou também que conhecer algumas características próprias de cada linguagem de programação, como semântica e sintaxe, tipo, nível e implementação é vital para fazer uma boa escolha, dentre as diversas opções existentes. Além disso, você conheceu importantes recursos que tornam o trabalho do programador mais fácil e rápido, como bibliotecas, frameworks e APIs.

Por fim, para solucionar os desafios propostos, você aprendeu a instalar duas ferramentas de desenvolvimento JavaScript.

Você evoluiu muito em seu conhecimento e ainda há muito mais para aprender.

Continue estudando e praticando, sempre!

Bom trabalho e até breve!

REFERÊNCIAS

DEVMEDIA. **Os 4 pilares da Programação Orientada a Objetos.** Disponível em: <<https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-aos-objetos/9264>>. Acesso em: 18 fev. 2021.

DEVS CHANNEL. **Operadores.** Disponível em: <<https://www.devschannel.com/logica-de-programacao/operadores>>. Acesso em: 30 jan. 2021.

DIGITAL HOUSE. **15 frameworks mais usados em programação que você precisa conhecer.** Disponível em: <<https://www.digitalhouse.com.br/blog/frameworks-mais-usados-em-programacao#:~:text=Framework%20%C3%A9%20um%20conjunto%20de,no%20desenvolvimento%20de%20um%20projeto>>. Acesso em: 18 fev. 2021.

FARIAS, Ricardo. **Estrutura de dados e algoritmos.** 2014. Disponível em: <[https://www.cos.ufrj.br/~rfarias/cos121/aula_10.html#:~:text=Uma%20pilha%20%C3%A9%20uma%20lista,dados%20entram%20e%20saem%20dela.&text=Na%20fila%20os%20elementos%20entram,\(%E2%80%9Cpela%20frente%E2%80%9D\).>](https://www.cos.ufrj.br/~rfarias/cos121/aula_10.html#:~:text=Uma%20pilha%20%C3%A9%20uma%20lista,dados%20entram%20e%20saem%20dela.&text=Na%20fila%20os%20elementos%20entram,(%E2%80%9Cpela%20frente%E2%80%9D).). Acesso em: 28 jan. 2021.

MACEDO, Diogo. **Conversão de Linguagens:** Tradução, Montagem, Compilação, Ligação e Interpretação. Disponível em: <[https://www.diegomacedo.com.br/conversoes-de-linguagens-traducao-montagem-compilacao-ligacao-e-interpretacao#:~:text=O%20processo%20de%20tradu%C3%A7%C3%A3o%20do,programa%20chamado%20Compilador%20\(Compiler\).>">https://www.diegomacedo.com.br/conversoes-de-linguagens-traducao-montagem-compilacao-ligacao-e-interpretacao#:~:text=O%20processo%20de%20tradu%C3%A7%C3%A3o%20do,programa%20chamado%20Compilador%20\(Compiler\).>](https://www.diegomacedo.com.br/conversoes-de-linguagens-traducao-montagem-compilacao-ligacao-e-interpretacao#:~:text=O%20processo%20de%20tradu%C3%A7%C3%A3o%20do,programa%20chamado%20Compilador%20(Compiler).). Acesso em: 18 fev. 2021.

MATHEUS, Diogo. **Tipos de dados no PHP.** 12 de set. de 2011. Disponível em: <<http://www.diogomatheus.com.br/blog/php/tipos-de-dados-no-php/>>. Acesso em: 28 jan. 2021.

MDN WEB DOCS. **BigInt.** 28 de mai. de 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/BigInt>. Acesso em: 28 jan. 2021.

MDN WEB DOCS. **Sintaxe e tipos.** 29 de out. de 2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals>. Acesso em: 28 jan. 2021.

MDN WEB DOCS. **Switch.** 12 de mar. de 2020. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/switch>>. Acesso em: 29 jan. 2021.

SENAI. **Série Tecnologia da Informação**(Lógica da programação). 2018. Curso online.

ZANELATO, Jessica. **Lógica da programação** - estruturas de repetição. 25 de fev. de 2018. Disponível em: <<https://podprogramar.com.br/logica-de-programacao-estruturas-de-repeticao/>>. Acesso em: 29 jan. 2021.

Créditos

CONFEDERAÇÃO NACIONAL DA INDÚSTRIA -CNI

Robson Braga de Andrade
Presidente

DIRETORIA DE EDUCAÇÃO E TECNOLOGIA - DIRET

Rafael Esmeraldo Lucchesi Ramacciotti
Diretor de Educação e Tecnologia

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL - SENAI - CONSELHO NACIONAL

Robson Braga de Andrade
Presidente

SENAI - Departamento Nacional

Rafael Esmeraldo Lucchesi Ramacciotti
Diretor-Geral

Gustavo Leal Sales Filho
Diretor de Operações

SENAI – DEPARTAMENTO NACIONAL UNIDADE DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA – UNIEP

Felipe Esteves Morgado
Gerente Executivo

Luiz Eduardo Leão
Gerente de Tecnologias Educacionais

*Anna Christina Theodora Aun de Azevedo
Nascimento*
Adriana Barufaldi
Bianca Starling Rosauro de Almeida
Laise Caldeira Pedroso
Coordenação Geral de Desenvolvimento dos
Recursos Didáticos Nacionais

SENAI - DEPARTAMENTO REGIONAL DE SÃO PAULO

Ricardo Figueiredo Terra
Diretoria Regional

Cassia Regina Souza da Cruz
Gerência de Educação

Izabel Rego de Andrade
Diretora da Escola de Educação Online

Adilson Moreira Damasceno
Audrey Castellani Aldecoa
Melissa Rocha Gabarrone
Coordenação

Adriana de Souza Farias
Adriana Valéria Lucena
Aldo Toma Junior
Danielli Guirado
Fernanda Pereira
Jeferson Paiva
Mait Paredes Antunes
Escritório de Projetos

Claudia Baroni Savini Ferreira
Camila Zanella Luckmann
Fernanda Correia Silva
João Francisco Correia de Souza
Juliana de Souza
Phillipe Rocca Datovo
Bruno Sancho Brandão
Clarice da Silva Elias
Cristiane de Barros Rodrigues Favareto
Cristina Yurie Takahashi
Fernanda Gehrke
Flávia dos Santos Silveira
José Luis de Freitas
Katya Martinez Almeida
Luciano Alves Junior
Marcelo Mauricio da Silva
Regina Kambara Hirata
Design de Aprendizagem

Helena Strada Franco de Souza

Vitoria Stefany Bessera Pinho

Elaboração de Conteúdo

Arthur Antunes Ferreira

Felipe Santos de Jesus

Rolfi Cintas Gomes Luz

Sostenes Vieira Dos Santos

Revisão de Conteúdo

Paula Cristina Bataglia Buratini

Allan Marcondes de Oliveira

Caio Marques Rodrigues

Ezequiel Regino Monção

Igor Freitas

Isabella Ferreira

Lucas André Silva de Melo

Matheus Antônio de Guimarães Elegânci

Pedro Lehi Rodrigues Muniz

Cleriston Ribeiro de Azevedo

Fabiano José de Moura

Juliana Rumi Fujishima

Design Digital

Camila Ciarini Dias

Getulio Azevedo Alves

Ederson Guilherme Antonio Silva

Juliana Souza e Silva

Rafael Marques Pimenta

Tiago Florencio da Silva

Produção Audiovisual

Rafael Santiano Apolinário

Douglas Lacerda da Conceição

Gustavo Vilela Santos

Luana Dorizo de Melo

Wesley Silveira

Desenvolvimento Tecnológico