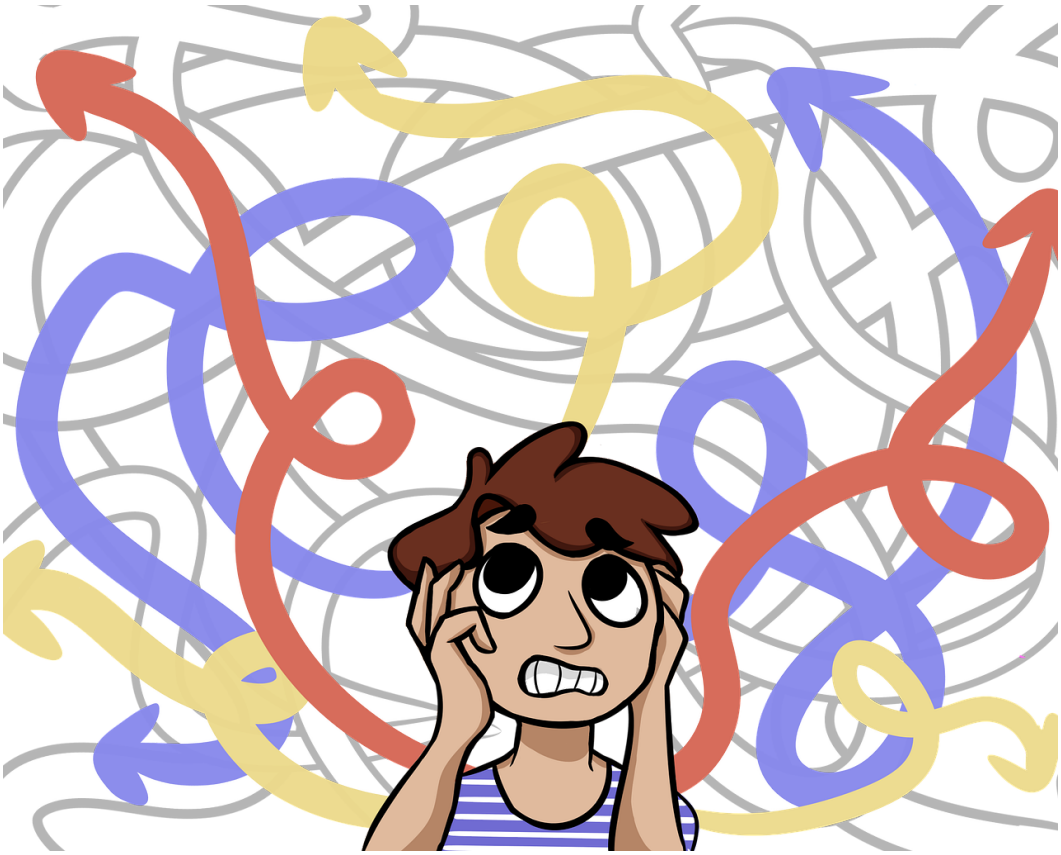
Introdução

Sumário

O que é lógica de programação?



É a ciência que estuda as leis e critérios de validade que regem o pensamento e a demonstração, ou seja, ciência dos princípios formais do pensamento.

A palavra lógica é originária do grego logos, que significa linguagem racional. Lógica é a análise das formas e leis do pensamento. Não se preocupa com a produção do pensamento, mas sim com a maneira pela qual um pensamento ou ideia é organizado e apresentado.

Exemplo:

1. Sandra é mais velha do que Ricardo.
2. Ricardo é mais velho que Pedro.

3. Logo, Sandra é mais velha do que Pedro.

Lógica aplicada à informática

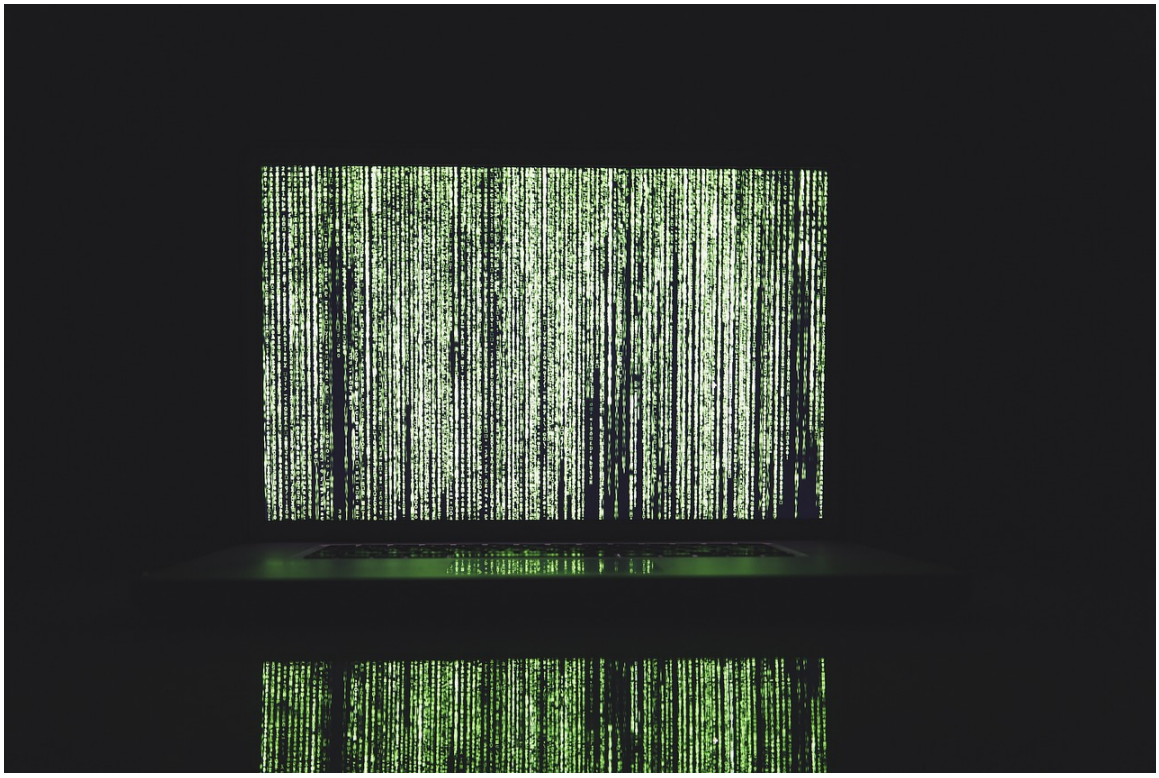
A lógica é utilizada em diversas ciências, principalmente na informática, onde pode ser encontrada em todas as suas áreas, desde **a criação do hardware até o software.**

É muito importante o uso da lógica para os profissionais de TI (desenvolvedores e analistas) que possuem como princípio entender a necessidade dos usuários e trazer uma solução rápida, eficiente e eficaz.

Na construção de software utilizaremos raciocínio lógico, onde serão construídos algoritmos, que serão transformados em programas de computador capazes de solucionar problemas complexos. É esta área específica que estudaremos ao longo desse curso

---

O que são algoritmos?



Algoritmo é uma sequência de passos que visam atingir um objetivo bem definido, representa um conjunto de regras para a solução de um problema.

Exemplos:

- Receita de um bolo
- Troca de um pneu
- resolução de uma equação do 2º grau

Importância da construção de um algoritmo

Um algoritmo tem por objetivo representar fielmente o raciocínio envolvido na lógica de programação. O estudo de algoritmos e lógica de programação é essencial no contexto do processo de criação de um software.

Uma vez concebida uma solução algorítmica para um problema, esta pode ser traduzida para qualquer linguagem de programação.

Tipos de algoritmos

- Descrição narrativa
- Pseudocódigo

- Gráficos:
  - Fluxograma
  - Nassin-Shneiderman-Chapin

Descrição narrativa

Utiliza linguagem natural para especificar os passos para a realização das tarefas.

Exemplo:

Algoritmo para calcular a área de um triângulo

Início

1. Pedir para o usuário digitar os valores de **b** e de **h**
2. Calcular a área de **s** usando a fórmula **s = (b X h)/2**
3. Exibir o valor de **s** na tela

Fim

Pseudocódigo ou português estruturado

Utiliza linguagem estruturada, assemelha-se a um programa escrito em uma linguagem de computador. Este é o tipo de algoritmo que usaremos durante o nosso curso.

Exemplo:

Início

1. Leia (b,h)
2. s ← (b * h)/2
3. Exiba (s)

Fim

Fluxograma (diagrama de blocos)

Forma universal de representação, pois utiliza figuras geométricas para ilustrar os passos a serem seguidos para a resolução de problemas.

Exemplo:

Algoritmo para calcular a área de um triângulo

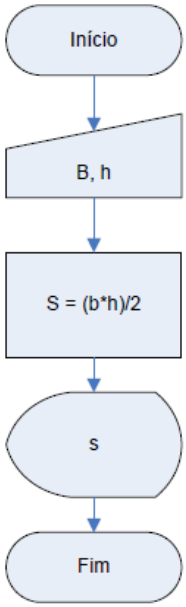
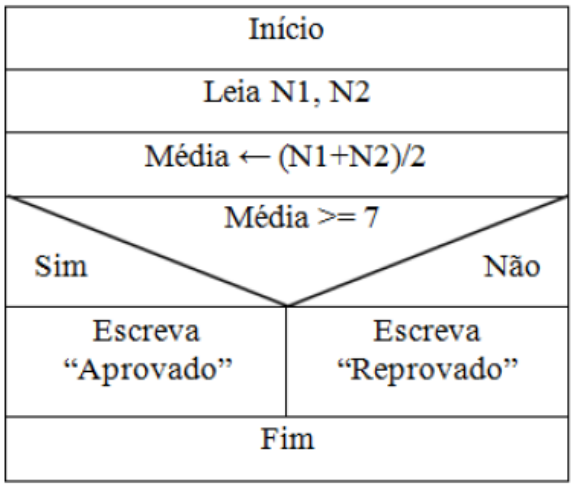


Diagrama de Nassin-Shneiderman-Chapin

A ideia básica deste diagrama é representar as ações de um algoritmo dentro de um único retângulo, subdividindo-o em retângulos menores, que representam os diferentes blocos de sequência de ações do algoritmo.



Boas práticas para construção de algoritmos - pseudocódigo

Identificação do algoritmo

Nome que representa o algoritmo.

Declaração de variáveis

Todas as variáveis utilizadas para a solução de problemas devem ser previamente declaradas.

Corpo do algoritmo

Área reservada para a solução do problema, deverão constar nesta área: entrada de valores para as variáveis, operações de atribuição, lógicas e matemáticas, abertura e fechamento de arquivos, laços de repetição e exibição de resultados.

```
Algoritmo Exemplo_1                              identificação do algoritmo
Variaveis
  nome, cargo : literal;
  idade, n_pessoas, tot_pessoas : inteiro;       declaração das variáveis
  salario : real;
Inicio
  n_pessoas <- 1
  tot_pessoas <- 0
  Enquanto (n_pessoas <= 50)Faca                 corpo do algoritmo
    Ler(nome, idade, cargo, salario);
      Se(idade <=30) e (salario >= 3000.00)Entao
        tot_pessoas <- tot_pessoas + 1;
  Fim-Enquanto
  Mostrar("O total de pessoas que atendem a condição é ", tot_pessoas);
Fim
```

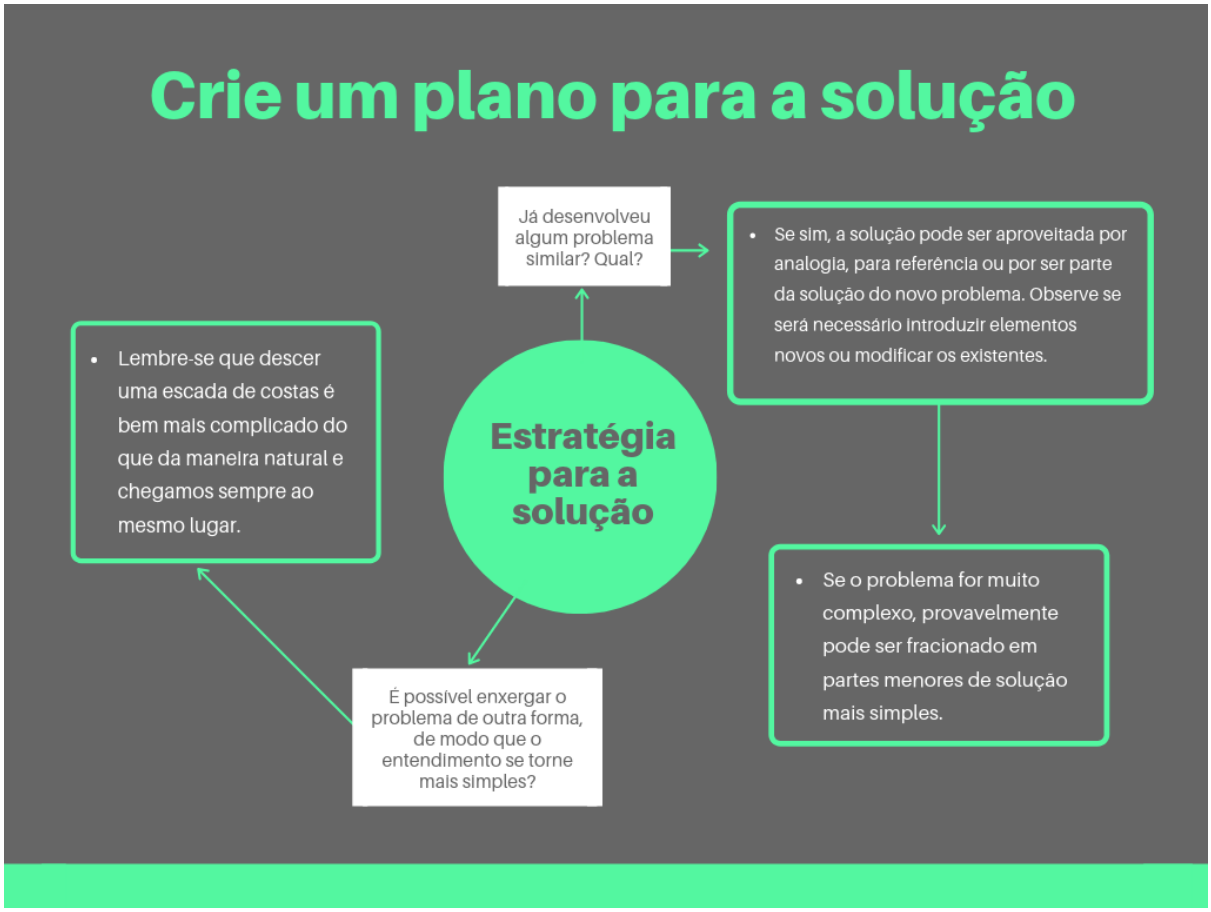
Solucionando problemas com algoritmos

Ao se deparar com um problema novo, tente entende-lo

- Faça um esboço informal de como ligar os dados às condições.
- É importante desenhar, rascunhar e usar de todos os artifícios para montar a "lógica" que leve a solução
- Essa é a primeira etapa, a análise do problema que consiste na compreensão e estratégia

Lembre-se que o algoritmo nada mais é que uma forma de representar a lógica que desejamos aplicar. Assim, antes de construir um algoritmo, precisamos definir alguma estratégia.

> Estratégia → Algoritmo → Programa



Formalize a solução

Não tente criar a solução inteira antes de rascunha-la.

▼ Siga os passos

1. Crie um **algoritmo informal** com as instruções que resolvam o problema ou que ao menos pareçam resolve-lo.
2. Verifique se cada passo desse **algoritmo está correto**, simulando-o num papel (teste de mesa ou simulação).
3. Identifique **os erros e os trate um por vez**, sem nunca perder de vista o objetivo real do programa. Esse processo ordenado logicamente faz você aprender a desenvolver soluções.

Quando se elabora um esboço da solução do problema, para em seguida ir-se refinando essa solução, até chegar a uma sequência básica de operações que resolva o problema, usou-se uma das principais técnicas relacionadas a construção de algoritmos, intitulada Top-Down.

> Essa técnica leva a geração de um pseudocódigo, que chamaremos de Português Estruturado.

O Português Estruturado, na opinião da maioria dos autores de livros de análise, programação ou algoritmos é muito mais eficaz que fluxogramas (estes são muito úteis em administração, que não é nosso caso).

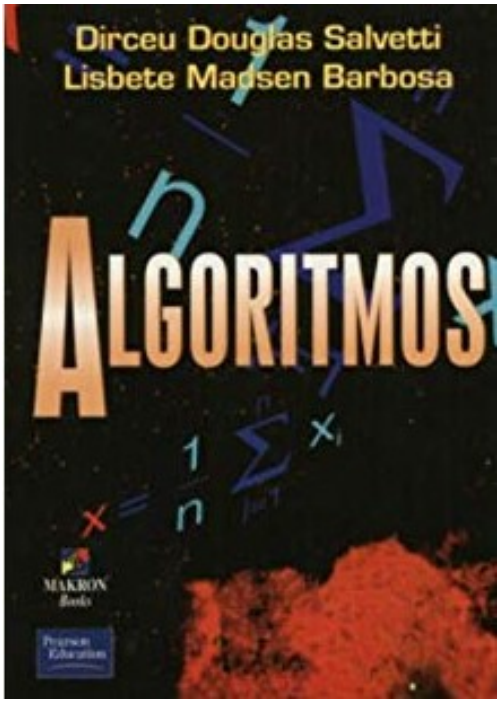
Exame dos resultados

- **Teste o algoritmo** com diversos dados de entrada e verifique os resultados (teste de mesa).
- Se o algoritmo não gerou resultado algum, o problema está na sua sintaxe e nos comandos utilizados. Volte e tente encontrar o erro.
- **Se o algoritmo gerou resultados, estes estão corretos**? Analise sua consistência. Se não estão corretos, alguma condição, operação ou ordem das operações está incorreta. Volte e tente encontrar o erro.

Otimização da solução

- É possível melhorar o algoritmo?
- É possível reduzir o número de passos ou dados?
- É possível conseguir uma solução ótima?

> O projeto do programa nada mais é que o algoritmo gerado, que visará otimizar o "binômio tempo-espaço , isto é, visando



> obter um programa que apresente um tempo de execução mínimo e com o melhor aproveitamento de espaço de memória"
> - Salvetti e Barbosa, 1998.

Implementação do programa

A implementação é sua codificação numa linguagem de programação. Em nosso curso o foco não é transcrever algoritmos em alguma linguagem específica, o objetivo principal é ensinar a base e a lógica de programação e desenvolvimento de algoritmos, e com isso aplicar no aprendizado de **qualquer** linguagem de programação.

Testes do programa

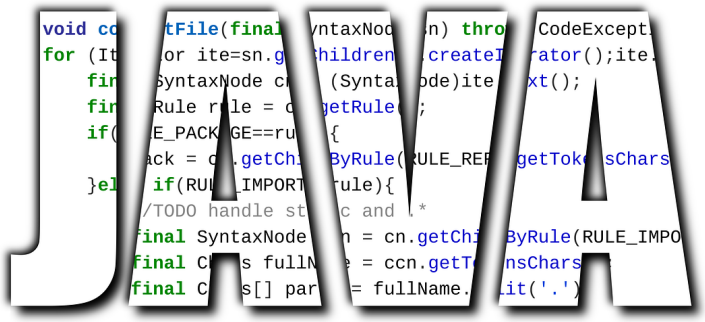
A fase de testes deve ser valorizada. Devemos lembrar que quem faz o algoritmo, raramente testa seus pontos deficientes. Isso ocorre porque se o analista tivesse percebido essas deficiências, provavelmente o programa não apresentaria esses problemas.

Negligenciar essa fase é fatal.

Verificação se o programa resolve todos os casos possíveis

A verificação do programa visa demonstrar que o algoritmo realmente resolve o problema proposto, qualquer que seja sua instância.

Linguagens de programação



- Os primeiros programas de computadores eram escritos em linguagem de máquina.
- Linguagem de máquina representa um conjunto de instruções que todo computador possui e, que seu processador é capaz de executar.
- Linguagem de programação se define em um conjunto de regras e convenções que são utilizadas parar escrever instruções em uma linguagem próxima da linguagem natural.
- O código fonte escrito em qualquer linguagem de programação dever ser traduzido para a linguagem de máquina.

▼ As linguagens de programação podem ser classificadas em alto nível e baixo nível.

  - Linguagens de alto nível foram desenvolvidas com o propósito comercial, facilitando e agilizando o processo de desenvolvimento.
    - Exemplos: Java, C#, Dephi, VB, PHP, etc.
  - Linguagens de baixo nível são linguagens onde as instruções são baseadas em linguagem de máquina.
    - Exemplo: Assembly.

- O processo de tradução pode ser por compilação ou interpretação:

1. Compilação o código fonte é convertido integralmente para código de máquina.
    - Exemplos: Java, C, C++, VB, Dephi, etc.
2. Interpretação o programa interpretador traduz cada linha do código fonte para a linguagem de máquina e o executa imediatamente.
    - Exemplos: JavaScript, ASP, Python, PHP, etc.

📩 Exercícios