

Algoritmos

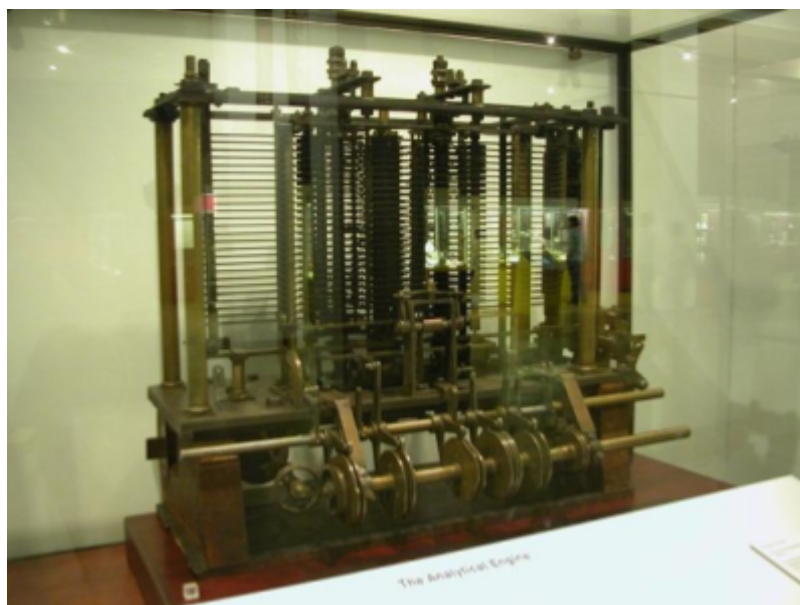
O que é Programação?

A tecnologia está cada vez mais presente no nosso cotidiano, não é mesmo? De redes sociais a transações bancárias, tudo envolve programação.



Porém, a programação continua sendo um mistério para muitos. Afinal, o que é programação e qual a sua importância?

Pode não parecer, mas a programação existe há muito tempo. A ideia do computador programável surgiu a partir da máquina analítica de Charles Babbage que, juntamente com Ada Lovelace, deu os primeiros passos na ideia de uma máquina que pudesse realizar cálculos matemáticos de acordo com as instruções pré-estabelecidas.



Máquina Analítica (Fonte da imagem: [Wikimedia Commons](#))

Desde então, a programação evoluiu gradativamente ao longo dos anos, tendo um crescimento acelerado nos últimos 50 anos com a criação de máquinas e linguagens cada vez mais complexas e capazes de realizar processos diversos.

Em outras palavras, podemos afirmar que:

“Programação é o processo que envolve a escrita, o teste e a manutenção de um programa para computadores, utilizando linguagem de programação, e que busca resolver um problema ou cumprir uma tarefa.”

Hoje a programação controla a nossa relação com o mundo, afetando o modo como vivemos não só na internet, mas também quando realizamos chamadas, quando fazemos exames médicos ou quando navegamos em nossos veículos.

A realidade é que grande parte da nossa vida depende de máquinas que, para funcionarem, precisam ser programadas por alguém. Isso porque, ao contrário do cérebro humano que possui uma inteligência própria, o computador necessita de instruções para funcionar.

Vamos entender um pouco mais sobre essas instruções no próximo módulo, bora lá?

Algoritmos

O que é Algoritmo?

Você já pensou que seu cotidiano é constituído por sequências de instruções e condições?

Quando falamos sobre algoritmos, é comum associá-los a um elemento tecnológico complicado e difícil de entender, mas que se encontra presente em tudo que fazemos. Quem nunca ouviu falar que uma coisa é culpa do algoritmo ou que devido a ele, só consegue ver um determinado gênero de filmes no Netflix?

A realidade é que o algoritmo não é um bicho de sete cabeças que todos falam. Ele, em sua melhor definição, é um conjunto de instruções que tem como objetivo resolver um problema ou realizar uma ação. Ainda não conseguiu visualizar? Vamos usar exemplos simples:

- 1. Uma receita de bolo:** Junte os ingredientes, mexa tudo até formar uma massa homogênea e asse o bolo até dourar;
- 2. Um manual de instruções de um aparelho eletrônico:** Verificação da voltagem, visão geral do aparelho e primeiros passos para seu funcionamento.
- 3. O trajeto de sua casa até o trabalho:** Descreva seu modo de deslocamento (a pé, ônibus, carro), percorra o caminho até chegar ao destino final.

Em nosso cotidiano, para realizar qualquer tarefa, desempenhamos ações organizadas em uma sequência lógica. A programação de um software é muito parecida com as demais atividades do seu dia a dia:

“é preciso saber quais atividades deverão ser realizadas antes de partir para a ação.”

Então, sabendo qual problema seu programa deverá resolver, você irá analisar e identificar todas as etapas necessárias para chegar a essa solução e só depois começará a programar o computador!

Algoritmos

O que é Lógica de Programação?

No módulo anterior aprendemos um pouco sobre a Lógica e como a utilizamos no dia-a-dia. Vimos que, inconscientemente, utilizamos algoritmos para a realização de tarefas simples até as mais complexas.

Aprendemos que é preciso ter ordem e sequência na execução dos passos para conseguirmos concluir uma tarefa, mas afinal, o que é lógica de programação e qual a sua importância para se tornar um programador?



Vimos que a lógica nos ajuda a organizar os pensamentos de forma coesa, nos permitindo então, criar e executar o processo inicial no desenvolvimento de uma aplicação: a criação de seu algoritmo. A lógica é o primeiro conhecimento que devemos ter quando decidimos nos tornar um desenvolvedor, porque é a partir dele que todo o restante fará sentido.

Aprender lógica de programação é essencial, pois cada linguagem tem as suas particularidades, como por exemplo sua sintaxe, seus tipos de dados e sua orientação, porém a lógica por trás de todas é a mesma. Ter o conhecimento sólido em lógica fará com que você se torne um programador completo.

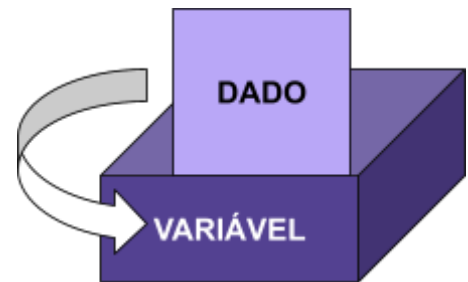
Algoritmos

Variáveis e constantes

Durante os módulos que estudamos, você percebeu que até o momento, não falamos sobre armazenamento de dados, vimos apenas sobre procedimentos, como um algoritmo.

Nesse módulo vamos falar um pouco sobre valores variáveis e fixos, no caso as estruturas de variáveis e constantes.

Durante o desenvolvimento de nossos algoritmos, por vezes, será necessário que guardemos alguns dados no computador para que os utilizemos depois, sempre que necessário. Esses dados ficarão armazenados na memória do computador, em locais específicos, e para que possamos manipulá-los, deveremos fazer uso das variáveis, que vão se encarregar de armazená-los.



As variáveis também podem receber nomes, chamados “identificadores”, para referenciar estas posições de memória de forma simples enquanto desenvolvemos algoritmos. É muito importante que estes identificadores sejam claros e relacionados ao tipo de valor que desejamos armazenar nelas, para que outras pessoas possam ler e compreender o funcionamento deste algoritmo.

No caso das variáveis esse valor que será armazenado pode se alterar, já nas constantes, esse valor é fixo.

Por exemplo:

- O nome dos clientes de uma loja, esse valor muda de pessoa para pessoa (**variável**)
- O valor de PI (3.14159265359...) que nunca vai mudar, é fixo (**constante**).

Podemos então, definir as constantes como um local na memória do computador que também armazena um dado, porém esse valor não se altera ao longo da execução do programa.

Algoritmos

Formas de Representação

Fluxograma

Para facilitar a visualização dos algoritmos, podemos utilizar um método visual chamado Fluxograma. Esse método fornece padronização na representação, maior rapidez para descrever tarefas e melhor visualização. Vamos conhecer os símbolos que são mais utilizados para construir fluxogramas:


Início e Fim

*Início**FIM*

Processo e operações

*(nota1 + nota2 + nota3) / 3*

Condição ou decisão

*idade é maior ou
igual a 16?*

Pseudocódigo

Já vimos nas aulas anteriores as formas de representação de algoritmos utilizando o Fluxograma, agora, iremos conhecer um modelo de representação chamado pseudocódigo.

Pseudocódigo é a escrita mais próxima da linguagem de programação, nele não utilizamos nenhuma informação técnica da linguagem, mas escrevemos de uma maneira bem parecida com as instruções computacionais.

Uma de suas principais vantagens é que tal representação não precisa seguir as mesmas regras de uma linguagem de programação específica, dessa forma, transferi-lo para qualquer uma delas torna o processo mais simples. Por outro lado, regras são regras e, até mesmo para o pseudocódigo, elas precisam ser aprendidas.

A seguir, uma representação de algoritmo escrito em pseudocódigo.

Algoritmo 1 - Exemplo de multiplicação de duas variáveis em pseudocódigo

```
ALGORITMO_INICIO           // Início do Algoritmo

    DECLARE x, y, m         // Declaração das variáveis

    LEIA n1, n2             // Leitura do valor das variáveis n1 e n2

    z ← x * y               // Variável z recebe a multiplicação entre x e y

    ESCREVA z               // Exibição do valor da variável z

FIM_ALGORITMO              // Fim do algoritmo
```

Uma boa prática, que precisamos adotar desde cedo, é comentar o código gerado. No exemplo anterior, notamos na mesma linha, após o código, temos “//”, neste caso, tudo o que vier após a barra dupla não será lido pelo programa. Em outras palavras, é possível deixar comentários para todos os programadores que lerem nossos códigos, inclusive nós mesmos!

Algoritmos

Operadores Aritméticos

Todos nós já utilizamos operadores aritméticos na escola, quando íamos desenvolver uma operação matemática, certo? Em algoritmos também utilizamos esses operadores. Eles são simples e também têm a mesma simbologia em linguagens de programação.

Alguns desses operadores mais comuns são (+, -, *, /), porém existem outros como o incremento ++ e o módulo % que não são tão reconhecidos assim.

Os operadores aritméticos ou sinais, como temos o costume de falar, são meios pelos quais podemos somar, subtrair, incrementar e decrementar os dados dentro do nosso código. Vamos explorar alguns desses sinais e entender melhor sobre o seu funcionamento dentro do nosso código.

Operador	Significado	Descrição	Exemplo
+	Soma	Operador de adição, tem como objetivo a soma ou união de dois valores para criar um novo valor.	10 + 2 //retorna 12
-	Subtração	Operador de subtração, tem como objetivo calcular a diferença entre dois números.	6 - 3 //retorna 3
*	Multiplicação	Operação de multiplicação entre dois números.	3 * 6 // retorna 18
/	Divisão	Operação que consiste em descobrir quantas vezes um número está contido em outro.	25 / 5 // retorna 5
%	Módulo	Diferente da divisão, ele devolve o resto de uma divisão inteira.	10 % 2 //retorna 0
++	Incremento	O operador incrementa um ao seu número.	valor = 2 valor++ //retorna 3
--	Decremento	O operador subtrai um do seu número atual.	valor = 4 valor-- //retorna 3

Tabela 1 - Lista de Operadores Aritméticos

Algoritmos

Operadores Relacionais

Para a construção de nossos algoritmos, assim como os operadores aritméticos, os quais vimos anteriormente, faremos uso também dos chamados Operadores Relacionais.

Os operadores relacionais são responsáveis por efetuar comparações entre valores, com o objetivo de mostrar ao programa, como prosseguir a partir da resposta apresentada.

Esses operadores vão resultar em duas respostas lógicas do tipo verdadeiro ou falso, conforme as diferentes combinações entre seus operandos. Vejamos a seguir, uma tabela com alguns destes nossos operadores.

Operador	Descrição	Exemplo
==	igual (retorna verdadeiro se os operandos forem iguais)	3 == 3 (retorna verdadeiro) 3 == 8 (retorna falso)
!=	diferente (retorna verdadeiro se os operando não forem iguais)	5 != 2 (retorna verdadeiro) 5 != 5 (retorna falso)
>	maior que (retorna verdadeiro se o operando esquerdo for maior que o operando direito)	10 > 6 (retorna verdadeiro) 10 > 30 (retorna falso)
>=	maior ou igual (retorna verdadeiro se o operando esquerdo for maior ou igual ao operando direito)	4 >= 4 (retorna verdadeiro) 4 >= 20 (retorna falso)
<	menor que (retorna verdadeiro se o operando esquerdo for menor que o operando direito)	2 < 9 (retorna verdadeiro) 2 < 0 (retorna falso)
<=	menor ou igual (retorna verdadeiro se o operando esquerdo for menor ou igual ao operando direito)	8 <= 10 (retorna verdadeiro) 8 <= 7 (retorna falso)

Tabela 2 - Lista de Operadores Relacionais

Algoritmos

Operadores lógicos

Além dos operadores aritméticos e relacionais, também vamos utilizar os operadores lógicos para podermos criar os nossos algoritmos. Vamos entender um pouco como eles funcionam?

Esses operadores nos permitem montar expressões lógicas que validam duas ou mais expressões e também nos retornam duas respostas que já conhecemos, verdadeiro ou falso.

Os operadores são E (representado por “&&”), OU (representado por “||”) e NÃO (representado por “!”).

Nome do Operador	O que ele faz
não	Negação
e	Conjunção
ou	Disjunção

Tabela 3 - Lista de Operadores Lógicos

A utilização dos operadores lógicos requer alguns recursos, como no caso da chamada Tabela Verdade. Existe uma tabela para cada tipo de operador, nela são apresentados os resultados das diferentes combinações que poderão ocorrer entre os operandos.

Resumindo de maneira simples, no uso do operador “não” o resultado sempre será o contrário do operando de entrada; já o operador “e” precisará que todos os operandos de entrada sejam do tipo verdadeiro para que seus resultados sejam do tipo verdadeiro também e; por fim, no uso do operador “ou” basta apenas que um de seus operandos de entrada seja do tipo verdadeiro para que seu resultado seja também do tipo verdadeiro.

Vejamos a seguir as tabelas para cada operador.

Operador “E”

Operando 1	Operando 2	OP1 “e” OP2
F (Falso)	F (Falso)	F e F = F (Falso)
F (Falso)	V (Verdadeiro)	F e V = F (Falso)
V (Verdadeiro)	F (Falso)	V e F = F (Falso)
V (Verdadeiro)	V (Verdadeiro)	V e V = V (Verdadeiro)

Tabela 4 - Tabela Verdade do Operador “e”

Operador “OU”

Operando 1	Operando 2	OP1 “ou” OP2
F (Falso)	F (Falso)	F ou F = F (Falso)
F (Falso)	V (Verdadeiro)	F ou V = V (Verdadeiro)
V (Verdadeiro)	F (Falso)	V ou F = V (Verdadeiro)
V (Verdadeiro)	V (Verdadeiro)	V ou V = V (Verdadeiro)

Tabela 5 - Tabela Verdade do Operador “ou”

Operador “NÃO”

Operando	“Não” Operando
F (Falso)	V (Verdadeiro)
V (Verdadeiro)	F (Falso)

Tabela 6 - Tabela Verdade do Operador “não”

Algoritmos

Precedência de Operadores

Nos módulos anteriores, conhecemos alguns operadores que vão nos ajudar na criação de nossos algoritmos. Mas será que existe alguma regrinha na hora de utilizá-los?

Quando temos mais de um operador sendo utilizado em uma mesma instrução, precisamos nos atentar à **precedência dos operadores**. A precedência se refere à prioridade dada ao operador quando a instrução for analisada, ou seja, algumas operações serão efetuadas primeiro.

Por exemplo, na expressão $2 + 2 * 5$, o resultado é 12 e não 20, porque o operador de multiplicação (*) tem prioridade de precedência que o operador de adição (+). Todos os operadores que aprendemos possuem regras de precedência, vamos ver cada um deles.

1. Parênteses

Se houver, resolver do mais interno para o mais externo

2. Operadores Aritméticos

Multiplicação ou divisão (o que vier primeiro)

Soma ou subtração (o que vier primeiro)

3. Operadores Relacionais

Todos os operadores de comparação têm precedência igual, ou seja, eles são avaliados na ordem da esquerda para a direita na qual aparecem.

4. Operadores Lógicos

Operador de Negação (!)

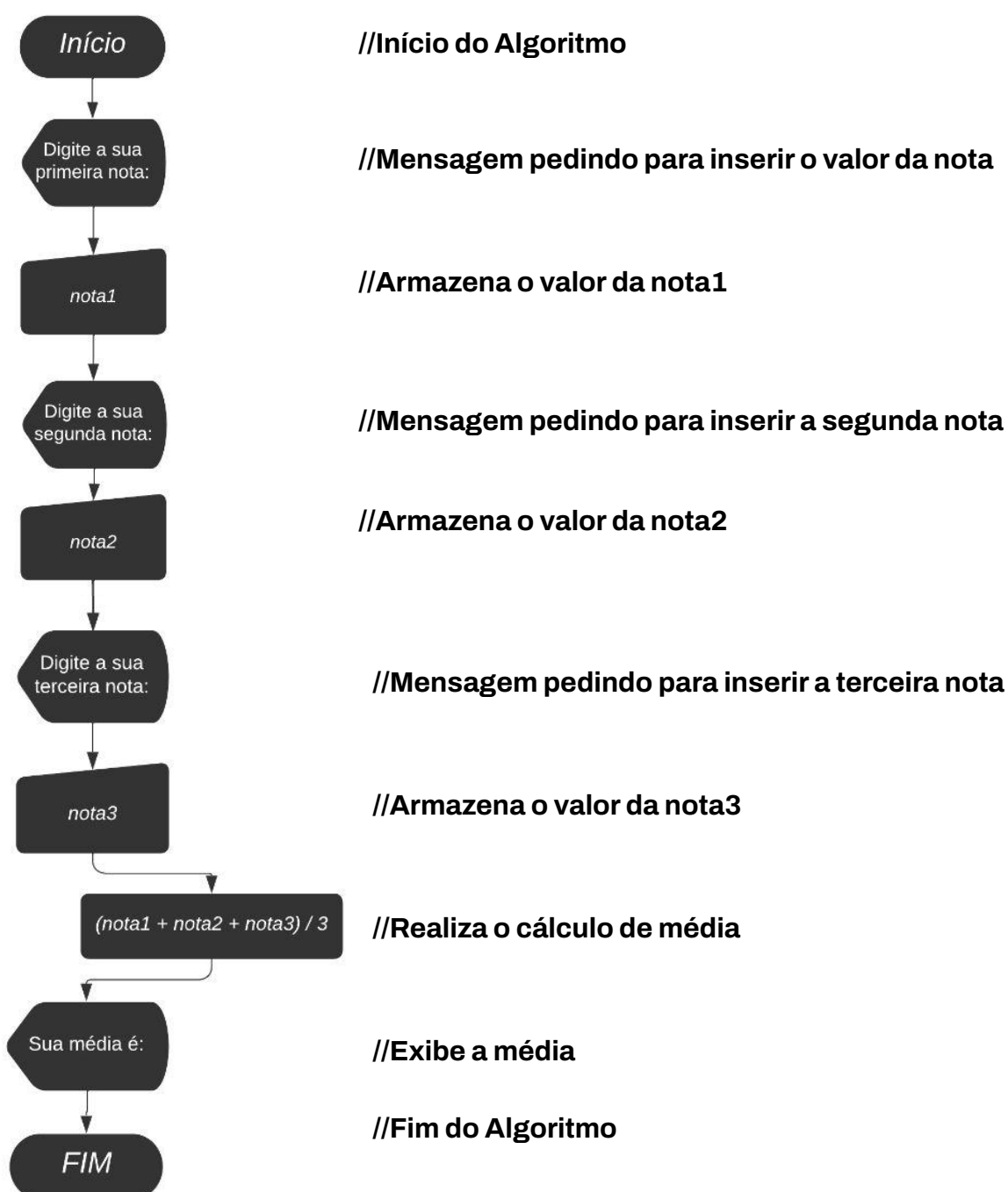
Operador de Conjunção (&&)

Operador de Disjunção (||)

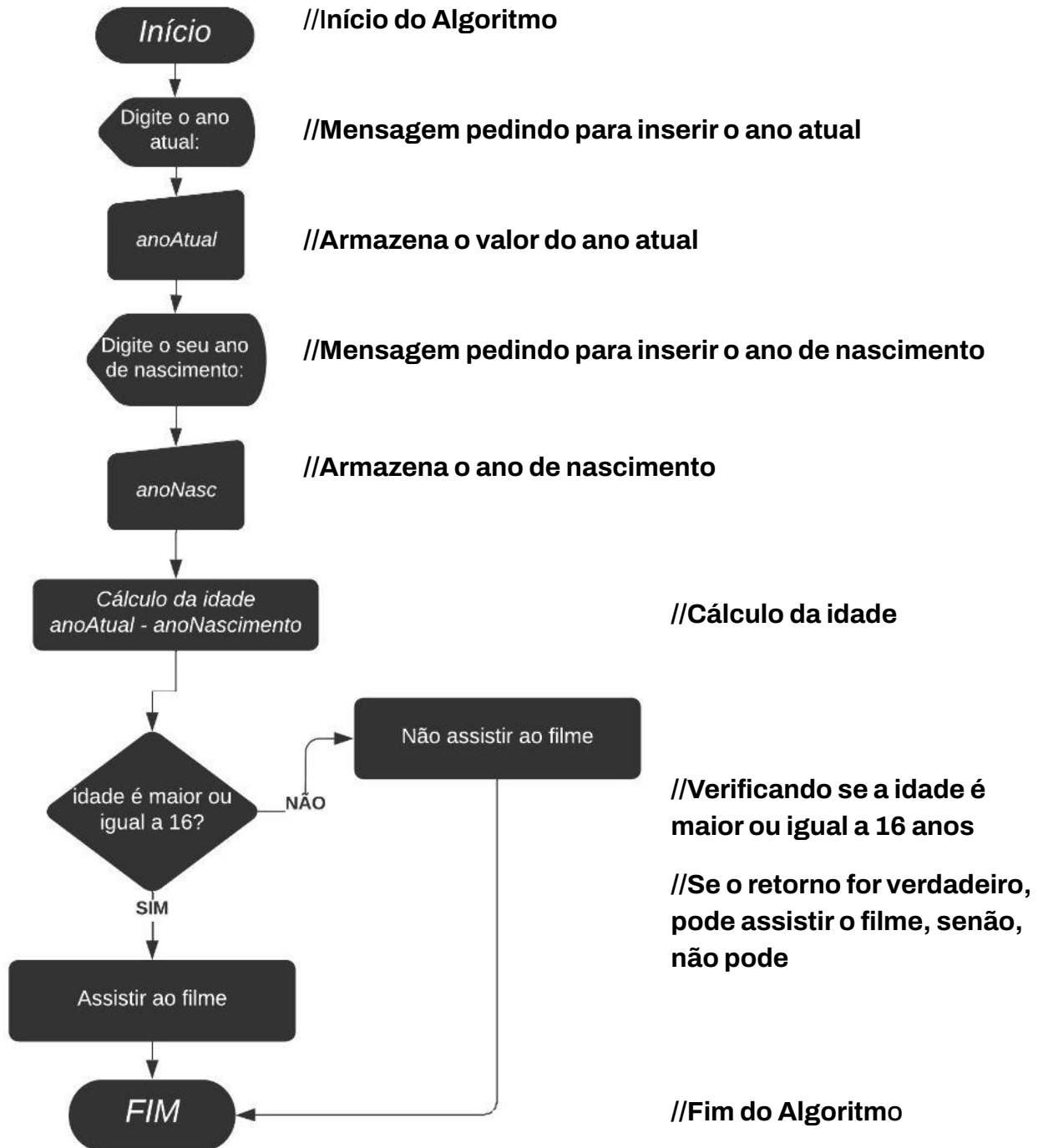
Algoritmos

Problemas Lógicos com Fluxograma

1. É semana de prova em uma escola e todos os alunos estão ansiosos para saber as suas médias. Vamos ajudá-los a calcular?



2. Um filme que você estava muito ansioso para assistir, finalmente vai estrear nos cinemas. Porém, o filme tem a idade classificatória de 16 anos. O cinema precisa de um algoritmo que calcule a idade de seus clientes para poder liberar a entrada. Vamos ver a resolução desse problema através de um fluxograma.



Algoritmos

Reescrevendo com Pseudocódigo

Agora, vamos reescrever os problemas vistos anteriormente, dessa vez utilizando o Portugol, também conhecido como Português estruturado.

1. É semana de prova em uma escola e todos os alunos estão ansiosos para saber as suas médias. Vamos ajudá-los nesse cálculo?

```
1  programa {
2      funcao inicio() {
3          inteiro nota1
4          escreva("Digite a sua primeira nota: ")
5          leia(nota1)
6
7          inteiro nota2
8          escreva("Digite a sua segunda nota: ")
9          leia(nota2)
10
11         inteiro nota3
12         escreva("Digite a sua terceira nota: ")
13         leia(nota3)
14
15
16         inteiro media = (nota1 + nota2 + nota3) / 3
17         escreva("Sua média é: ", media)
18     }
19 }
20
```

2. Um filme que você estava muito ansioso para assistir, finalmente vai estrear nos cinemas. Porém, o filme tem a idade classificatória de 16 anos. O cinema precisa de um algoritmo que calcule a idade de seus clientes para poder liberar a entrada. Vamos ver a resolução desse problema através de um fluxograma.

```
1  programa {
2      funcao inicio() {
3          inteiro anoAtual
4          escreva("Digite o ano atual: ")
5          leia(anoAtual)
6
7          inteiro anoNasc
8          escreva("Digite seu ano de nascimento: ")
9          leia(anoNasc)
10
11         inteiro idade = anoAtual - anoNasc
12
13         se(idade >= 16){
14             escreva("Você pode assistir ao filme!")
15         }senao{
16             escreva("Sinto muito! Você não pode assistir ao filme!")
17         }
18     }
19 }
20
21 }
22
```


Exercícios

1 - Dado os pseudocódigos abaixo, montar o fluxograma equivalente para cada item.

a) Algoritmo para calcular quanto se vai pagar no frete de um determinado produto, sendo que a empresa cobra 1.50 o km e taxa de envio de R\$ 9,00.

```
1 /*
2   Algoritmo para calcular valor a pagar no frete de um determinado produto,
3   sabendo que a empresa cobra 1.50 o km e taxa de envio de R$ 9,00.
4 */
5
6 programa {
7   funcao inicio() {
8
9       real kmrodado, taxa, valor_frete
10
11       escreva ("Digite os km rodados: ")
12       leia (kmrodado)
13
14       taxa = 9.0
15
16       valor_frete = (kmrodado * 1.50) + taxa
17
18       escreva ("O valor do seu frete é ", valor_frete)
19   }
20 }
21 }
```

b) Algoritmo para saber qual o IMC (Índice de Massa Corporal) de uma pessoa.

```
1 /*
2   Algoritmo para saber qual o IMC (Índice de Massa Corporal) de uma pessoa.
3 */
4
5 programa {
6   funcao inicio() {
7
8       real altura, imc, peso
9
10      escreva ("Digite sua altura: ")
11      leia (altura)
12
13      escreva ("Digite seu peso: ")
14      leia (peso)
15
16      imc = peso / (altura * altura)
17
18      escreva ("O seu IMC é ", imc)
19   }
20 }
21 }
```

c) Algoritmo que ler dois valores e efetua as trocas dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A.

```
1 /*
2     Algoritmo que leia dois valores e efetue a troca dos valores de forma que
3     a variável A passe a possuir o valor da variável B e
4     a variável B passe a possuir o valor da variável A.
5 */
6 programa {
7     funcao inicio() {
8         inteiro a, b, troca
9
10        escreva ("Digite o valor de A: ")
11        leia (a)
12
13        escreva ("Digite o valor de B: ")
14        leia (b)
15
16        troca = a
17        a = b
18        b = troca
19
20        escreva ("\n0 novo valor de A é: ", a)
21        escreva ("\n0 novo valor de B é: ", b)
22
23    }
24 }
```

2 - Agora, analise o fluxograma e escreva o algoritmo correspondente em Portugol:

