# Aula 01-Introdução ao Java

## Introdução

Olá, meus estudantes, sejam bem-vindos à nossa aula de Algoritmo e Programação em Java do Instituto da Oportunidade Social. Nessa aula, você vai conhecer um pouco da história do Java, aprender sobre as ferramentas básicas do Java e fazer uma introdução à lógica.

## Sobre o Java

Java é uma linguagem de programação orientada a objetos, de alto nível e baseada em classes, que foi projetada para ter poucas dependências de implementação. Ele é uma linguagem de programação de propósito geral destinada a permitir que os desenvolvedores programem o código uma vez e possam executar em qualquer lugar, ou seja, o código Java compilado pode ser executado em todas as plataformas que suportam Java sem a necessidade de recompilação.

A linguagem Java foi desenvolvida por James Gosling e sua equipe da Sun Microsystems e lançada em meados de 1995. Ela é baseada nas linguagens C e C++. Originalmente, os compiladores Java, máquinas virtuais e bibliotecas de classes foram lançados pela Sun sob licenças proprietárias. Atualmente, a Sun refez as licenças e a maioria das suas tecnologias está sob a licença GPL-2.0 (General Public License, version 2). Isso contribuiu para a popularização do Java e, hoje, de acordo IEEE Spectrum, ele ocupa o segundo lugar no ranking de Top Programming Languages 2021 (Figura 1), disponível em: <https://spectrum.ieee.org/top-programming-languages-2021>.



Figura 1. Top Programming Languages 2021 do IEEE Spectrum.

## Configuração do Java

Antes de começar a implementar seus programas e começar a desenvolver aplicações em Java, você precisa realizar algumas configurações no seu computador para que tudo execute perfeitamente como desejado. A primeira configuração é a instalação do JDK, mas antes vamos entender a diferença entre JDK, JRE e JVM.

### JDK vs JRE vs JVM

JDK (Java *Development Kit*) é um kit de desenvolvimento Java usado para implementar aplicativos e aplicações em Java. Ele está disponível para os diversos sistemas operacionais, tais como: Windows, macOS, Solaris e Linux. O JDK auxilia os desenvolvedores Java a codificar e executar programas Java. Você pode instalar mais de uma versão do JDK no seu computador.

|  |
| --- |
| Porque usar o JDK?   * JDK têm ferramentas necessárias para escrever programas Java e JRE para executá-los. * Inclui um compilador, Java application launcher, Appletviewer, etc. * O compilador converte o código escrito em Java em código de bytes. * O Java application launcher abre um JRE, carrega as classes necessárias e executa seu método principal. |

Para instalar o JDK, você pode acessar o site da Oracle: <https://www.oracle.com/java/technologies/javase-downloads.html> e em caso de dúvidas na instalação seguem dois tutoriais que podem te auxiliar na instalação e na configuração do Java: Bóson Treinamentos (<https://www.youtube.com/watch?v=KeDhlDXezMs>) e Área Restrita Brasil (<https://www.youtube.com/watch?v=k8SWBprwXyQ>).

|  |  |
| --- | --- |
| No Windows, para abrir o terminal, você pode clicar no botão de pesquisa na barra de ferramentas, localizado no canto inferior esquerdo ao lado do botão do Windows. Então digitar **cmd**, como mostrado na Figura 2, clique em Prompt de comando e então digite o comando **java -version** e, depois, o comando **javac -version**. | Figura . Prompt de comando. |

Se tudo foi instalado corretamente uma mensagem semelhante a mostrada na (Figura 3) deverá aparecer no terminal.

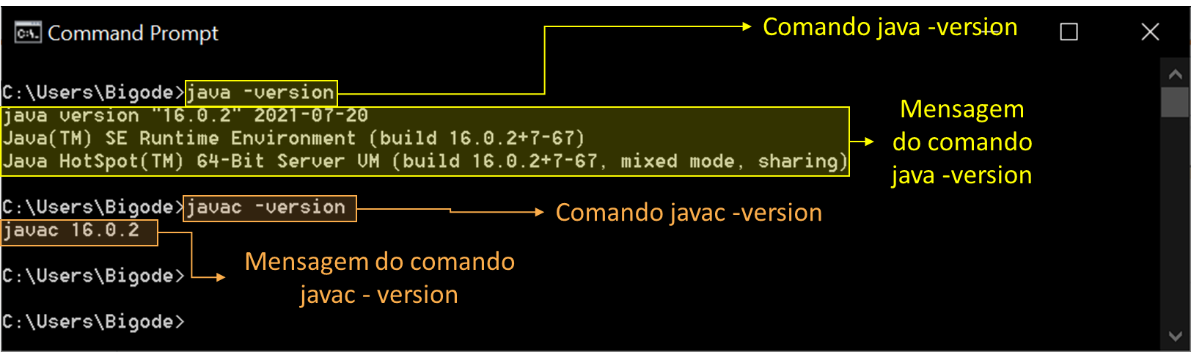


Figura . Terminal mostrando que o JDK foi instalado corretamente.

Dica: Sempre que você instalar o JDK é necessário verificar se o path foi configurado corretamente, caso os comandos java -version ou o javac -version retornem que não foram encontrados, você pode rever o vídeo do Boson Treinamentos a partir do minuto 5:45 ou o Área Restrita Brasil a partir do minuto 6:10, que você verá como configurar manualmente o caminho do JDK.

O JRE (Java *Runtime* *Environment*) é ambiente de execução Java, que contém a JVM, a biblioteca padrão do Java, a Classloader, ferramentas de configuração e plug-in para o navegador. Podemos pensar no JRE como um pedaço de software desenvolvido para executar outro software. Se você nunca programou em Java, mas já executou aplicações em Java no seu computador, você precisou de instalar o JRE para abrir esses programas. O JRE está disponível para download em: <https://www.oracle.com/java/technologies/javase-jre8-downloads.html>. Todas as versões do JDK já vêm incluído em suas ferramentas o JRE.

|  |
| --- |
| Porque usar o JRE?   * JRE contém a biblioteca de classes, a JVM e outros arquivos de suporte para executar uma aplicação desenvolvida em Java. Ele não contém as ferramentas de desenvolvimento Java, tais como: debugger, compilador, etc. * Ele usa importante módulos de classe, tais como: math, swingetc, util, lang, awt e biblioteca de execução. * Se você já executou alguma aplicação Java, então o JRE está instalado no seu sistema. |

As aplicações implementadas em Java são compiladas em bytecode, que podem ser executados em qualquer máquina virtual Java (JVM), independente da arquitetura. Então, podemos entender que a JVM é a *engine* que fornece o ambiente de execução (*runtime*) para código em Java ou aplicações desenvolvidas em Java. Ele converte o Java *bytecode* em linguagem de máquina para ser executada no computador. A JVM é parte do JRE e não é instalada separadamente.

|  |
| --- |
| Porque usar o JVM?   * JVM permite executar uma aplicação Java sem depender da plataforma (*hardware*). * Ela tem inúmeras bibliotecas, ferramentas e *frameworks*. * Uma vez que você executa um programa Java, você pode executá-lo em qualquer plataforma muitas vezes. * JVM vem com o compilador JIT (Just-in-Time), que converte o código fonte Java em linguagem de baixo nível (linguagem de máquina). |

### Como o JDK funciona?

|  |  |
| --- | --- |
| Como foi visto, o KDK com as ferramentas necessárias para implementar e executar programas em Java. A Figura 4 mostra um diagrama de blocos com alguns recursos do JDK:   * **JDK e JRE**: O JDK permite programadores criarem programas em Java, que são executados pelo JRE, que inclui a JVM e uma biblioteca de classes. * **Class Libraries**: é um grupo de biblioteca carregadas dinamicamente, que o programa do Java chama no tempo de execução. | Figura . Funcionalidade do JDK. |
| * **Compiladores**: é um programa Java que aceita arquivo de texto (código-fonte) desenvolvidos pelos programadores e compila em um arquivo Java, que geralmente é um arquivo em *bytecode*. * **Debugger**: é um programa Java que permite desenvolvedores testar e debuggar (encontrar erros) no código programado. * **JavaDoc**: é uma documentação criada pela Sun Microsystems pra o Java. | |

### Como o JRE funciona?

|  |  |
| --- | --- |
| Como citado, o JRE é responsável por garantir a execução de uma aplicação Java no seu computador. A Figura 5 mostra os componentes importantes do JRE:   * **Class loaders**: Class loader carrega várias classes que são necessárias para executar um programa Java. A JVM utiliza três tipos de class loaders: bootstrap class loader, extensions class loader e system class loader. * **Byte code verifier**: verifica o bytecode (programa compilado), que queremos executar. * **Interpreter (Interpretador)**: lê linha por linha para executar o programa. | Figura . Funcionamento do JRE. |

* **Run-time**: é um sistema usado em programação para descrever o período de tempo o qual o programa está executando.
* **Hardware**: é a plataforma para a qual o bytecode foi interpretado, ou seja, é onde o programa irá executar.

### Como o JVM funciona?

A Figura 6 mostra os componentes importantes da JVM:

* **Class Loader**: é um subsistema usado para carregar os arquivos de classes.
* **Method Area**: JVM Method Area armazena a estrutura das classes, tais como: metadados, o código para os métodos Java e constant pool.
* **Heap**: todo array de objetos e instância de variáveis são armazenadas na Heap, que é uma parte na memória compartilhada por múltiplas threads.
* **JVM language Stacks**: armazena variáveis locais e resultados parciais. Cada thread tem sua própria JVM language Stack, que foi criada no momento que a thread é criada.
* **PC Registers**: armazenam o endereço da instrução, que está sendo executada.
* **Native Method Stacks**: retem a instrução do código nativo.
* **Execution Engine**: é um tipo de, que é usada para testar o software, o hardware ou o sistema completo.
* **Native Method Interface**: é um framework de programação, que permite o código Java chamar bibliotecas e aplicações nativas, enquanto é executado.
* **Native Method Libraries**: é uma coleção de bibliotecas nativas (C, C++), que são necessárias para a execução da *engine*.

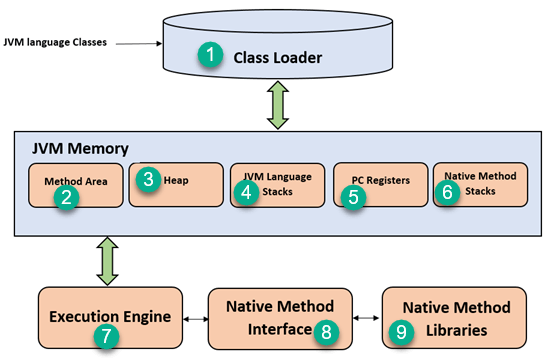


Figura . Funcionamento do JVM.

## Compilação e execução do programa Java

|  |  |
| --- | --- |
| A linguagem Java é uma linguagem de alto nível, que possui uma sintaxe próxima do inglês e todo programa feito em Java precisa ser compilado para a linguagem de máquina. A linguagem de máquina é a linguagem que os computadores entendem formada de valores binários (0s e 1s) e escrever programas nessa linguagem seria muito complicado. | No 50 Real programmers code in binary |

Por isso, sugiram as linguagens de programação, primeiro vieram as linguagens de baixo nível, que estão mais próximas da linguagem de máquina e geralmente formada de mnemônicos como o Assembly. E com o tempo, vieram as linguagens de programação de alto nível, que é ideal utilizar pois estão mais próximas do nosso idioma. Porém usar linguagens de alto nível requer o uso um tradutor para estabelecer a comunicação entre o programa desenvolvido e o computador. Tradutores de linguagem de programação são também conhecidos como compiladores ou interpretadores.

O Java é uma linguagem híbrida em termos de compilação, isso porque ela possui um processo de compilação e um processo de interpretação. A compilação é o processo de traduzir o código na linguagem Java para bytecodes e a interpretação é o processo de interpretar os bytecodes pela máquina virtual Java (JVM). Como o Java é multiplataforma o processo de interpretação realizado pela JVM é necessário porque cada sistema operacional precisa ter uma implementação da JVM (Figura 7).

Timeline

Description automatically generated with medium confidence

Figura 7. Processo de compilação e interpretação de um código Java.

## Introdução à logica

|  |
| --- |
| “Todo mundo deveria aprender a programar um computador porque isso ensina você a pensar.” Steve Jobs |

Lógica pode ser definida com a “arte de bem pensar”. Ela está diretamente relacionada à coerência e racionalidade, ou seja, está ligado ao que é correto. Resumidamente, podemos dizer que lógica é colocar “ordem no pensamento”. Quando expressamos pensamentos de forma correta, estamos sendo lógicos. Portanto, a lógica se relaciona também com a “correção do pensamento”. A lógica, então, determina quais operações são válidas ou não.

A lógica não envolve somente números, ela pode ser um conjunto de regras racionais para obtenção de um conhecimento, ou seja, passos a seguir para aprende algo novo. A lógica auxilia a formular melhor as suas ações diante de problemas. Através da lógica é possível interpretar um problema complexo e procurar entendê-lo melhor. Em computação o pensamento lógico é indispensável, sem lógica não há programação.

|  |
| --- |
| Dica: para entender melhor um pouco mais sobre lógica, você pode assistir o vídeo do canal Acima da média, disponível no link: <https://www.youtube.com/watch?v=8kP3vuZgv3k> |

### Algoritmo

Um algoritmo é uma sequência finita de ações (instruções) encadeadas, que seguem uma determinada lógica e tem por objetivo solucionar um problema. Algoritmos são como uma sequência ordenada de passos, que deve ser seguida para a realização de uma tarefa, por exemplo: como ir para a escola, comprar um carro, fazer uma receita de bolo, etc.

|  |
| --- |
| Definição: instrução é a informação que indica a um computador uma ação elementar a ser executada. |

O exemplo mais clássico de algoritmo é como fazer um bolo? Você precisa de diversos ingredientes, tais como: farinha, ovos, manteiga, açúcar, fermento etc., mas o algoritmo te fala como fazer o bolo, ou seja, qual ordem devemos seguir para ter sucesso na execução do bolo.

Podemos entender que o algoritmo tem por base o mesmo funcionamento do computador, ou seja, os algoritmos têm uma estrutura sequencial como a mostrada na Figura 25. Essa estrutura sequencial significa que cada instrução deve ser executada em sequência. Sendo assim, você tem uma entrada que no exemplo abstrato da receita de bolo são os ingredientes, em seguida tem o processamento, que seria como o bolo deve ser feito (os passos a serem seguidos), e no final da sequência a saída, que é o bolo pronto.



Figura . Funcionamento de um algoritmo no computador.

### Etapas de um Algoritmo

Um algoritmo possui três fases: Valores iniciais, Sequência Lógica de instruções e Resultado. A Figura 26 mostra cada uma dessas etapas e descrição do que cada uma corresponde.

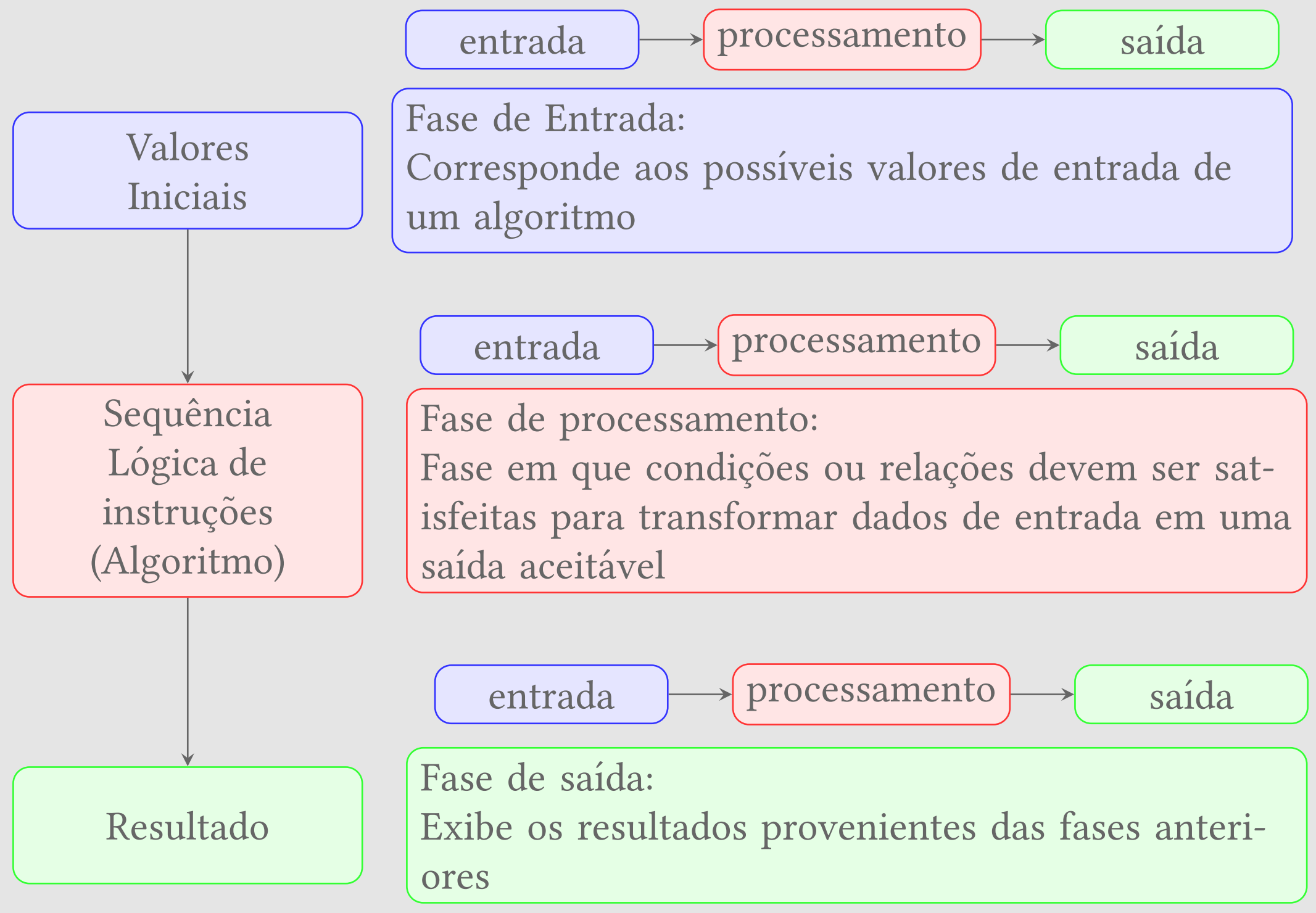


Figura . Etapas de um algoritmo.

|  |
| --- |
| Dica: se você quer saber mais sobre algoritmo, recomendo o vídeo do canal Diolinux, que explica de forma rápida e bem clara o que é um algoritmo. O link está disponível em: <https://www.youtube.com/watch?v=z1XTcKKRbKM>  E se você quiser mais inspiração para começar a programar logo, aconselho assistir o vídeo do Code.org, onde várias personalidades como Bill Gates, Mark Zuckerberg, entre outros falam sobre suas experiencias com programação. O vídeo está em inglês, mas possui legendas em português. O link do vídeo é <https://www.youtube.com/watch?v=nKIu9yen5nc> |

## Glossário

Aqui estão algumas definições e conceitos que você precisa conhecer para entender melhor o que está por trás de um programa em execução no seu computador (Fonte: Wikipédia).

* Array: é uma estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave.
* Biblioteca: é uma coleção de subprogramas utilizados no desenvolvimento de software. Bibliotecas contém código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma modular.
* Código-fonte: é o conjunto de palavras ou símbolos escritos de forma ordenada, contendo instruções em uma das linguagens de programação existentes, de maneira lógica.
* Compilador: é um programa de computador (ou um grupo de programas) que, a partir de um código fonte escrito em uma linguagem compilada, cria um programa semanticamente equivalente, porém escrito em outra linguagem, código objeto. Classicamente, um compilador traduz um programa de uma linguagem textual facilmente entendida por um ser humano para uma linguagem de máquina , específica para um processador e sistema operacional.
* Debug: é o processo de encontrar e reduzir defeitos num aplicativo de software ou mesmo em hardware. Erros de software incluem aqueles que previnem o programa de ser executado e aqueles que produzem um resultado inesperado.
* Engine:
* Framework: em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Um framework pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Ao contrário das bibliotecas, é o framework quem dita o fluxo de controle da aplicação, chamado de Inversão de Controle.
* Hardware: é um termo técnico que foi traduzido para a língua portuguesa como equipamento, e pode ser definido como um termo geral da língua inglesa, que se refere à parte física de computadores e outros sistemas microeletrônicos.
* Heap: em computação, é a forma de gerenciamento de recursos na memória do computador. O gerenciamento de memória envolve entre outros conceitos a liberação de memória e alocação dinâmica de porções de memória para as requisições dos programas.
* IDE: do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. As características e ferramentas mais comuns encontradas nos IDEs são: Editor, Compilador, Linker, Depurador (debugger), Modelagem de dados (modeling), Geração de código, Distribuição (deploy), Testes Automatizados (automated tests), Refatoração (refactoring), entre outras.
* Interpretador: é um programa de computador que lê um código fonte de uma linguagem de programação interpretada e o converte em código executável. Seu funcionamento pode variar de acordo com a implementação. Em alguns casos, o interpretador lê o código fonte linha a linha e o converte em código objeto (ou bytecode) à medida que o executa, em outros casos, converte o código fonte por inteiro e depois o executa.
* Linguagem de máquina: é a linguagem que o seu computador entende. Um programa em código de máquina consiste em uma sequência de bytes que correspondem a instruções a serem executadas pelo processador. As instruções do processador, chamadas de opcodes, são representadas por valores em hexadecimal.
* Linguagem de baixo-nível: trata-se de uma linguagem de programação que segue as características da arquitetura do computador. Assim, utiliza somente instruções que serão executadas pelo processador.
* Linguagem de alto-nível: é uma linguagem, que utiliza instruções abstratas. Ela é uma linguagem com um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana.
* Registrador: é a memória dentro da própria CPU que armazena n bits. Os registradores estão no topo da hierarquia de memória, sendo assim, é um tipo de memória mais rápida e financeiramente mais custosa. Apesar do alto custo por bit armazenado, sua velocidade de acesso é essencial para o funcionamento dos computadores modernos e, portanto, são incluídos, ainda que em menor capacidade, mesmo em processadores de baixo custo.
* Software: trata-se de uma sequência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado (informação) ou acontecimento.
* Stack (Pilha): é um tipo abstrato de dado e estrutura de dados baseada no princípio de Last In First Out (LIFO), ou seja "o último que entra é o primeiro que sai" caracterizando um empilhamento de dados. Usada para armazenar temporariamente variáveis de uma função.
* Thread: é a tarefa que um determinado programa realiza, é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrencialmente.

## Exercícios de lógica

1. Considere a seguinte situação:

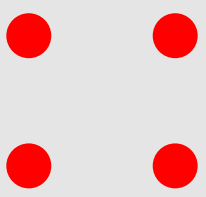
– um comboio parte de Belo Horizonte a uma velocidade de 120 Km/h em direção a São Paulo;

– ao mesmo tempo em que outro comboio parte de São Paulo a uma velocidade de 80 Km/h em direção a Belo Horizonte.

Quando os comboios se encontram. . .

– qual deles está mais próximo de Belo Horizonte? Aquele que parte de Belo Horizonte ou o que parte de São Paulo?

1. 4 pontos e 3 retas: Una os 4 pontos com apenas três retas sem levantar a ponta do lápis e acabando no ponto onde começou:



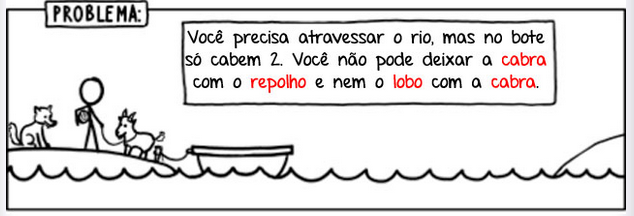
1. O lobo, a Cabra e o Repolho: Um homem precisa atravessar um rio com um barco;

– seu barco possui capacidade apenas para carregar ele e mais uma de suas 3 cargas:

– um lobo, uma cabra e um repolho.

• O que o homem deve fazer para conseguir atravessar o rio sem perder suas cargas?

– Sendo que, o lobo não pode ficar sozinho com a cabra e a cabra não pode ficar sozinha com o repolho



1. A corrida dos 8 carros: Oito carros, de marcas e cores diferentes, que nada tem a ver com suas cores da Fórmula 1, estão alinhados, lado a lado, para uma corrida. Estabeleça a ordem em que os carros estão dispostos, baseando-se nas seguintes informações:

* A FERRARI está entre os carros “vermelho” e “cinza”;
* O carro “cinza” está à esquerda da LOTUS;
* A MACLAREN é o segundo carro à esquerda da FERRARI e o primeiro à direita do carro “azul”;
* O carro da TYRREL não tem carro a sua direita e está logo depois do carro “preto”; A
* carro “preto” está entre o TYRREL e o carro “amarelo”;
* A JORDAN não tem carro algum à esquerda: está a esquerda do carro “verde”;
* A direita do carro “verde” está o MARCH;
* A LOTUS é o segundo à direta do carro “creme” e o segundo a esquerda do carro “marrom”;
* O carro WILLIAN é o segundo à esquerda da BENETTON;

1. Em que número o carro está estacionado?

Uma imagem contendo relógio

Descrição gerada automaticamente